LERMO: A Novel Web Game for AI-Enhanced Sign Language Recognition

Adilson Medronha[1*], Luís Lima[1*], Janaína Claudio[2], Lucas Kupssinskü[1], Rodrigo C. Barros[1]

[1]School of Technology, Pontifícia Universidade Católica do Rio Grande do Sul
[2]School of Humanities, Pontifícia Universidade Católica do Rio Grande do Sul
{adilson.medronha, luis.lima97}@edu.pucrs.br
{rodrigo.barros, lucas.kupssinsku, janaina.claudio}@pucrs.br

Abstract

Sign language is a visual and gestural communication system used by deaf and hearing-impaired people. Despite numerous deep learning methods proposed for automatic interpretation, a gap persists in developing applications that effectively utilize these models for assisting sign language studies and inclusion. We introduce LERMO (**https://lermo.app/**), a web game merging machine learning and gamification to enhance sign language fingerspelling. Inspired by Wordle™, LERMO offers an interactive word-guessing game where users can play using a video camera. We create a new dataset of labeled landmark fingerspelling and design our model to ensure optimal speed and efficiency to run on a web browser. We survey approximately 40 users, which find LERMO user-friendly and innovative. From those, 95% believe LERMO could be used to enhance fingerspelling skills.

Introduction

The World Federation of the Deaf has highlighted the global utilization of more than 200 sign languages, catering to over 70 million deaf individuals (World Federation of the Deaf 2023). Sign languages, such as the American Sign Language (ASL), Brazilian Sign Language (LIBRAS), and others, facilitate effective communication for individuals with hearing impairments. These languages convey detailed information using nonverbal elements like facial expressions, hand gestures, and body movements. The comprehension of sign language plays a crucial role in promoting inclusion and mutual understanding between the hearing and deaf communities.

Research in sign language recognition (SLR) has gained significant attention, primarily after the effectiveness of deep learning in computer vision (Jiménez-Salas and Chacón-Rivas 2022). SLR presents unique challenges for automated recognition systems due to their complexity and variability across different sign languages. However, despite the significant advancements in research and technology, the integration of AI-driven technology in learning and dissemination of sign languages still remains relatively underdeveloped.

Sign language is a complete and intricate form of communication containing all the nuances found in spoken languages. Beyond the initial learning stage, mastering this language is not easily achieved. Becoming proficient requires a significant period of exposure and dedicated practice (Kemp 1998). In order to combine practice and learning, educational digital games can be an effective tool since they combine fun and engagement with practical learning and interactive entertainment (Prensky 2001).

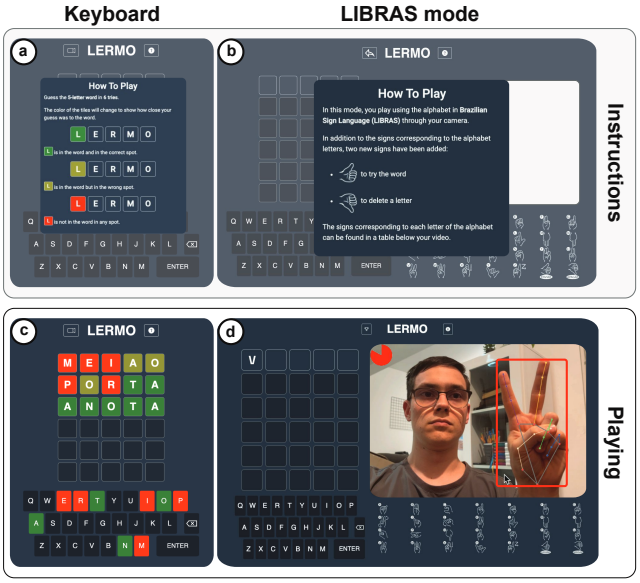

Figure 1: (a, b) Keyboard and sign language mode instructions, (c) an example of letter color status, (d) illustration of landmarks, progress circle, and bounding box status.

With that in mind, we introduce LERMO, an interactive web game combining machine learning, gamification, and dactylology (*fingerspelling*) studies in LIBRAS. Inspired by popular games such as Wordle™ (American English) and TERMO (Brazilian Portuguese), LERMO provides an engaging word-guessing experience via fingerspelling recognition. We evaluate LERMO's usability and effectiveness by applying a user survey in a group of more than 40 users. Results suggest that users find LERMO to be a valuable tool for developing proficiency in LIBRAS fingerspelling, motivating the hearing community to start their learning process towards better communicating with the hearing-impaired.

*These authors contributed equally.

Related Work

Within the domain of SLR, the methodologies often employed mainly fall into two categories: sensor-based recognition and computer vision-based recognition (Tasmere and Ahmed 2020). Although sensor-based approaches such as computerized gloves can provide satisfactory accuracy since the sensors collect data and features directly from the user's hand, the drawback is that users must always use specialized hardware to communicate, making it less attractive and more expensive (Dalal, Kacheria, and Venkataramanan 2022). In contrast, computer vision-based recognition and deep learning techniques have gained considerable attention in the recent years. This approach offers ease of use by not requiring any wearable devices like data gloves or motion trackers. Nevertheless, it often involves acquiring substantial computational resources while being susceptible to changes in background conditions and variations in lighting (Hassan, Assaleh, and Shanableh 2019).

