CHILI
computer-human interaction
in learning and instruction

LEARN
Center for Learning Sciences

Finger Spelling Recognition for American Sign Language

Author:
Yahya Skalli

Supervisor(s):
Daniel Carnieto Tozadore

Professor:
Pierre Dillenbourg

January 10, 2025

Contents

1 Introduction

Finger spelling is a method of spelling words using hand movements to represent the letters of the alphabet. In the context of data science, Finger Spelling Recognition (*FSR*) refers to the use of machine learning and computer vision techniques to automatically identify these hand gestures. FSR is particularly important for improving communication between the Deaf community and non-signers, as it bridges the gap for those who do not know how to finger spell.

FSR is further motivated by the critical role finger spelling plays in language acquisition and literacy development, particularly for Deaf children. According to Baker et al. [1], finger spelling serves as a bridge between print and sign language, enhancing vocabulary knowledge and phonological awareness, which are essential for reading development. FSR systems not only improve accessibility but can also support educational tools that teach and reinforce these skills. Additionally, by enabling seamless interaction between signers and non-signers, FSR systems can reduce communication barriers and promote social integration. The development of FSR systems also contributes to fostering a deeper understanding of its cultural and linguistic significance.

However, FSR is a challenging domain due to the complexity of hand movements, variations in hand positions, and motion. This project focuses on building a system for recognizing finger-spelled American Sign Language (ASL) gestures, addressing some of these challenges as stated here [2], which include the diverse configurations of hand shapes, high similarity between gestures, environmental factors such as lighting and background changes, variability in hand pose and orientation, and issues of occlusion. Additionally, [3] highlights the variability in signing styles across individuals and contexts, which further complicates recognition.

2 Related Works

The field of sign language recognition has seen significant advancements in recent years, particularly in the integration of machine learning and computer vision techniques. Joshi et al. [4] proposed a real-time sign language recognition system that combines the YOLOv5 and Random Forest Classifiers to achieve high accuracy and efficiency. Their study highlighted the Random Forest Classifier's exceptional precision (0.98) and recall (0.97), making it effective for interpreting ASL gestures. Additionally, their system incorporated Generative AI (GenAI) for sentence formation and text-to-speech functionality, providing a comprehensive solution for bidirectional communication.

Pugeault and Bowden [2] developed an earlier framework for real-time ASL finger spelling recognition using handcrafted features and machine learning classifiers. Although their work laid the groundwork for later developments, its reliance on traditional feature extraction methods limits its scalability and adaptability compared to modern deep learning approaches. Additionally, their work covered only 24 letters of the ASL alphabet, excluding J and Z, as these involve hand shape movements.

Mazalov and Protasov [3] extended this field by exploring transformer-based architectures for sign language translation. Their approach demonstrated the potential of capturing both spatial and temporal dependencies, a feature particularly useful for continuous sign language recognition.

Alyabrodi et al. [5] focused on optimizing machine learning models for sign language recognition through advanced feature engineering and hyperparameter tuning. Their findings emphasized the importance of balancing model complexity with training dataset variability to improve recognition performance.

These studies collectively underscore the progress and challenges in sign language recognition. Building upon this foundation, this project leverages Random Forest, Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) classifiers to address the nuances of ASL finger spelling recognition. By combining insights from prior research and novel methodologies, this work aims to enhance **recognition accuracy** and practical **applicability**.

3 Dataset

The dataset used for this project was sourced from Kaggle[1]. It consists of 87,000 images of 200x200 pixels, distributed across 29 classes: 26 for the alphabet letters, along with ”nothing,” ”space,” and ”delete.” For simplicity and efficiency, the dataset was reduced to cover only the 26 letters of the American alphabet, resulting in a total of 81,000 images.

The dataset includes variations in hand position, lighting, and background, which makes it ideal for training models capable of handling real-world variability.

4 Methodology and Algorithm

The data analysis is based on three main steps:

4.1 Data Collection

The original dataset contains 29 classes. We removed the extra three classes (nothing, space, delete) and retained only the 26 alphabet letters. This resulted in a dataset with 81,000 images, each sized 200x200 pixels, with around 3,000 images per class.

