INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 4, PP. 1105–1112
Manuscript received October 15, 2024; revised October, 2024. doi: 10.24425/ijet.2024.152513

The ASL Dataset for Real-Time Recognition and Integration with LLM Services

Michał Chwesiuk, and Piotr Popis

***Abstract*—This study aims to investigate the impact of hand gesture recognition techniques on the efficiency of American Sign Language (ASL) interpretation, addressing a critical gap in the existing literature. The research seeks new insights into the challenges of automated sign language recognition, contributing to a deeper understanding of accessibility in communication for the deaf and hard-of-hearing community. The study employs a quantitative approach, using a dataset comprising hand gesture images representing the static letters of the ASL alphabet collected from multiple users. Data were collected from various individuals to ensure diversity and analyzed using machine learning models to evaluate their effectiveness in recognizing ASL signs. The results reveal that the machine learning models implemented achieved a high accuracy rate in recognizing hand gestures, indicating that person-specific variations do not significantly hinder performance. These findings provide evidence that the proposed dataset and methodologies can improve the reliability of sign language recognition systems, offering significant implications for the development of more inclusive communication technologies. This research offers a novel perspective on sign language recognition, providing valuable insight that extends the current understanding of gesture-based communication systems. The study's findings contribute to advancements in accessibility technologies, highlighting areas for future research and practical applications in improving communication for the Deaf and hard of hearing community.**

***Keywords*—dataset; gesture recognition; hand landmarks; machine learning; human-computer interaction; american sign language**

I. Introduction

THE American Sign Language (ASL) is a vital communication tool for the Deaf and hard of hearing communities (DHH), which rely heavily on hand gestures to convey letters, words, and sentences. As technology advances, there is an increasing demand for systems that can accurately interpret sign language through automated hand gesture recognition, enabling accessibility in Human-Computer Interactions (HCI) methods. Machine learning, particularly in the field of computer vision, offers promising solutions to this challenge by enabling the development of models that can recognize and translate these gestures into text or speech in real time.

This paper introduces a novel dataset that focuses on hand gestures representing the ASL alphabet. The collected images are structured hierarchically, where hand gestures for each letter of the ASL alphabet are organized by individual users. Each user contributes a complete set of images representing the 26 letters, offering a diverse range of hand gestures that account for person-specific variations in sign language expression. Prepared collection procedure provide option for remote acquisition of additional set of images.

The dataset structure allows for training recognition algorithms based on rule-based rules, as well as machine learning algorithms, not only to recognize individual ASL letters but also to generalize across different users, improving the robustness and adaptability of the models in real-world scenarios.

In this study, we explore the structure and potential applications of this dataset, providing insights into how it can be used to develop models that improve the accessibility and usability of ASL recognition systems. Furthermore, we discuss the implications of using person-specific data for improving model generalization, addressing the challenges and opportunities in the field of gesture recognition.

Authors are with Warsaw University of Technology, Poland (e-mail: Michal.Chwesiuk@pw.edu.pl, 01186032@pw.edu.pl).

II. Related Works

Gesture recognition for ASL has garnered significant attention in the field of machine learning and computer vision. Various studies have contributed to the development of datasets, algorithms, and methodologies to improve the accuracy of gesture recognition systems. One prominent contribution is the MS-ASL dataset, introduced by Vaezi Joze and Koller [1]. This large-scale dataset serves as a benchmark for understanding ASL and is crucial for training models to recognize static and dynamic gestures. The comprehensive nature of this dataset allows for robust evaluation and comparison of different machine learning approaches, setting a foundation for further advancements in ASL recognition. Second well known dataset is MNIST [2] that offers images in size 28x28, but has large variation in the number of samples per class, which may be leading to convergence in predictions. In a recent study, Abdulhussein and Raheem [3] focused on static hand gestures of ASL using deep learning techniques. Their work demonstrates the effectiveness of convolutional neural networks in classifying static letters, indicating that deep learning models can significantly enhance gesture recognition performance when sufficient training data is available. Zaki and Shaheen [4] explored a combination of new vision-based features for sign language recognition. Their research highlights the importance of feature extraction in improving recognition rates, employing various techniques such as principal component

analysis and hidden Markov models. This study underscores the need for innovative approaches to capture the nuances of hand gestures effectively. Furthermore, Núñez-Marcos et al. [5] conducted a survey on sign language machine translation, discussing the integration of gesture recognition systems with translation algorithms. Their findings emphasize the potential of combining recognition systems with natural language processing techniques to facilitate real-time communication for the hearing impaired, expanding the applicability of gesture recognition technologies beyond classification. Similar study [6], where translation is focused on real-time automatic speech recognition rather than video gestures introduces app that enable users to communicate with different languages, study breaks the barrier in communication between people knowing same language. Article [7] is the next study in which authors adress the barrier of communication between deaf community and current LLM models widely used in many types of work, study also takes the step of processing motion-based phases with LSTM and mediapipe [8], but does not aim for communication with LLM-based popular services. Other study by Vadalia and Chilukala [9] also presents a system for real-time word-level ASL translation in to text, authors using tensorflow [10] models, their work focuses on high accuracy and speed, in comparision our approach focuses on building sequences from captured letters, and another factor is that we introduces Python based web interface connected with that is available for every computer. Other study which approaches ASL recognition and speech to text translation by Chinmay Bhat and Vanita Agarwal [11] where authors where focused on the various languages so not only ASL, but 300 different sign-languages, application aims also to communicate signers and non-signers using computer vision and text to speech methodology through human-computer interaction.