Sign language categorizes gestures into static and dynamic. Static hand gestures do not rely on a specific temporal sequence and can be presented in any order. These gestures, such as fingerspelling, involve individual static hand signs or positions representing specific letters or components, enabling flexible and non-sequential communication. However, some letters in LIBRAS alphabet are represented with hand movements, making them dynamic, including: *H, J, X*, and *Z*, while the others are static gestures.

Static hand gesture recognition benefits from convolutional neural networks (CNNs) due to their ability to analyze image data and extract patterns and features (Jiménez-Salas and Chacón-Rivas 2022). In contrast, dynamic gestures require retaining previous states and typically involve a repeated chain-like structure of modules. These gestures often convey specific meanings or information and function similarly to complete phrases or signs representing some words. One of the prominent deep learning techniques employed in this field is recurrent neural networks (RNNs), including long short-term memory (LSTM) networks, because this structure allows capturing and retaining long-term relationships within sequential data (Areeb and Nadeem 2021).

Das, Ahmed, and Ali (2020) proposed a convolutional neural network consisting of four convolutional layers, four pooling layers, and four fully-connected layers to recognize static signs for the 26 letters of the ASL alphabet. The maximum recognition rate reached was 94.34%. Another study employing CNNs is the work of Kumari and Anand (2022). They proposed a method that utilizes CNNs, specifically VGG16, VGG19, and MobileNet, leveraging an end-to-end fine-tuning approach to identify static hand gestures corresponding to the 24 classes of ASL. The model achieved an average validation accuracy of 94.9%. In another deep convolutional neural network approach, Chanda and Nyeem (2022) applied a U-Net architecture (Ronneberger, Fischer, and Brox 2015) for semantic segmentation to obtain segmented output images. Subsequently, they systematically input these segmented images into a CNN-based classifier, achieving a recognition rate of 97.15%.

There is also related work in the recognition of static gestures in LIBRAS. In their study, Pizzolato, dos Santos Anjo, and Pedroso (2010) aimed to recognize static and dynamic gestures, including LIBRAS alphabet and word recognition. They preprocessed their images using binarization and edge detection to disambiguate the confusion between some highly similar signs in the LIBRAS alphabet (such as *F* and *T*). They employed a two-level architecture, grouping signs with similar hand postures for a preliminary artificial neural network (ANN) classification. To solve the problem with highly-similar classes, they applied a second ANN to recognize specific characteristics of these signs.

Bastos, Angelo, and Loula (2015) employed shape descriptors (HOG and ZIM) to extract information related to the edges and shapes of hands in digital images. Furthermore, they utilized a skin detection algorithm that leveraged components from RGB, HSV, and YIQ color spaces, along with a two-stage neural network classifier, to recognize static gestures. The recognition reached an average of 97%.

Furtado, De Oliveira, and Shirmohammadi (2023) introduced a system enabling real time sign recognition and translation into Brazilian Portuguese text. They applied a preprocessing step to enhance their recognition process to detect the hand within the images. This preprocessing involved removing the background and converting the image to grayscale. They achieved an accuracy of 97% by using a neural network classifier known as Inception-V3.

Regarding educational games designed for acquiring proficiency in sign language, Bouzid et al. (2016) introduces MemoSign. The game does not have a sign recognition system. Instead, it includes components such as a learning version of the Memory Match Game, a 3D virtual avatar, and SignWriting, which is a system designed to graphically represent sign gestures through symbols. When a player turns over a card containing a SignWriting notation, a 3D virtual avatar initiates the interpretation of the notation content through visual-gestural modalities. The objective is for the player to select the card corresponding to the sign presented by the avatar, thereby matching the pair. The signing avatar acts as a guide to help players understand SignWriting notation content by presenting and interpreting transcribed gestures in a natural and comprehensible manner. MemoSign supports two sign languages: American Sign Language and Arabic-Tunisian Sign Language.

Gameiro, Cardoso, and Rybarczyk (2014) introduced the Kinect-Sign, a Kinect sensor-based game for Portuguese fingerspelling recognition. The game is divided into two modes: school and competition mode. In school mode, users learn sign language alphabet through exposed lessons with a repetition-based approach. In competition mode, users apply their acquired skills in interactive settings, such as quiz games or guessing a five-letter word, similar to LERMO. However, both modes utilize a distance-based gesture recognition algorithm using bitmaps captured by a depth camera. The algorithm assesses user gestures by comparing them with a pre-stored dataset, utilizing pixel-wise comparison, hardcoded masks, and five specific conditions. Their approach is designed to accommodate specific hardware dependencies, particularly relying on the Kinect sensor.