4.2 Data Extraction

We used MediaPipe to extract hand landmarks with a confidence level of 0.8. For each detected hand, 21 landmarks were identified, and for each landmark, we extracted the X and Y coordinates, resulting in a total of 42 features per image (21 landmarks x 2 coordinates). After this step, we were left with 53,572 images that had detectable hand landmarks. The data was then split into training and testing sets, with 80% used for

[1] `https://www.kaggle.com/datasets/grassknoted/asl-alphabet`

training and 20% for testing. See Figure 5 to view the class proportions after the extraction process.

We observed that classes “M” and “N” had lower proportions in the dataset. This imbalance could be due to the hand positioning of these signs, which often involves overlapping fingers that are harder to detect accurately (see Figures 6a and 6b). This limited visibility may cause the landmark extraction model to occasionally misidentify or fail to detect these specific gestures.

4.3 Data Training

ASL recognition is complex because it involves accurately detecting and interpreting highly nuanced hand shapes, positions, and movements, which vary between individuals and even across slight shifts in gesture. Based on related work and previous research, we chose to train models that are most commonly used and proven effective for this type of problem. These models have consistently demonstrated their ability to handle the complexities of ASL recognition tasks, including variations in hand shapes, positions, and gestures. We trained four models on the dataset to tackle these challenges:

1. **Random Forest Classifier (from `sklearn.ensemble`):** The Random Forest Classifier (RFC) was chosen for its strength in handling high-dimensional and complex datasets, making it an effective option for ASL finger spelling recognition. RFC’s ability to capture non-linear relationships and its robustness against overfitting are particularly advantageous for this task. In the work of Joshi et al. [4], the RFC achieved an accuracy of 98.98%, showcasing its effectiveness for ASL recognition.

2. **Multi-Layer Perceptron Classifier (from `sklearn.neural_network`), using the Adam solver with a learning rate of 10^{-5}:** The Multi-Layer Perceptron (MLP) was selected for its flexibility and capacity to automatically learn feature representations. This adaptability makes it well-suited for capturing complex patterns in ASL gestures through its customizable layers. MLP has demonstrated strong performance, achieving an accuracy of 96.15% in the work of Pigou et al. [6].

3. **Support Vector Machine (SVM) with RBF kernel (from `sklearn.svm`):** The Support Vector Machine (SVM) was chosen for its effectiveness in high-dimensional spaces and its ability to form precise decision boundaries. The RBF kernel allows SVM to capture non-linear relationships essential for distinguishing ASL gestures. In the work of Garcia and Valles [5], the SVM achieved an accuracy of 81.89%.

4. **K-Nearest Neighbors (KNN) with $k = 5$ (from `sklearn.neighbors`):** The K-Nearest Neighbors (KNN) algorithm was selected for its simplicity and speed. KNN’s approach of comparing gestures to stored examples makes it naturally adaptable to variations in hand positions, without requiring a training phase. Its application in ASL recognition is also discussed by Garcia and Valles [5], where it achieved an accuracy of 83.40%.

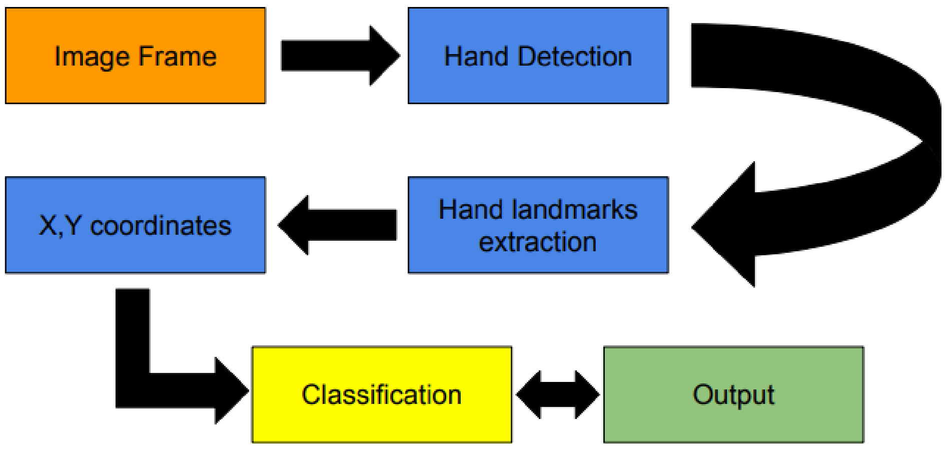


Figure 1: A flowchart depicting the algorithm processing.