These studies collectively illustrate the advancements in the field of ASL gesture recognition, highlighting the need for continued research to refine methodologies and improve system performance, especially in light of the challenges posed by limited training data.

III. Methodology

This section outlines the main stages involved in the proposed real-time ASL interpretation (Fig. 1). It consists of three main modules - identifying ASL letters from hand gestures, transformation of the letters sequence into readable text image classifier and sending request to external LLM model. In this work, we will focus on integration with the ChatGPT service widely used as artificial intelligence chatbot developed by the OpenAI. Each module will be discussed separately in the corresponding section below, involving used strategies for classification and corrections.

A. Gesture Classification

The first part of the system involves detection and classification of the ASL hand gestures captured through a collected image (via webcam or external camera) into individual letters. For this task, we experimented with two different methods: hand landmarks detection with Random Forest Classification, and classifier based on Convolutional Neural Network (CNN) architecture.

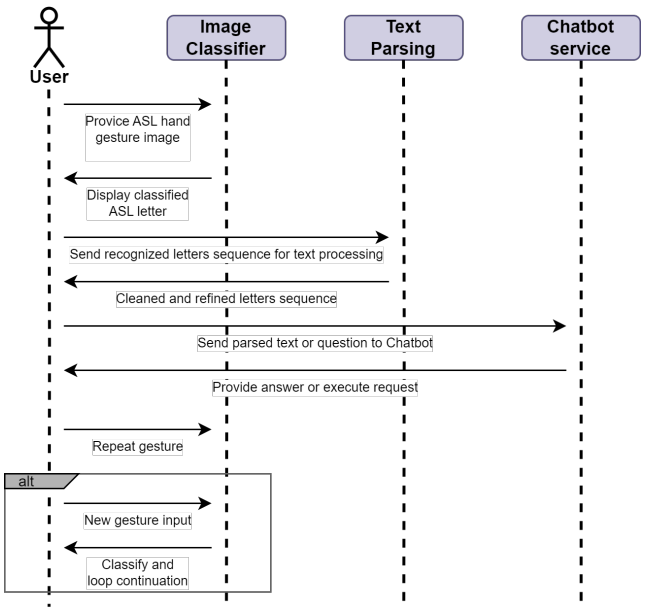
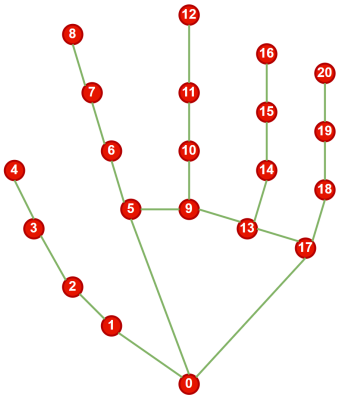


Fig. 1. Diagram of ASL Interpretation System workflow. The arrow heads points to data transfer direction between modules and user of the system. The additional alternative module contains functionality that allows adding new gestures to the classifier.

1) Hand Landmark Detection: For real time hand landmark detection, we utilized MediaPipe framework, a machine learning based method developed by Google [12]. MediaPipe's Hand module is designed to efficiently detect and track 21 key points on the human hand across a sequence of images or video frames (Fig. 2). The software leverages a pipeline architecture, combining a deep neural network for hand region detection with a regression model that estimates landmark positions. This approach allows for high-precision tracking, making it suitable for applications in gesture recognition and human-computer interaction (fig. 3). The framework's cross-platform compatibility and real-time performance were essential for our application.

2) Random Forest Classifier: This algorithm works by creating an ensemble of decision trees to classify input hand gestures into the corresponding ASL letters. Despite being a relatively simple model, the Random Forest classifier has shown moderate success in classifying static ASL letters.

3) Convolutional Neural Network (CNN): The CNNs have been proven to be highly effective in sign language recognition, including ASL and Indian Sign Language (ISL) gestures, as they can learn spatial hierarchies of features from input images [13]–[15]. To validate dataset presented in this work and prove this thesis, we used LeNet and ResNet [16] CNN architecture to identify ASL hand gestures [17]. It has a relatively simple architecture compared to modern used CNNs, which makes it computationally efficient and better suited for real-time related tasks (Fig. 4).