Figure 2: LERMO sign recognition workflow.

LERMO

Our work focuses on developing a browser-based word-guessing game called LERMO. We draw inspiration from Wordle™, which belongs to a subgenre of deductive guessing games, such as Bulls & Cows, Mastermind™, and Jotto. The objective is to guess a five-letter word within six attempts by selecting letters one at a time. As seen in Figure 1-a, after each attempt, the player receives feedback: green for letters in correct placement, yellow for incorrect placement, and red if letter is not a part of the word. A keyboard displays letters and their corresponding colors (indicating correctness) to help the player in making informed choices (see Figure 1-c). This feedback system assists the player to deduce the word correctly within the number of tries.

To effectively play using LIBRAS, LERMO captures frames through the user's camera, identifies hands within the frames, and estimates the signs. When users initiate in video mode, it requests permission to access the camera. Once a user grants this permission, the system instantly showcases the live camera feed. We added additional feedback layers to enhance the user experience and allow gameplay using only the natural interface (camera and hand signs). During sign recognition, the system overlays a bounding box over one of the user's hands to indicate which hand it recognizes, and a progress circle indicates the sign recognition progress (see Figure 1-d). Additionally, we introduce two control signs (delete and send) to assist in gameplay, eliminating the need for keyboard use. A table containing signs representing letters of the alphabet is located below the user's video feed, assisting those who may not be familiar with it yet.

Frontend and Backend

LERMO[1] is a client-side application that runs exclusively in user's browser. Frontend is developed in Angular. We refer to "backend" as the independent and offline model training process. We create our neural-based models from scratch using Python and PyTorch. For the remaining four models, we use Scikit-learn, a machine learning library. The optimal model is saved as an Open Neural Network Exchange (ONNX) file and integrated into frontend using onnxruntime-web JavaScript library. We invoke this API to perform inference for each landmark captured by MediaPipe (Lugaresi et al. 2019). Additionally, our database is stored in cloud using Firebase, comprising a total of 5124 words.

[1]https://github.com/adilsonmedronha/LERMO

In video mode, LERMO employs the MediaPipe hand landmark detector. It captures the user's frame, identifies hands within it, and estimates the positions of 21 landmarks. The palm detection model is reactivated when the hand landmarks model encounters difficulties in tracking hands within the frames. Among the 21 data points, we only use x and y coordinates, disregarding z due to their less accurate estimation, reducing dimensions from 63 to 42.

As long as the user's hand remains within video frame and hand landmarks are successfully detected, we apply feature scaling to standardize all data points. This preprocessing centers these points around the wrist coordinates $(0, 0)$, ensuring that our model remains invariant to changes in video resolution and hand translation. Finally, we feed these landmarks to a multilayer perceptron (MLP) model. Since individual frames during video capture may be blurred, resulting in poor landmark recognition, we accumulate predictions over 100 frames and use the mode of the predictions as the classification outcome. By employing the most frequent class, we have a stable classification process, even when some frames are blurred or when MediaPipe fails to detect landmarks. We use 100 predictions because the average frames per second (FPS) on 3 different machines was 20 (5 seconds per letter: time duration of progress circle), providing a comfortable interval for model confidence and user awaiting. Figure 2 presents an overview of these steps.

Datasets

In SLR, datasets for static signs in LIBRAS are relatively abundant (Furtado, De Oliveira, and Shirmohammadi 2023; de Souza and Pizzolato 2013; Bastos, Angelo, and Loula 2015) while datasets for dynamic signs remain scarce (CEFET/RJ-LIBRAS (Araujo et al. 2016; Gameiro et al. 2020), MINDS-LIBRAS (Rezende, Almeida, and Guimarães 2021), LIBRAS-UFOP (Cerna et al. 2021)). The lack of available sign language datasets represents an ongoing challenge (Bragg et al. 2019). Indeed, all datasets we encountered for static signs primarily had concerns related to hand background variability, low resolution, size variation, among others, mainly catering to image-based approaches.

With our objective centered on introducing LERMO, which employs a fingerspelling approach based on keypoints, we created a tool for image collection and landmark estimation. Leveraging the fact that MediaPipe has been extensively trained on a dataset with over $100,000$ samples, we do not have issues related to background, skin color diversity, and image generation across various scenarios, such as