5 Results

The performance of four different models — SVM, KNN, MLP, and RFC — was evaluated based on various metrics, including accuracy, training and inference time, memory usage, and cross-validation accuracy, as shown in Table 1. The top-N accuracy for RFC, MLP, and SVM was compared in Table 2.

Model	**Accuracy (%)**	**Training Time (s) / Memory Usage (MB)**	**Inference Time (s) / Memory Usage (MB)**	**Mean CV Acc (%)**
SVM	97.41	15.35 / 2.83	0.32 / 0.74	96
KNN	97.87	0.01 / Negligible	0.54 / 1.01	84
MLP	97.57	50.10 / 0.07	0.01 / Negligible	97
RFC	99.19	27.01 / 1.79	0.21 / Negligible	86

Table 1: Performance Comparison of Different Models

Model	**Top-1 Accuracy (%)**	**Top-3 Accuracy (%)**	**Top-5 Accuracy (%)**	**Top-9 Accuracy (%)**
RFC	99.1	99.89	99.93	99.96
MLP	97.57	99.72	99.88	99.92
SVM	97.41	97.90	99.70	99.75

Table 2: Top-N Accuracy Comparison

In the following section, we will provide a more detailed explanation of the results, supported by various graphs to visualize the performance differences between the models.

5.1 RFC Model Results

From Table 1, RFC achieves the highest accuracy at 99.19%, with a balanced training time of 27.01 seconds and reasonable memory usage. While the model takes longer to train compared to KNN and SVM, it delivers a more accurate and robust performance, especially when generalizing to unseen data.

The top-N accuracy (Table 2) shows RFC's strong predictive ability, achieving: - **Top-1 Accuracy**: 99.1% - **Top-9 Accuracy**: 99.96%

These high Top-N accuracies reinforce RFC's capability to predict correctly within the top-ranked predictions, making it well-suited for real-time applications.

The learning curve and confusion matrix provide further insights into the performance of the Random Forest Classifier (RFC) model.

The learning curve (Figure 11a) shows the relationship between the training set size and model performance, specifically accuracy. The **red line** represents the training accuracy, which is consistently at 100% across all training sizes, indicating that RFC perfectly fits the training data. However, the **green line** represents the cross-validation accuracy, which improves steadily as more data is added, stabilizing around 98% for larger training sets. This indicates that while the model fits the training data perfectly, its generalization to unseen data improves with a larger dataset, eventually reaching a near-perfect score. The slight gap between the training and validation scores suggests some overfitting, but with a sufficiently large dataset, RFC generalizes well.

The confusion matrix (Figure 12a) provides a detailed breakdown of the model's predictions for each class (A-Z). The majority of the predictions are correctly classified, as shown by the high number of instances along the diagonal, particularly the **green cells**, which indicate no misclassifications for those classes.

However, certain classes have significant misclassification rates, as highlighted by the **red and orange cells**. The most notable errors are: **Class U**: 19 instances of U were misclassified as either **R** or **V**, which is the highest number of misclassifications for a single class. This is likely due to the visual similarity between these gestures. In particular, the hand shapes of U and V involve extended fingers with slight variations in positioning, while R also involves overlapping fingers, making it difficult for the model to distinguish between them. Refer to Figures 9, 3, and 7 for a visual comparison of these gestures.

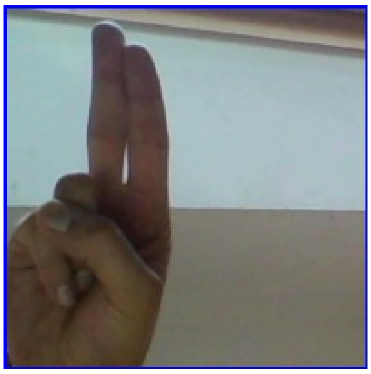


Figure 2: Hand gesture for the letter U.