0	WRIST		
1	THUMB_CMC	11	MIDDLE_FINGER_DIP
2	THUMB_MCP	12	MIDDLE_FINGER_TIP
3	THUMB_IP	13	RING_FINGER_MCP
4	THUMB_TIP	14	RING_FINGER_PIP
5	INDEX_FINGER_MCP	15	RING_FINGER_DIP
6	INDEX_FINGER_PIP	16	RING_FINGER_TIP
7	INDEX_FINGER_DIP	17	PINKY_MCP
8	INDEX_FINGER_TIP	18	PINKY_PIP
9	MIDDLE_FINGER_MCP	19	PINKY_DIP
10	MIDDLE_FINGER_PIP	20	PINKY_TIP

Fig. 2. Hand landmarks detected by MediaPipe Tracking module, which include key points for each finger and the wrist.

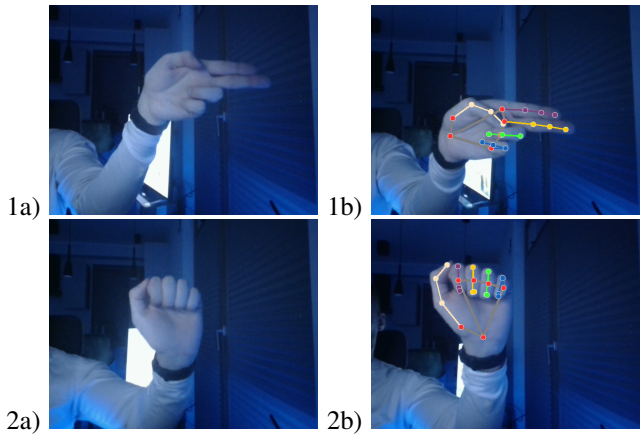


Fig. 3. Pairs of hand images with corresponding hand landmarks detected using the MediaPipe framework.

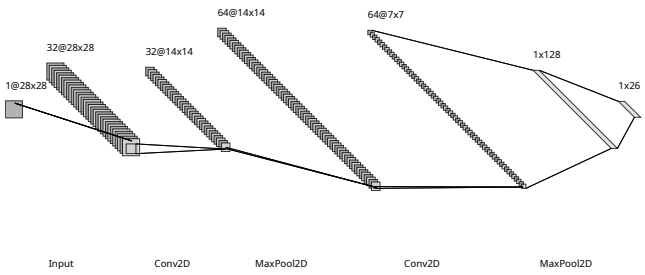


Fig. 4. LeNet CNN, generated with NN-SVG tool [17].

B. Transforming Letter Sequences into Human-Readable Text

Once the ASL letters are recognized from the frames, the next step involves transforming these sequences into readable sentences. Due to the inherent noise and inconsistencies in the gesture recognition process, raw letter sequences often contain errors, such as repeated characters or incorrect letter assignments. To address this, we tested several text processing strategies to clean and refine noisy letter sequences before querying the LLM service.

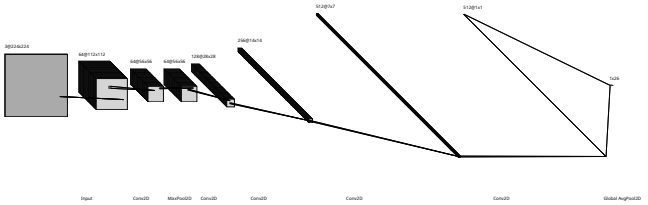


Fig. 5. ResNet18 DNN architecture, generated with NN-SVG tool.

1) Basic Strategies for Text Processing: The following text processing strategies were implemented to denoise letter sequence after ASL recognision and improve the accuracy of target sentences:

- **Phonetic Correction Strategy:** This strategy uses phonetic algorithms like Soundex to correct letters based on their phonetic similarity [18]. While this method attempts to match letters to likely sounds, it struggled with the unique structure of ASL and was not highly effective.
- **Remove Repetitions Strategy:** This strategy removes consecutive repeated characters from the letter sequence. It proved useful in reducing noise but did not address errors related to incorrect letter assignments.
- **Autocorrection Strategy:** Leveraging automatic correction techniques, this method attempts to correct individual words within the sequence using a predefined dictionary of words. Although it helps with simple errors, it fails to handle context and sentence structure effectively.
- **Levenshtein Correction Strategy:** This approach uses the Levenshtein distance to find the closest match to misspelled words from a corpus of valid words [19]. Although it performs well in some cases, it does not adequately account for context or grammar.
- **Majority Vote Strategy:** This strategy smooths the sequence of letters by using a sliding window to select the most frequent letter within the window. It reduces some noise but fails to consistently produce coherent sentences.
- **LLM Strategy:** This strategy leverages the GPT-4o language model included in ChatGPT service to correct noisy text sequences and transform them into coherent and grammatically correct sentences. After applying the initial noise reduction and preprocessing steps, the text is passed to LLM for final corrections.

2) Combined Strategies for Text Processing: Among all tested strategies, the LLM correction approach produced the most acceptable results (tab. III). ChatGPT has proven its ability to understand context and correct unreadable letter sequences, transforming the output into readable and coherent sentences. After preprocessing the letter sequences with one or more of the strategies listed above, LLM strategy was able to accurately interpret and refine the text. This approach yielded the best results for the transformation of the ASL sequences

into human-readable sentences, making it a critical final step in the pipeline.