Models	Metrics	Actor 1		Actor 2		Actor 3		Actor 4		Merged	
		Normal	Aug.	Normal	Aug.	Normal	Aug.	Normal	Aug.	Normal	Aug.
MLP	Accuracy	0.8420	0.8360	0.8072	0.7872	0.8861	0.8935	0.8781	0.8628	0.8988	**0.9017**
	F1 Score	0.8193	0.8157	0.7736	0.7476	0.8640	0.8799	0.8633	0.8395	0.8787	**0.8846**
	Precision	0.8528	0.8452	0.8277	0.7829	0.8819	**0.9144**	0.9091	0.8676	0.8965	0.9125
SVM	Accuracy	0.7653	0.7687	0.7227	0.7293	0.8181	**0.8712**	0.8490	0.8594	0.8643	0.8566
	F1 Score	0.7362	0.7371	0.6569	0.6739	0.7827	**0.8439**	0.8275	0.8432	0.8423	0.8343
	Precision	0.7666	0.7600	0.7057	0.7331	0.8556	0.8598	**0.8868**	**0.8868**	0.8648	0.8511
KNN	Accuracy	0.7764	0.7919	0.7290	0.7504	0.8503	0.8579	0.8325	0.8470	0.8698	**0.8701**
	F1 Score	0.7424	0.7502	0.6682	0.6980	0.8284	0.8340	0.8001	0.8246	0.8465	**0.8476**
	Precision	0.7539	0.7578	0.6945	0.7236	0.8396	**0.8889**	0.8506	0.8496	0.8649	0.8695
Random Forest	Accuracy	0.6654	0.7203	0.6141	0.6319	0.7931	0.8217	0.8011	0.8101	**0.8650**	0.8445
	F1 Score	0.6276	0.6745	0.5572	0.5786	0.7631	0.8059	0.7845	0.7946	**0.8411**	0.8219
	Precision	0.6893	0.7187	0.5805	0.6082	0.7982	0.8515	0.8299	0.8482	**0.8706**	0.8468
Naive Bayes	Accuracy	0.5089	0.5626	0.5353	0.5387	0.6925	0.6578	0.6771	0.6941	**0.7143**	0.6694
	F1 Score	0.4862	0.5053	0.4801	0.4825	0.6580	0.5866	0.6801	0.6494	**0.6819**	0.6061
	Precision	0.6022	0.5880	0.5172	0.5209	0.7150	0.6707	**0.7688**	0.7348	0.7299	0.7177

Table 1: Performance of the five models tested on Actor 5 data. The models were trained on the respective augmented and normal data subsets for each actor, and a merged version was also tested. Bold numbers highlight the maximum in each row.

real world, indoor, outdoor images, and similar challenges.

During the data collection process, we request five actors to perform all classes within 20 seconds each. This approach allows us to capture a range of finger positions, overlaps, and different phalanx angles, ensuring landmark diversity, as seen in Figure 3. We also flip the images to ensure that, artificially, each actor had data from the opposite hand. Moreover, we apply three augmentation techniques, including seven random angles for hand rotation, forward and backward inclination, and image resizing.

We create two data versions for each actor: original and an augmented dataset. We employ random undersampling to guarantee exactly 500 samples per class. Finally, we compile a merged dataset comprising the original and augmented data spanning from **Actor 1** to **Actor 4**, resulting in 500 samples per class, 125 samples per class for each actor.

Performance Optimization

We evaluate the following classifiers in our datasets: MLP, Support Vector Machines (SVM), K-Nearest-Neighbors (KNN), Random Forest (RF), and Naïve Bayes (NB). These models must be efficient and effective because it needs to classify signs correctly while being lightweight so it can be executed in a web browser. KNN is a straightforward non-parametric machine learning algorithm commonly employed for classification and regression tasks. However, a drawback of KNN is that it stores all training data, which can adversely affect the user inference speed since it requires iterating through the entire training set. To the best of our knowledge, there is currently no vectorized API or any package approach available for this in the frontend of the system. To address this issue, we reduced the dataset precision from 16 bits to 8 bits. While this precision adjustment reduced memory usage by half (from 2.5 MB to 1.3 MB), it did come at the cost of accuracy. For the MLP model, we took optimization steps during training. This involved implementing mixed precision techniques and quantization to

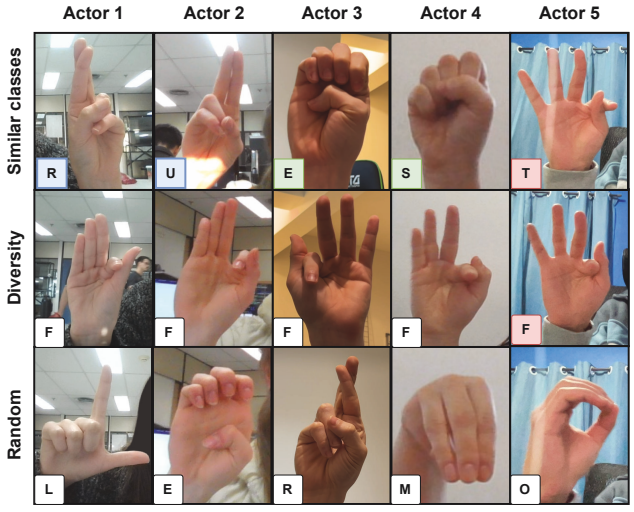

Figure 3: LERMO dataset. In the "Similar Classes" category, the pairwise color coding highlights signals with high similarity, posing a challenge for models to distinguish them. "Diversity" category showcases significant variations within the same class, ensuring better landmark generalization. The "Random" category consists of randomly-sampled images.

enhance inference speed. The final MLP model is remarkably compact, with a file size of only 100KB and $27,326$ parameters. In the following section, we detail the design choices of the final selected model running in frontend.