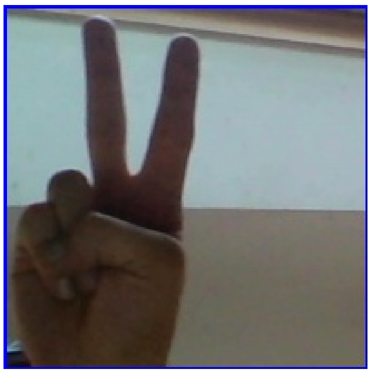


Figure 3: Hand gesture for the letter V.

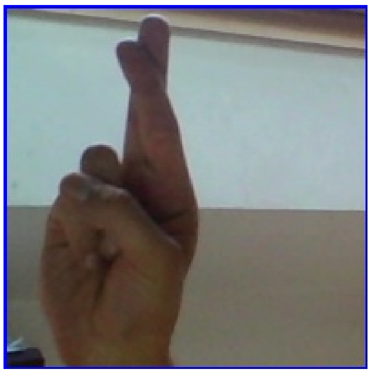


Figure 4: Hand gesture for the letter R.

Additionally, confusion between **M** and **N** was observed, with both classes having a few misclassifications. These errors are likely due to the visual similarity in hand shapes for these letters (as can be seen in Figures 6a and 6b). This suggests that the model struggles to differentiate between gestures with subtle variations, such as those involving overlapping fingers or similar finger positioning.

5.2 MLP Model Results

From Table 1, the MLP classifier achieves an accuracy of 97.57%, with a training time of 50.10 seconds. While this training time is longer compared to KNN and SVM, the MLP delivers strong accuracy and generalization performance, especially as seen in the high top-N accuracy.

The top-N accuracy (Table 2) shows MLP's strong predictive ability, achieving: - **Top-1 Accuracy**: 97.57% - **Top-9 Accuracy**: 99.92%

These results demonstrate the robustness of MLP in providing accurate predictions for ASL gestures.

The learning curve and confusion matrix provide further insights into the performance of the MLP model.

The learning curve (Figure 11b) shows that the training accuracy remains consistently high, above 98%, while the cross-validation accuracy improves steadily as the training set size increases, stabilizing around 97.5%. This indicates that the MLP model performs well in generalization, with a slight gap between training and validation performance, suggesting minimal overfitting.

The confusion matrix (Figure 12b) provides a detailed breakdown of the model's predictions for each class (A-Z). The majority of the predictions are correct, particularly for the **green cells**, which indicate no misclassifications. However, significant misclassifications are highlighted by the **red and orange cells**, where the model struggled to distinguish certain classes.

The most notable misclassifications are: - **Class R**: 65 instances of R were misclassified as **U**, which is the highest number of misclassifications for a single class. - **Class T**: 43 instances of T were misclassified as **L**, showing confusion between these two gestures. - **Class U**: 26 instances were misclassified as **V**, and 15 instances as **K**. - **Class W**: 12 instances of W were misclassified as **V**.

These errors are likely due to the visual similarities between these gestures, making it difficult for the model to differentiate them. Refer to Figures 7, 8, 9, and 10 for a visual comparison of these gestures.

These visual comparisons help explain the model's difficulty in distinguishing gestures that involve subtle finger placements, such as the set (R,K,U,V,W), (M, N) and the set (T,L), where overlapping or extended fingers can easily cause confusion for the model.

5.3 SVM and KNN Model Results

Both the SVM and KNN models exhibit similar error patterns as the MLP model, particularly in the confusion of gestures with subtle finger differences. However, SVM and KNN performed notably better for the set of letters W, X, Y, and Z, where they made fewer errors compared to the MLP model. On the other hand, the MLP model outperformed both SVM and KNN for the remaining letters. Despite these strengths and differences, all three models showed challenges in distinguishing certain subtle gestures.

5.3.1 SVM Model

The SVM model achieves an accuracy of **97.41%**, with misclassification errors resembling those seen in the MLP model but showing distinct differences for certain challenging classes.