To improve the overall quality of the output before sending it to the Chatbot, we conducted additional tests that involved combination of multiple preprocessing strategies in a sequence. Each combination aimed to reduce noise and errors, allowing the LLM to focus on refining the sentence structure rather than correcting fundamental errors. Below are a few examples of the combined pipelines:

1) **Pipeline 1:**
   Remove Repetitions + Auto-Correction + LLM
   This combination first removes repeated characters, then applies autocorrection to fix basic spelling errors, before sending the text to LLM for contextual and grammatical correction.
2) **Pipeline 2:**
   Phonetic Correction + Majority Vote + LLM
   Phonetic correction is applied to adjust any misclassified letters, followed by smoothing with the majority vote strategy. LLM is then used to finalize the sentence and ensure it's coherence.
3) **Pipeline 3:**
   Levenshtein Correction + LLM
   Levenshtein correction is applied to find the closest word matches, followed by LLM to correct sentence structure and ensure grammatical accuracy.
4) **Pipeline 4:**
   LLM
   LLM correction without any preprocessing, to verify if preprocessing does not reduce LLM efficiency.

IV. Dataset Acquisition

In this section, we detail the systematic approach undertaken to create a robust and comprehensive ASL dataset tailored for training real-time recognition models. The dataset forms the cornerstone of our research, enabling the development of an accurate and generalizable recognition system. The process encompassed participant recruitment, data acquisition, meticulous annotation, and thorough preprocessing to ensure high-quality input for ASL recognistion model training.

Several existing ASL datasets employ controlled collection environments to maintain consistency in background, lighting conditions, and image parameters [20]. Although this method is regarded as standard practice, it diverges from real-world contexts in which the DHH community frequently use widely accessible cameras in diverse settings for video communication. Our dataset collection approach is different from the MS-ASL dataset, in which the captured images are sourced from publicly available videos of individuals communicating in ASL [21], but it is similar to collection method of MNIST ASL dataset, except that the images in mnist [2] are in very low quality and small size.

A. Collection procedure

For data collection, we introduce the *ASL Data Collecting* package [22], designed to streamline the acquisition and management of hand gesture datasets. This tool provides a user-friendly command-line interface that enables the capture of hand images with real-time landmark detection, making it highly effective for generating large-scale ASL gesture datasets. The package also includes features for uploading, downloading, and managing files in dataset storage systems, which are crucial for efficiently handling large datasets. By automating these processes, the package significantly reduces the manual effort required for data collection, thereby improving the efficiency of developing ASL recognition models and providing the opportunity for continuous dataset expansion.

Developed in Python, the software leverages OpenCV and MediaPipe for image processing and hand landmark detection tasks. The *ASL Data Collecting* package has been made publicly available on the Python Package Index (PyPI) repository, allowing for easy and efficient expansion of the dataset with additional labeled images [23].

Each data collection procedure consists of 26 segments, corresponding to the collection of image samples for each letter of the ASL alphabet. For each letter, the software attempts to capture 100 images. To ensure the accuracy of the collected images, real-time hand landmark detection is employed, which provides feedback if the hand is not properly detected. If the image collection process for a given letter takes an excessive amount of time, the procedure automatically transitions to the next letter. Upon completion, the collected images are packaged and prepared for upload to the dataset storage. For security and validation purposes, the upload process is decoupled from the collection procedure, allowing for the verification of images prior to their upload.

B. Dataset characteristics

We asked 16 volunteer person to conduct ASL dataset collection procedure (age between 22 and 34 years). Before the experiment, each participant was familiarized with the motivation behind the data collection. The average participant finished a session in approximately 7 minutes. During each session, 100 images were collected for each of the 26 ASL letters, with a few exceptions where full data was not obtained. All sessions were successfully conducted. Example images from dataset are presented in Table I.

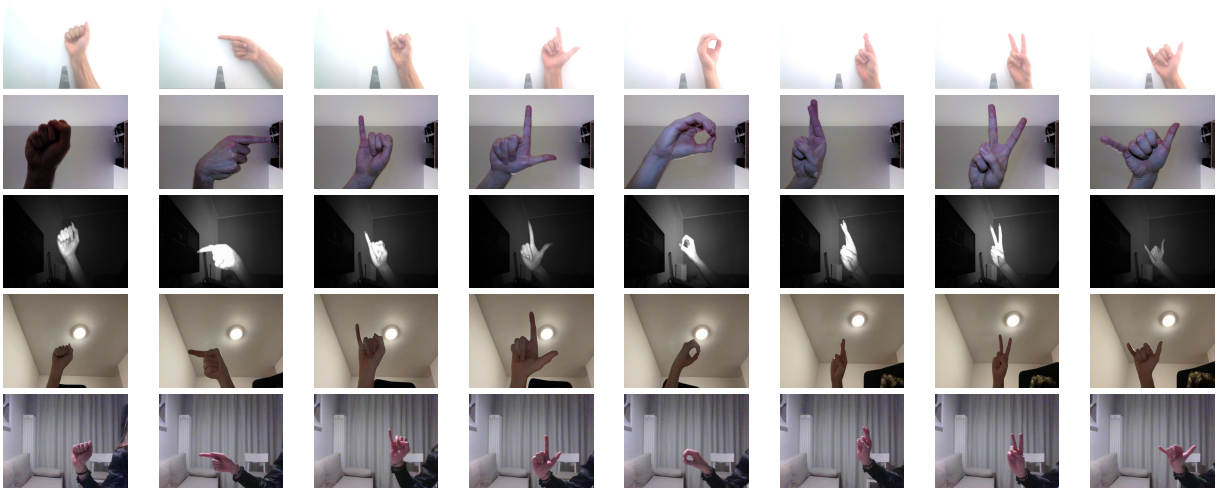
The collected dataset, titled the ASL Hands dataset [24], has been made publicly available on the Kaggle platform. This provides a dedicated space for open access, enabling researchers and developers to utilize the dataset for ASL-related tasks and further research.