MLP

We selected the MLP model to be in frontend due to its superior performance and efficiency. The model architecture is illustrated in Figure 5. Concerning the training procedure, we set a batch size of 512 samples, conducted for 300 epochs with early stopping, and employed the Adam optimizer (Kingma and Ba 2017) with an initial learning rate of 10^{-4}. The output layer of the model is designed to have 25 classes plus two for control signals (thumbs up and down: submit a word and delete a letter, respectively).

Quantitative Results

In this section, we conduct a comprehensive experimental evaluation to assess the performance of the models in SLR (letter classification) using hand landmarks. Our objective is to understand better which letters pose the most significant challenges to the models. This will help us identify areas where improvements can be made. Our analysis comprises eight experiments, as detailed in Table 1, in which we validate the performance of MLP, SVM, KNN, RF, and NB classifiers using the test dataset (Actor 5). It is important to mention that we did not use any Actor 5 data during training.

As shown in Table 1, the resulting data collected by Actor 2 produces less favorable overall metrics than other Actors. This could be attributed to a unique challenge posed by Actor 2, stemming from limited finger joint mobility, which led to difficulties in conveying gestures. However, this diversity can benefit the classifiers, as it helps them generalize and account for potential biases that may arise when working alongside individuals with limited hand mobility.

The confusion matrix of the final model (MLP trained over the merged dataset), as shown in Figure 4, presents clear evidence that the similarities among the highlighted classes presented in Figure 3 (*R, U, E, S, T, F*), indeed posed challenges in differentiation, as observed during the classification test. This behavior was also reported in feedback provided by four of the survey respondents in **Q7**, where they indicated specific letters were wrongly classified during LERMO video mode. Furthermore, Figure 4 includes four latent spaces of these misclassified classes for visual inspection. Note that those are difficult to linearly separate.

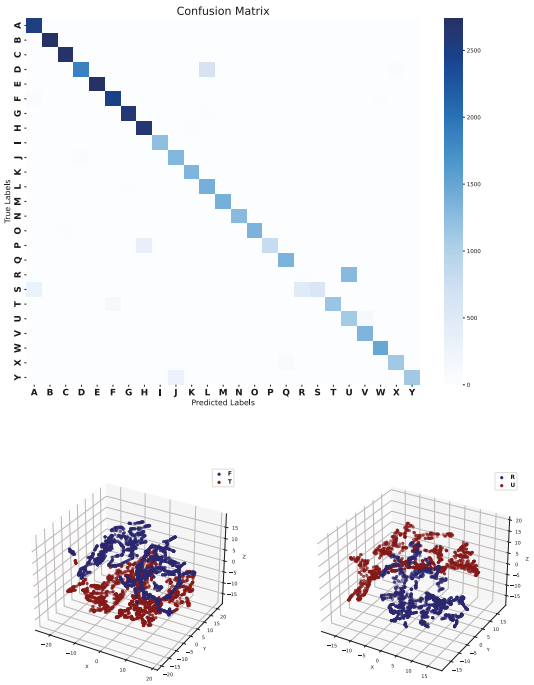

Figure 4: MLP confusion matrix.

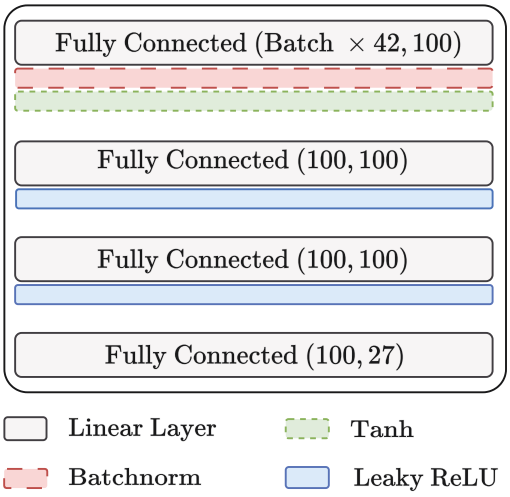

Figure 5: MLP model backbone.

User Experience Surveys

We conducted a comprehensive questionnaire survey to gain a deeper understanding of user experience. Specifically, we aimed to assess LERMO's effectiveness in facilitating fingerspelling practice in LIBRAS and its ability to deliver an engaging blend of learning and entertainment.

We distributed the questionnaire online to gather insights from diverse respondents. The questionnaire includes both single-choice questions and a 5-point Likert scale. We aimed to capture different perspectives and experiences by including individuals with and without previous experience with LIBRAS. With 41 responses, we collected data from participants aged under 18 to over 60. Among the participants, two are either deaf or hearing-impaired.