From the learning curve (Figure 11d), it is evident that the SVM model exhibits a steady improvement in cross-validation accuracy as the size of the training set increases. The gap between training and cross-validation scores narrows significantly as the model is provided with more data, indicating a reduction in overfitting and improved generalization. Early in training (e.g., with fewer than 10,000 samples), the model shows a larger discrepancy between training and validation scores, reflecting overfitting at small training set sizes. However, as more data are added, the validation score stabilizes, and the model achieves near-optimal performance.

Notably, the classes **O** and the set **(F, G)** remain problematic for the SVM model, as observed from the confusion matrix. These misclassifications could be attributed to overlapping feature distributions in the input space. In contrast, the SVM model shows superior performance in distinguishing gestures **T** and **L**, where the decision boundary formed by the model probably separates these classes more effectively compared to other classifiers.

5.3.2 KNN Model

The KNN model achieves an accuracy of **97.87%**, following a similar trend in error patterns but with distinct characteristics. The learning curve (Figure 11c) for KNN highlights a rapid convergence of cross-validation accuracy, even with a relatively small training set size. Unlike SVM or MLP, the validation accuracy for KNN reaches close to its maximum potential earlier in the curve, indicating that the model performance depends less on large training datasets. This behavior aligns with KNN's nature, where performance is heavily influenced by the local neighborhood defined by the number of neighbors (k) rather than the complexity of the model.

However, the training score for KNN remains higher than the cross-validation score across the learning curve, suggesting a slight overfit. This may occur because KNN inherently memorizes training data and is sensitive to noisy or ambiguous samples.

Interestingly, KNN excels in correctly predicting the gestures **Z**, **Y**, **W**, and **F**, which are challenging for the other models. This can be attributed to KNN's ability to leverage local patterns effectively, especially when clear clusters exist for certain gestures in the feature space. Conversely, KNN struggles with classes where such clusters are not well-defined, leading to highly polarized results—either very accurate predictions or significant misclassification rates.

These insights underline the trade-offs between the SVM and KNN models in terms of their learning behaviors and classification patterns, as visualized in Figure 11.

6 Application

The application developed for ASL finger spelling recognition named **Signify** implements the **RFC** model, which was selected after a thorough evaluation of multiple machine learning models. The RFC model outperformed all other models in terms of accuracy (99.19%) and robustness, making it the ideal choice for this application. By leveraging the strengths of RFC, the app delivers high accuracy in real-time gesture recognition while maintaining efficient inference speed and memory usage.

The application was developed collaboratively by a team of seven members, with each contributing to different aspects of the project. My primary responsibility was handling the camera logic, ensuring seamless integration with the gesture recognition model to provide real-time feedback. The source code for the application is available on GitHub at the following link: `https://github.com/Signify-epfl/signify-app`.

This section discusses the application's features, implementation details (focused on camera logic and model integration), and results from a user survey.

6.1 Overview

The main feature of the application is to help users practice Finger Spelling Recognition (FSR) for American Sign Language (ASL). The app provides an engaging and interactive experience with several features accessible from the main screen. Below is an overview of the functionalities:

6.1.1 Home Screen

The **Home Screen** serves as the central hub for accessing the app's main features, as shown in Figure 13a.

- **Try Hand Signs:** A button labeled *"Try Hand Signs Here"*, as highlighted by the *red rectangle* in Figure 13a, navigates the user to a practice screen, as shown in Figure 13b. On the practice screen, users can view hand signs for specific letters using a **LetterSwitchDictionary**, as highlighted by the *yellow rectangle* in Figure 13a, which displays the letter along with its corresponding icon.

  Additionally, on the practice screen, users can access a live **camera preview**, allowing them to replicate the signs in real-time. The app provides continuous and instant feedback by analyzing the user's hand gestures. A dedicated box on the screen displays the detected sign, ensuring the user knows whether their gesture matches the expected sign. Note that Figure 13b includes an **emulated camera preview**, as the image was captured during development.

- **Exercise Selector:** Users can select exercises categorized by difficulty levels (easy, medium, and hard), allowing progressive learning, as highlighted by the *white rectangle* in Figure 13a

  - For **easy** exercises, users are provided with a set of three words, each consisting of three letters, as illustrated in Figure 13e.
  - For **medium** exercises, users are given three words, each containing 5 to 7 letters.
  - For **hard** exercises, users must finger spell an entire sentence.