C. Dataset Evaluation

The class distribution was analyzed to verify that all ASL letters were sufficiently represented across the datasets (Fig. 6). Blue bars in the distribution chart indicate letters for which data from all participants was successfully collected, while red bars represent instances where data was missing for specific letters. Completeness of the dataset was ensured by capturing 100 images per letter, wherever possible. Gaps in data collection were identified and steps were taken to either recollect missing data or apply data augmentation techniques to supplement underrepresented classes.

TABLE I
EXAMPLE IMAGES FROM COLLECTED ASL HANDS DATASET FOR SELECTED LETTERS AND RANDOM PARTICIPANTS

A	G	I	L	O	R	V	Y



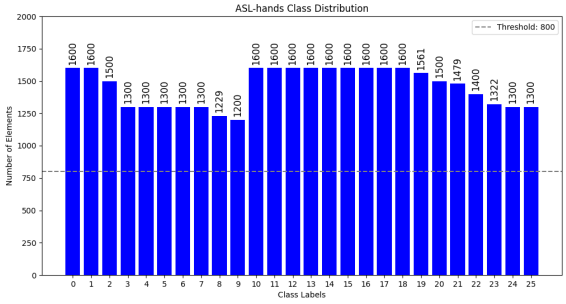


Fig. 6. The class distribution in the ASL Hands dataset. The blue bars represent letters for which the dataset includes complete data from all participants.

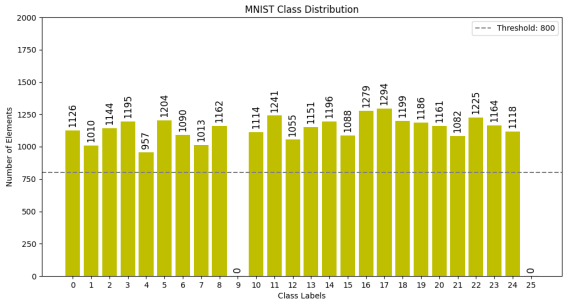


Fig. 7. The class distribution in the Mnist [2] dataset. The yellow bars represent letters for which the dataset includes data.

Each frame was recorded only when hand landmarks are present, which implies that hand are present. Correctness of ASL sign in its corresponding class will be discussed in Section VI by comparing gesture recognision results using our dataset and Sign Language MNIST dataset [2].

V. IMPLEMENTATION

In this section, we demostrate implementation details of custom web application for communication with chatbox service with ASL and discuss sign classification details.

the systematic approach undertaken to create a robust and comprehensive ASL dataset tailored for training real-time recognition models. The dataset forms the cornerstone of our research, enabling the development of an accurate and generalizable recognition system. The process encompassed participant recruitment, data acquisition, meticulous annotation, and thorough preprocessing to ensure high-quality input for ASL recognistion model training.

A. Web Application

The primary objective of the application [25] is to enable users to convert ASL hand gestures into readable text and seamlessly transmit this text to a chatbot service. The user interface is designed with an emphasis on simplicity and accessibility, offering an intuitive and user-friendly experience. The application captures gestures in real-time via a webcam, allowing for smooth interaction. Once the gestures are detected and converted into text, users have the opportunity to edit and refine the output before sending it to a LLM such as ChatGPT for further processing.

The application is developed using Python's FastAPI framework [26], chosen for its high performance, scalability, and ease of integration. For real-time hand gesture detection, MediaPipe is employed, which ensures accurate and efficient gesture recognition, crucial for effective ASL interpretation. To ensure portability and ease of deployment across different environments, the entire system is containerized using Docker, enabling deployment on any machine with minimal configuration, thereby promoting broad accessibility.

Interface of the application is presented in Fig. 8. The typical user flow begins with running the application in a Docker container and passing the necessary configurations

for ChatGPT. Once the application is launched in a browser, the user clicks the *Start Camera* button to initiate the real-time gesture capture. As gestures are translated into text, the user can press the *Correct Text* button to refine the output into human-readable form. Finally, by clicking the *Send Chat* button, the user submits the text to the chatbot service, and the response from the LLM is displayed in a designated window for further interaction. This streamlined process ensures an efficient and user-friendly approach to ASL-based communication with chat services.