To align with the central question of our research, we inquired whether participants have any prior experience with LIBRAS or if they are currently learning it. In Figure 6, **Q1** illustrates that most participants are not learning LIBRAS. **Q2** shows that 92.3% have either basic or no proficient level

in LIBRAS. That emphasizes the necessity of practical tools and resources, such as LERMO, to facilitate the learning and utilization of LIBRAS for adwider audience.

The effectiveness of the sign recognizer is under assessment in **Q7**. While most signs were correctly recognized, some signs faced misclassification, as reported by respondents, including signs for *F, T, R, U* and *E*. This issue is related to the noise introduced by variations in finger positions, which can affect the accuracy of landmark detection by MediaPipe, leading to the misclassification of highly-similar signs. As observed in Figure 3, the signs for *E* and *S* are very similar, differing only in the position of the thumb. This variation in thumb position may have caused landmark overlap in MediaPipe's estimation, depending on how the user performed the sign. MediaPipe may also encounter challenges when estimating landmarks in depth. Despite the thumb's oscillation being the only distinction between the signs for letters *F* and *T* (refer to Figure 3 to observe the similarity), the index finger points towards the screen, potentially disturbing the landmark detection due to the depth estimation leading to wrong classification.

To assess system's performance on different devices, **Q8** aims to understand the speed of sign recognition, allowing us to measure how effectively the system operates across various hardware configurations. As shown in Figure 6, only 10.8% of the users reported slow sign recognition speed. Note that LERMO does not currently offer a mobile version. Therefore, these results rely on respondents who used personal computers, typically equipped with better hardware capabilities. This question is particularly significant because our model operates in real time, and making it efficient across different devices is crucial for a good user experience.

Limitations

Mobile Device UX/UI: LERMO is currently unsuitable for mobile-device use for two primary reasons. In order to offer precise feedback to users, as depicted in Figure 1, it requires access to their guessing status and the history of the letters they have tried. They may also need to reference the signing table as required. Most importantly, users must capture their self-camera feed within a small (typically 6.0-inch) screen while sharing space with all components cited above.

Mobile Hardware Limitations: recognizing the limitations of mobile hardware is crucial. These devices often face constraints related to processing power, memory, and graphics capabilities, which can present additional challenges when running an application like LERMO. This is important since it runs a two-stage neural network model for hand and landmark estimation and another MLP letter classifier in the user's web browser in real time.

Word Level Recognition: to improve LERMO's capabilities to recognize word-level sign language signals, we need to develop an effective way to communicate the user's intent to start and finish word sign input, ensuring that our model operates within the accurate time frame. To this end, we need to evaluate communication interface approaches suitable for both smartphone and desktop environments.

Conclusions and Future Work

This paper introduces a novel web-based game that leverages AI technology to facilitate user practice in LIBRAS sign language fingerspelling. We have contributed to the field by introducing a novel sign language application and a new dataset based on hand landmarks. To evaluate the performance, we conducted comprehensive experiments with five different machine learning models, achieving better results with an MLP in the proposed dataset. As seen in Table 1, the MLP-based model exhibited better performance compared to other machine learning algorithms, excelling in precision, F1 score, and accuracy. Additionally, as demonstrated by our survey results, 94.9% (strongly agreeing and agreeing) of participants recognized LERMO as a valuable resource for individuals without disabilities interested in acquiring sign language skills.

Furthermore, in our efforts to disseminate LERMO and improve accessibility and user convenience, we intend to develop a mobile version given the widespread use of smartphones. Addressing concerns related to colorblindness, we plan to incorporate a texture effect for each red, green and yellow color status. In our upcoming studies, we aim to introduce three new features and allow users to interact with American Sign Language in an English version:

Word Sign Level Recognition: once the user correctly guesses the word through fingerspelling, they can reproduce the corresponding word sign. If the sign is correct, they will see a new word for fingerspelling followed by the word-level attempt. This process can be repeated up to three more times, offering a rewarding experience for users to improve their vocabulary and proficiency in LIBRAS or ASL. The new specialized model can be seen as a teacher, providing real-time feedback to guide users in learning the correct sign through observation and trial.

Leaderboard: this feature will introduce an enriched gamification layer, adding an competitive element to overall experience. Users can monitor and compare their performance with community if they choose to log in. The number of attempts, time taken, and consecutive guessed successes will contribute to their final score. LERMO will have 12 seasons per year and the leaderboard will reset every month.

AI Versus Mode: for a harder game mode, users have the opportunity to challenge an expert "*artificial intelligence algorithm*" rooted in information theory. The screen will be split, with the user's board on the left and the AI's blurred progress on the right. In video mode, we simulate the display of a robotic hand forming individual letters for letter-level interaction. At the sign-word level, we employ a complete virtual agent to execute the entire temporal sign.

LERMO is a promising tool in assisting people in practicing sign language, being effective and efficient in letter-level SLR, and also being reported as a positive experience by its users. Moreover, we plan to keep LERMO accessible online indefinitely, allowing individuals to continue learning with a fun experience that will benefit everybody involved.