  Here is an example of how the user should proceed:

  1. Check the icon to perform the gesture in front of the camera.
  2. The app processes the input and verifies if it matches the intended one.
  3. Move on to the next letter of the current word until three words are completed.
  4. The exercise is marked as complete when the user successfully finger-spells the entire set.

- **Letter Dictionary:** This feature provides a comprehensive and scrollable view of the entire ASL alphabet. Each letter is displayed in a card format, similar to the example for **Letter A** in Figure 13a. The cards include:

  - A picture used to train the model.
  - Icons representing the hand sign for the letter.
  - Tips and instructions for properly forming the hand sign.

  The dictionary is highlighted by the *green rectangle* in Figure 13a and is implemented using a **Lazy List** to ensure efficient resource management, particularly when displaying all 26 letters of the alphabets.

- **Quiz Mode:** Accessible via the *pink square* at the top of the app in Figure 13a, the **Star** button allows users to test their knowledge of ASL. In this mode. Users interact with the quiz by selecting the correct answers for presented hand signs. This feature encourages learning and retention by associating visuals with correct hand sign gestures as shown in Figure 13c

- **Quest Mode:** At the top of the app, as highlighted by the *violet square* in Figure 13a, a **Calendar** button allows users to be tasked with finger spelling entire words. The app provides a helpful infinite loop video demonstrating how to finger spell the word, guiding users through the corresponding hand signs as shown in Figure 13d.

- **Feedback:** A dedicated section, highlighted by the *orange square* in Figure 13a, allows users to send their feedback to help improve the app. The screen displayed after clicking on this section is shown in Figure 13f.

6.1.2 Challenge Screen

The **Challenge Screen** enables users to send challenges to their friends and participate in competitions at any time and place. Two types of challenges are available:

- **Chrono Mode:** A best-of-three (Bo3) mode that focuses on speed, with three rounds to determine the fastest and most accurate user.
- **Sprint Mode:** A timed mode where users try to finger spell as many words as possible within 30 seconds, testing both speed and consistency.

Additionally, users can view the **Challenge History**, which includes:

- Results of previous challenges (e.g., 2-0 or 2-1 wins/losses).
- Detailed statistics such as completion time, number of rounds won, and more.

6.1.3 Profile Screen

The **Profile Screen** displays all shared user and app-related information, including:

- Username.
- Statistics (e.g., total challenges, exercises and quests completed.
- Profile picture.
- List of friends.

6.2 Model Integration

The app was developed using **Kotlin** and is designed exclusively for Android devices. It utilizes:

- **Google Authentication:** Allows users to securely log in and synchronize their data across devices.
- **Firebase Firestore:** For managing user data, challenges, and feedback.
- **MediaPipe for Android:** For real-time hand landmark detection and gesture recognition.

To enable the phone to use the model for recognizing hand signs, three key components are required:

1. **Loading the Model:** The initial model was stored in `.pickle` format, which is incompatible with Android devices. To resolve this, the model was converted into the `.onnx` format, known for its efficiency and suitability for mobile platforms. Once converted, the model was integrated into a **data class** within the data layer. This

data class is responsible for managing the model and ensuring smooth interaction with the **ViewModel**. The **ViewModel**, adhering to the **MVVM architecture**, acts as a mediator, bridging the user interface (UI) with the data layer, and ensuring synchronization between the model's predictions and the app's UI components.

2. **Extracting Hand Landmarks:** Using the MediaPipe library for Android, a task was created to process the input stream (camera feed) and detect any hands present. The library efficiently extracts hand landmarks, which are key points used for further processing.

3. **Computing the Output:** After obtaining the hand landmarks, the app runs a session using the `onnx` library to compute the output. This session processes the landmarks and predicts the hand sign that the user is making, providing real-time feedback.

This integration of `onnx` with MediaPipe, coupled with the MVVM architecture, ensures seamless performance and efficient interaction between the app's components.

6.3 Survey

To evaluate the app's effectiveness and user satisfaction, a survey was conducted with a diverse group of users, including beginners, intermediate learners, and advanced signers. A total of 10 participants responded to the survey forms. The survey aimed to gather feedback on various aspects of the app, such as its usability, accuracy, and features.