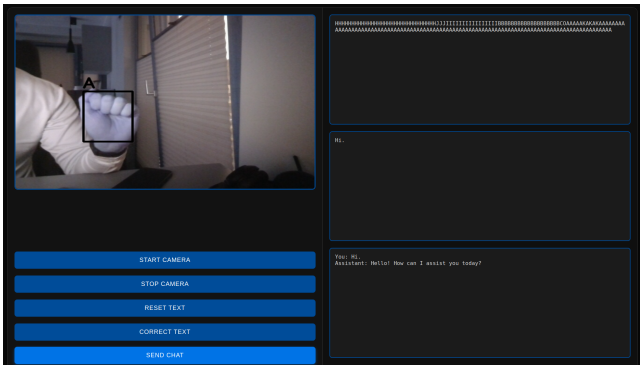


Fig. 8. Screenshot of the web application developed for communication with a chatbot using ASL. The upper window displays the detected letters derived from real-time ASL gesture recognition, the middle window shows the preprocessed text, and the lower window contains the chatbot's response generated after processing the input.

B. Models

a) LeNet: architecture [27], visualization4, implemented using the PyTorch framework, is a classic convolutional neural network designed for grayscale images. For this project, it processes 28x28 grayscale inputs, ideal for simpler image tasks like handwritten digit recognition. The model's structure involves a series of convolutional and pooling layers followed by fully connected layers, making it lightweight and efficient. It's a strong baseline for small-scale image classification problems.

b) Resnet18: architecture [16], also implemented in PyTorch, works with 224x224 RGB images and uses residual connections to solve the vanishing gradient problem in deep networks. It's deeper than LeNet, with 18 layers(counting hidden), allowing it to handle more complex image data. ResNet's skip connections help retain information, making it powerful for both simple and complex tasks.

c) Models training: The trained models for ASL gesture recognition have been made publicly available on a GitHub repository [28]. This repository contains training scripts for a variety of machine learning models, including AdaBoost, LeNet (CNN), ResNet18 (DNN), and Random Forest. All models have been specifically adapted to recognize ASL letters. These models were trained using both the MNIST dataset [2] and the collected ASL Hands dataset. To ensure consistency and facilitate deployment across diverse computing environments, the models have been containerized using a Docker image.

Widely recognized for its superior performance in image classification tasks [29]. Our CNN and DNN models was trained using the ASL Hands Dataset [30] and enhanced through image augmentation techniques to increase robustness. The architecture of the model includes multiple convolutional layers, pooling layers, and fully connected layers, all designed to extract high-level features from the input frames and accurately classify gestures into corresponding letters. Previous works have demonstrated CNN architectures to be effective in both static and dynamic gesture classification, with real-time systems achieving impressive accuracy rates in challenging environments [13], [15]. All images were preprocessed using the MediaPipe Hands module to detect hand bounding boxes, which are then cropped and used as input. Data augmentation techniques, such as rotation and scaling, were also applied to improve model generalization.

VI. Experiment Results

The experiments were conducted to evaluate the effectiveness of different machine learning models and text processing strategies for real-time ASL interpretation. The primary focus was on two parts: (1) gesture classification from images, and (2) transforming noisy letter sequences into coherent, human-readable text. For each experiment, we assessed the accuracy of the entire pipeline from gesture classification to final text output.

The following table summarizes the accuracy results for different combinations of image classification models (Random Forest, ResNet, LeNet) and text processing strategies, with the final correction step always involving ChatGPT.

TABLE II
Models Evaluation

Evaluation	Mnist	ASL-hands
AdaBooster	15.7	36.5
Random Forest	82.0	96.8
LeNet	99.8	93.2
ResNet18	99.0	99.6

TABLE III
Experiment Results for Text Processing Pipelines

Text Processing Pipeline	Accuracy (%)
Pipeline 1	19.1
Pipeline 2	34.3
Pipeline 3	26.1
Pipeline 4	68.7

TABLE IV
Experiment Results for Gesture Recognition and Text Processing Pipelines

Image Classification Model	Text Processing Pipeline	Accuracy (%)
Random Forest	Pipeline 4	50.0
LeNet	Pipeline 4	25.0
ResNet18	Pipeline 4	50.0

TABLE V
AVERAGE TIME MEASUREMENT FOR TEXT PROCESSING PIPELINES

Text Processing Pipeline	Average Processing Time (s)
Pipeline 1	4.678
Pipeline 2	6.769
Pipeline 3	11.073
Pipeline 4	0.769

TABLE VI
TIME MEASUREMENT FOR GESTURE RECOGNITION MODELS

Image Classification Model	Test 1 (s)	Test 2 (s)	Test 3 (s)	Test 4 (s)
Random Forest	0:56.730	0:05.495	0:09.634	0:08.270
LeNet	0:26.464	0:03.127	0:05.269	0:06.333
ResNet18	0:54.109	0:05.701	0:07.747	0:08.510

Across all combinations, LLM consistently demonstrated its superior capability in transforming noisy sequences into coherent and grammatically correct sentences, highlighting its indispensable role in the system.