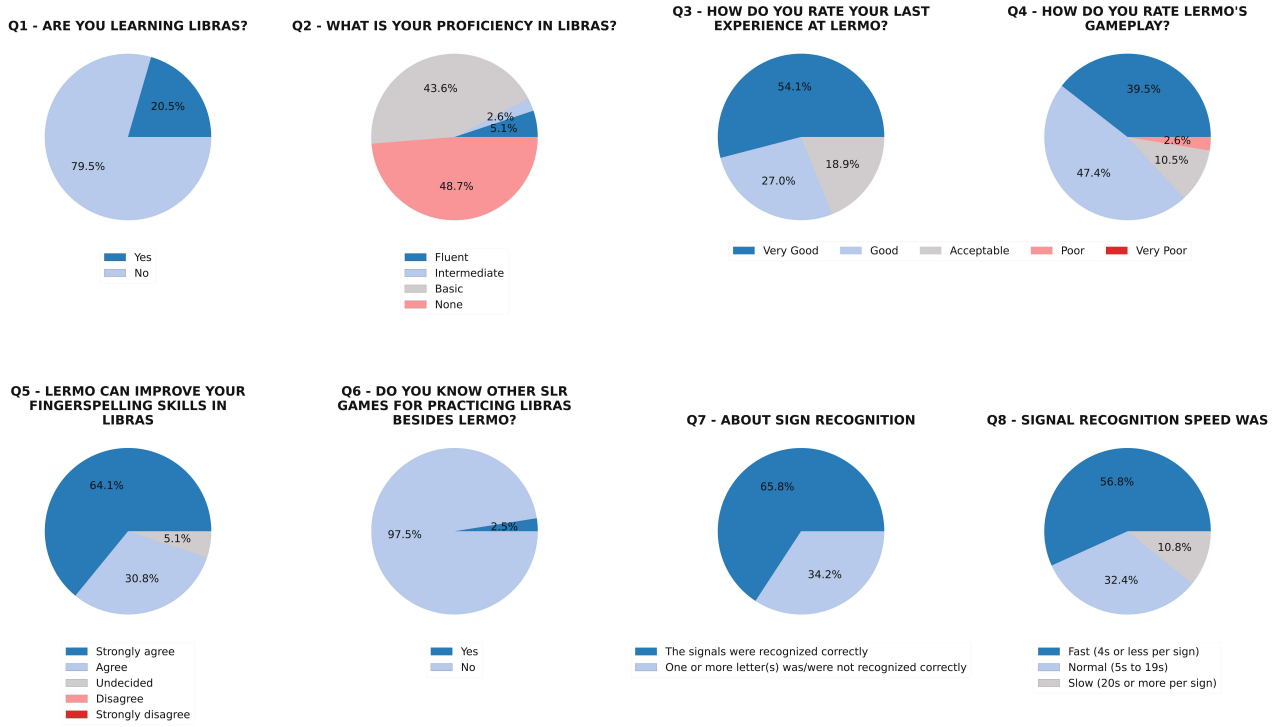

Figure 6: Questionnaire surveys.

Acknowledgements

We would like to acknowledge Google for funding this research with the Google Award for Inclusion Research (AIR). Special thanks to Nathan Gavenski for his valuable input and review and also to Professors Filipe Zabala and Avelino Zorzo for their insightful suggestions.

References

Araujo, G.; Corbo, A. R.; Lacerda, A.; Pecoraro, L.; and Monteiro, C. 2016. Um sistema de baixo custo para reconhecimento de gestos em LIBRAS utilizando visão computacional. In *Proceedings of the XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, 349–352.

Areeb, Q. M.; and Nadeem, M. 2021. Deep Learning Based Hand Gesture Recognition for Emergency Situation: A Study on Indian Sign Language. In *Proceedings of the Second International Conference on Data Analytics for Business and Industry (ICDABI)*, 33–36. New York City, NY: IEEE.

Bastos, I. L.; Angelo, M. F.; and Loula, A. C. 2015. Recognition of Static Gestures Applied to Brazilian Sign Language (LIBRAS). In *Proceedings of the Twenty-Eighth SIBGRAPI Conference on Graphics, Patterns and Images*, 305–312. Los Alamitos, California: IEEE Computer Society.

Bouzid, Y.; Khenissi, M. A.; Essalmi, F.; and Jemni, M. 2016. Using Educational Games for Sign Language Learning - A SignWriting Learning Game: Case Study. *Journal of Educational Technology & Society*, 19(1): 129–141.

Bragg, D.; Koller, O.; Bellard, M.; Berke, L.; Boudreault, P.; Braffort, A.; Caselli, N.; Huenerfauth, M.; Kacorri, H.; Verhoef, T.; Vogler, C.; and Ringel Morris, M. 2019. Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective. In *Proceedings of the Twenty-First International ACM Conference on Computers and Accessibility (SIGACCESS)*, 16–31. New York City, New York: Association for Computing Machinery.