Survey Design The survey was structured to capture both quantitative and qualitative feedback. Participants were asked to evaluate the app's performance and provide suggestions for improvement through the following questions:

- **Accuracy and Performance:**
  - How accurate do you find the app in recognizing your hand gestures for different letters?
  - Did the app correctly recognize the finger-spelled letters you tried to demonstrate most of the time?
  - Were there any particular letters that the app struggled to recognize? If yes, which ones?
  - How often did you have to retry a letter because the app misinterpreted your gesture?
  - Did the recognition speed meet your expectations for real-time feedback?
  - How responsive was the app when recognizing your hand gestures? (Scale of 1-5)
- **Features and Effectiveness:**

  - Which features of Signify do you find most effective in helping you gain experience with ASL fingerspelling? (Select all that apply)

- **Suggestions and Feedback:**
  - What improvements would you suggest for enhancing the accuracy of finger spelling recognition?
  - After using the app, are you able to spell your full name using American Sign Language (ASL) fingerspelling? If yes, how confident are you in your ability?
  - Other remarks.

Survey Results The results of the survey provided valuable insights:

- **Accuracy:** 100% of users rated the app's gesture recognition accuracy as 4 or 5, highlighting its reliability in identifying hand signs. However, some users noted occasional challenges with specific letters, such as *"U"*, *"R"*, and *"V"*, reflecting findings from the confusion matrix. Users also mentioned that these errors were more frequent under varying lighting conditions or when their hand positions were not perfectly aligned with the camera.
- **Ease of Use:** The app's interface was well-received, with 100% of users rating it as intuitive and user-friendly. Users appreciated the straightforward navigation and the clear instructions provided for each task, enabling an effortless learning experience.
- **Responsiveness:** Most users appreciated the app's real-time feedback and smooth performance. A small subset suggested increasing a bit the latency during recognition of repeated letters (a *"p" "p"*le).
- **Feature Usefulness:** The most frequently used features were the quests, exercises and quizzes with 90% of users reporting that these elements were effective in helping them practice and gain confidence in ASL finger spelling.

User Suggestions Participants provided constructive feedback to improve the app:

- Expand the training dataset to include more diverse hand shapes, skin tones, and lighting conditions to improve robustness and fairness.
- Improve recognition accuracy for certain edge cases, particularly letters with similar hand shapes, such as *"R"* and *"V"*.
- Add more gamified elements, such as leaderboards and rewards, to enhance engagement and provide users with a sense of progress.

Conclusion The survey confirmed that the app is effective in helping users learn ASL finger spelling, with strong ratings for accuracy and usability. Most users found the app to be a valuable tool for practicing ASL, particularly due to its engaging features like daily quests, quizzes and exercises. Feedback also revealed areas for improvement, such as enhancing recognition accuracy for similar gestures and expanding the dataset to accommodate greater diversity.

Future updates will prioritize incorporating user suggestions, such as additional gamification elements, targeted practice modes, and improved responsiveness. These enhancements aim to provide a more inclusive, engaging, and efficient learning experience for users, ensuring the app continues to meet and exceed their expectations.

7 Conclusion

This paper presented the development of **Signify**, a mobile application for recognizing American Sign Language (ASL) finger spelling gestures. The project evaluated the performance of four machine learning models—Random Forest (RFC), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN)—to identify the best model for real-time ASL recognition. The **RFC model** emerged as the top performer, achieving an impressive accuracy of 99.19%, and was subsequently integrated into the application.

The app's development was a collaborative effort by a team of seven, with contributions spanning various technical components. The integration of the model required significant transformations, such as converting the initial `.pickle` model into the `.onnx` format to ensure compatibility with Android. The `onnx` model was coupled with MediaPipe for real-time hand landmark detection and implemented using the MVVM architecture to maintain a clean separation of concerns between the UI and the data processing layers.

The application provides a range of features to enhance user learning and engagement:

- Real-time practice with instant feedback through a live camera preview.
- Guided exercises and quizzes with progressive difficulty levels.
- A letter dictionary for comprehensive learning.
- A gamified experience with challenge modes to compete with friends.