A. Results and Observations

VII. Results and Observations

a) Gesture Classification Accuracy: The models tested for ASL gesture recognition showed significant differences in performance. **ResNet18** achieved the highest accuracy with **99.6%** on the ASL-hands dataset, as shown in Table II. **LeNet** followed with **93.2%**, and **Random Forest** also performed well with **96.8%**, the reason og behaviour is probably a not a very big dataset, being an ensemble method of decision trees can generalize better in such situations. However, **AdaBooster** had a much lower performance, only achieving **36.5%**, which is consistent with expectations given its simplicity.

b) Text Processing Pipelines: Regarding text processing accuracy, **Pipeline 4** (ChatGPT-only) provided the best results with **68.7%** accuracy (see Table III). **Pipeline 2**, which included phonetic correction and majority vote strategies, achieved **34.3%**, while **Pipeline 3** and **Pipeline 1** had lower accuracies of **26.1%** and **19.1%**, respectively. Preprocessing strategies like removing repetitions or applying majority voting could oversimplify the input data or preprocess new errors. The clear takeaway here is that using ChatGPT alone for text correction yields better accuracy.

c) Gesture and Text Processing Pipelines Combined: When combining gesture recognition models with text processing pipelines (Table IV), the combination of **ResNet18** and **Pipeline 4** reached an accuracy of **50.0%**, which was the same result as **Random Forest** paired with the same pipeline. Meanwhile, **LeNet** paired with **Pipeline 4** had a lower accuracy of **25.0%**. These results suggest that models like **ResNet18** and **Random Forest**, when combined with ChatGPT-based correction, are more reliable for generating readable text from ASL gestures.

d) Time Measurements for Text Processing Pipelines: The processing times of the text pipelines varied considerably. As shown in Table V, **Pipeline 3** took the longest, averaging **11.073 seconds**, likely due to the more complex steps involved. **Pipeline 2** had an average processing time of **6.769 seconds**, followed by **Pipeline 1** at **4.678 seconds**. Unsurprisingly, **Pipeline 4** (ChatGPT-only) was the fastest, with an average of **0.769 seconds**.

e) Time Measurements for Gesture Recognition Models: As indicated in Table VI, the processing speeds for the gesture recognition models also varied. **Random Forest** was the slowest overall, with times ranging from **5.495 to 56.730 seconds**. **LeNet** was faster, taking between **3.127 and 26.464 seconds**, while **ResNet18** had times between **5.701 and 54.109 seconds**. While ResNet18 isn't the quickest model, its high accuracy makes the extra processing time a fair trade-off.

VIII. Conclusion

The results of our experiments shows that video communication with LLM is advanced problem, summarized in Table IV, indicate notable differences in the accuracy of the various machine learning models employed for American Sign Language (ASL) gesture recognition. In summary, while both the Resnet18 Neural Network (DNN) and Random Forest models performed commendably, the LeNet was far away and stands behind. CNN anyway stands out as the most effective approach for ASL gesture recognition, the crucial difference between Random Forest and Resnet18 was its learning time, and fact that CNN required more data augumentation. The dataset, consisting of images from only 16 individuals, each providing 100 images per class across 26 classes, poses limitations on the amount of training data available. Given this restricted dataset, the Random Forest model seems to work very well for real-time ASL to text translation.

In contrast, the CNN's architecture is better suited for extracting hierarchical features from images, allowing it to learn more effectively from the limited data available. The findings suggest that while Random Forest can still provide reasonable accuracy, the CNN's can find more details, which sometimes leads to misclassification an required more data preprocessing. The time efficiency of processing samples seems to be almost same for Random Forest and Resnet18, but slightly slower for simple LeeNet.

If more individuals were added to the dataset, it could significantly enhance the models' performance. A larger dataset would enable the ensemble learning technique to capture more variations in hand shapes, sizes, and styles, leading to improved generalization and accuracy. Similarly, CNNs would benefit from the increased data, allowing for a more comprehensive feature extraction process and potentially reducing overfitting. Furthermore, with a more extensive dataset, researchers could explore more complex models or deeper architectures, leading to further enhancements in gesture recognition capabilities.

The current dataset, with its systematic collection of images and corresponding landmarks, provides a solid foundation for training machine learning models. It is structured to facilitate model training by ensuring that each ASL symbol is adequately represented across various individuals, promoting diverse hand gestures and minimizing bias in training. This

ensures that models trained on this dataset can generalize well to unseen data, an essential characteristic for robust gesture recognition systems.

Future research could investigate ways to preapre more augmentations of the existing dataset or explore transfer learning approaches to further improve the models' performance in gesture recognition tasks with limited samples. The addition of more diverse individuals would not only improve model accuracy but also make the system more robust to variations in hand gestures across different users, paving the way for more effective real-world applications.