Cerna, L. R.; Cardenas, E. E.; Miranda, D. G.; Menotti, D.; and Camara-Chavez, G. 2021. A multimodal LIBRAS-UFOP Brazilian Sign Language dataset of minimal pairs using a Microsoft Kinect sensor. *Expert Systems with Applications*, 167: 114179.

Chanda, B.; and Nyeem, H. 2022. Automatic Hand Gesture Recognition with Semantic Segmentation and Deep Learning. In *Proceedings of the First International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*, 1–6. New York City, New York: IEEE.

Dalal, S.; Kacheria, R.; and Venkataramanan, V. 2022. A Comparative Study on Sign Language Recognition Methods. In *Proceedings of the Second International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, 1–7. New York City, New York: IEEE.

Das, P.; Ahmed, T.; and Ali, M. F. 2020. Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network. In *Proceedings of the Eighth IEEE Region 10 Symposium (TENSYMP)*, 1762–1765. New York City, New York: IEEE.

de Souza, C. R.; and Pizzolato, E. B. 2013. Sign Language Recognition with Support Vector Machines and Hidden Conditional Random Fields: Going from Fingerspelling to Natural Articulated Words. In Perner, P., ed., *Proceedings of the Ninth Machine Learning and Data Mining in Pattern Recognition*, 84–98. Berlin, Heidelberg: Springer Berlin Heidelberg.

Furtado, S. L.; De Oliveira, J. C.; and Shirmohammadi, S. 2023. Interactive and Markerless Visual Recognition of Brazilian Sign Language Alphabet. In *Proceedings of the Fortieth IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 01–06. New York City, New York: IEEE.

Gameiro, J.; Cardoso, T.; and Rybarczyk, Y. 2014. Kinect-Sign: Teaching Sign Language to "Listeners" through a Game. In Rybarczyk, Y.; Cardoso, T.; Rosas, J.; and Camarinha-Matos, L. M., eds., *Innovative and Creative Developments in Multimodal Interaction Systems*, 141–159. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-55143-7.

Gameiro, P. V.; Passos, W. L.; Araujo, G. M.; de Lima, A. A.; Gois, J. N.; and Corbo, A. R. 2020. A Brazilian Sign Language Video Database for Automatic Recognition. In *Proceedings of the Seventeenth Latin American Robotics Symposium (LARS), Brazilian Symposium on Robotics (SBR) and Workshop on Robotics in Education (WRE)*, 1–6. New York City, New York: IEEE.

Hassan, M.; Assaleh, K.; and Shanableh, T. 2019. Multiple Proposals for Continuous Arabic Sign Language Recognition. *Sensing and Imaging*, 20(1): 4.

Jiménez-Salas, J.; and Chacón-Rivas, M. 2022. A Systematic Mapping of Computer Vision-Based Sign Language Recognition. In *Proceedings of the Fifth International Conference on Inclusive Technologies and Education (CONTIE)*, 1–11. New York City, New York: IEEE.

Kemp, M. 1998. Why is Learning American Sign Language a Challenge? *American Annals of the Deaf*, 143(3): 255–259.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kumari, D.; and Anand, R. S. 2022. Static Alphabet American Sign Language Recognition using Convolutional Neural Networks. In *Proceedings of the Second International Conference on Advances in Computing, Communication and Materials (ICACCM)*, 1–6. New York City, New York: IEEE.

Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uboweja, E.; Hays, M.; Zhang, F.; Chang, C.-L.; Yong, M.; Lee, J.; Chang, W.-T.; Hua, W.; Georg, M.; and Grundmann, M. 2019. MediaPipe: A Framework for Perceiving and Processing Reality. In *Proceedings of the Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR)*.

Pizzolato, E. B.; dos Santos Anjo, M.; and Pedroso, G. C. 2010. Automatic Recognition of Finger Spelling for LIBRAS Based on a Two-Layer Architecture. In *Proceedings of the Twenty-Fifth ACM Symposium on Applied Computing (SAC)*, 969–973. New York City, New York: Association for Computing Machinery.

Prensky, M., ed. 2001. *Digital Game-Based Learning*. New York City, New York: McGraw-Hill.

Rezende, T. M.; Almeida, S. G. M.; and Guimarães, F. G. 2021. Development and Validation of a Brazilian Sign Language Database for Human Gesture Recognition. *Neural Comput. Appl.*, 33(16): 10449–10467.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the Eighteenth Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 234–241. Cham: Springer International Publishing.

Tasmere, D.; and Ahmed, B. 2020. Hand Gesture Recognition for Bangla Sign Language Using Deep Convolution Neural Network. In *Proceedings of the Second International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 1–5. Los Alamitos, California: IEEE Computer Society.

World Federation of the Deaf. 2023. Our Work. http://wfdeaf.org/our-work/. Accessed: 2023-08-22.