REFERENCES

[1] H. Vaezi Joze and O. Koller, "Ms-asl: A large-scale data set and benchmark for understanding american sign language," in *The British Machine Vision Conference (BMVC)*, September 2019.

[2] Tecperson, "Sign language mnist," 2017. [Online]. Available: https://www.kaggle.com/datasets/datamunge/sign-language-mnist

[3] A. A. Abdulhussein and F. A. Raheem, "Hand gesture recognition of static letters american sign language (asl) using deep learning," *Engineering and Technology Journal*, vol. 38, no. 6A, 2024.

[4] M. M. Zaki and S. I. Shaheen, "Sign language recognition using a combination of new vision based features," *Pattern Recognition Letters*, vol. 32, no. 4, pp. 572–577, 2011.

[5] A. Núñez-Marcos, O. P. de Viñaspre, and G. Labaka, "A survey on sign language machine translation," *Expert Systems with Applications*, vol. 213, p. 118993, 2023.

[6] A. Gopi, S. D. P, S. T, J. Stephen, and B. VK, "Multilingual speech to speech mt based chat system," in *2015 International Conference on Computing and Network Communications (CoCoNet)*, 2015, pp. 771–776.

[7] J. T. S. Ru and P. Sebastian, "Real-time american sign language (asl) interpretation," in *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*, 2023, pp. 1–6.

[8] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," 2020.

[9] M. R. Chilukala and V. Vadalia, "Translating sign language to english text in real time using deep learning models," in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, 2022, pp. 1296–1301.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[11] C. Bhat, R. Rajeshirke, S. Chude, V. Mhaiskar, and V. Agarwal, "Two-way communication: An integrated system for american sign language recognition and speech-to-text translation," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–7.

[12] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[13] P. Sahane, "Duplex sign language communicator," *International Journal for Research in Applied Science and Engineering Technology*, 2021, a system utilizing NLP and CNN for sign language translation focusing on Indian Sign Language.

[14] A. Dumbre, S. Jangada, S. Gosavi, and J. Gupta, "Classification of indian sign language characters utilizing convolutional neural networks and transfer learning models with different image processing techniques," in *2022 3rd International Conference on Advances in Computing, Communication, and Control (AIC)*, 2022.

[15] M. N. Saiful, A. A. Isam, H. A. Moon, R. T. Jaman, M. Das, M. R. Alam, and A. Rahman, "Real-time sign language detection using cnn," in *2022 International Conference on Data Analytics and Business Intelligence (ICDABI)*, 2022.

[16] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[17] A. LeNail, "Nn-svg: Publication-ready neural network architecture schematics," *Journal of Open Source Software*, vol. 4, no. 33, p. 747, 2019.

[18] A. S. Dhanjal and W. Singh, "Tools and techniques of assistive technology for hearing impaired people," in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 2019, pp. 205–210.

[19] A. S. Lhoussain, G. Hicham, and Y. Abdellah, "Adaptating the levenshtein distance to contextual spelling correction," *International Journal of Computer Science and Applications*, vol. 12, no. 1, pp. 127–133, 2015.

[20] A. L. Barczak, N. H. Reyes, M. Abastillas, A. Piccio, and T. Susnjak, "A new 2d static hand gesture colour image dataset for asl gestures," 2011.

[21] H. R. V. Joze and O. Koller, "Ms-asl: A large-scale data set and benchmark for understanding american sign language," *arXiv preprint arXiv:1812.01053*, 2018.

[22] Michal Chwesiuk and Piotr Popis, "Asldatacollector: Cli tool for managing and processing hand image datasets for asl recognition." https://github.com/sqoshi/asldatacollector, 2024, accessed: 2024-10-20.

[23] P. Michał Chwesiuk, "Asldatacollector: A python package for collecting asl image data," 2024. [Online]. Available: https://pypi.org/project/asldatacollector/

[24] Michal Chwesiuk and Piotr Popis, "Asl hands," 2024. [Online]. Available: https://www.kaggle.com/datasets/piotrpopis/asl-hands

[25] P. Popis, "hands-to-text: A web application and python package for converting sign language gestures into text." https://github.com/sqoshi/hands-to-text, 2024, accessed: 2024-10-20.

[26] S. Ramírez, "Fastapi framework, high performance, easy to learn, fast to code, ready for production," *https://fastapi.tiangolo.com/*, 2018.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[28] Michal Chwesiuk and Piotr Popis, "htt-models: Package for training, processing, and versioning models for hands-to-text." https://github.com/sqoshi/htt-models, 2024, accessed: 2024-10-20.

[29] N. Bhavana and G. S. Shenoy, "Empowering communication: Harnessing cnn and mediapipe for sign language interpretation," in *2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET)*, 2023.

[30] H. V. Joze and O. Koller, "Ms-asl: A large-scale data set and benchmark for understanding american sign language," in *The British Machine Vision Conference (BMVC)*, September 2019.