User feedback, collected through a survey of 10 participants, highlighted the app's strengths in accuracy, responsiveness, and ease of use. Constructive suggestions included improving gesture recognition for visually similar signs and expanding the dataset to encompass more diverse conditions.

Signify demonstrates the potential of machine learning to bridge communication gaps between signers and non-signers. Future enhancements will focus on refining gesture recognition, adding new features like J and Z hand sign **movement** detection, and expanding accessibility to include additional sign languages. This work lays the foundation for broader applications of gesture recognition in education, accessibility, and social inclusion.

References

[1] Sharon Baker et al. “The Importance of Fingerspelling for Reading: Research Brief”. In: (2021). URL: https://www.researchgate.net/profile/Sharon-Baker-3/publication/351349269_The_Importance_of_Fingerspelling_for_Reading_Research_Brief/links/6092bd2c458515d315fbf1c1/The-Importance-of-Fingerspelling-for-Reading-Research-Brief.pdf.

[2] Nicolas Pugeault and Richard Bowden. “Spelling It Out: Real-Time ASL Fingerspelling Recognition”. In: *Personal Pages at Surrey* (2011). URL: https://personalpages.surrey.ac.uk/r.bowden/publications/2011/ICCV/WS/PugeaultBowden2011b.pdf.

[3] Viktor Mazalov and Vladimir Protasov. “Sign Language Recognition and Translation with Transformers”. In: (2020), pp. 5976–5983. URL: https://aclanthology.org/2020.lrec-1.737.pdf.

[4] Harita Joshi et al. “Real-Time Sign Language Recognition and Sentence Generation”. In: *SSRN* (2024). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4992818.

[5] Ala’ Alyabrodi et al. “Enhancing the Performance of Sign Language Recognition Models Using Machine Learning”. In: *2023 24th International Arab Conference on Information Technology (ACIT)*. 2023, pp. 01–07. DOI: 10.1109/ACIT58888.2023.10453781.

[6] Lionel Pigou et al. “Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2017). URL: https://ieeexplore.ieee.org/document/7830097.

8 Appendix

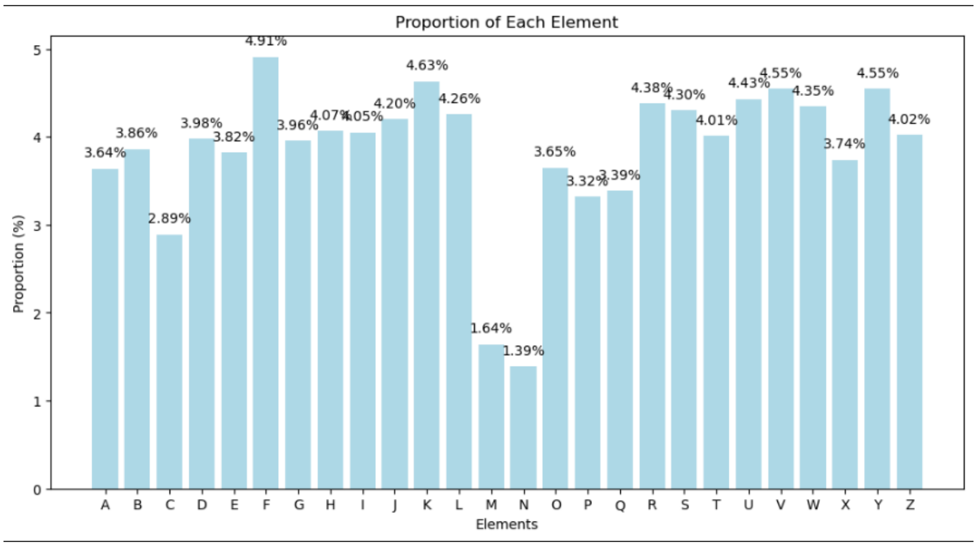
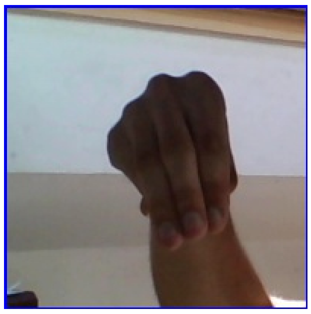


Figure 5: Class proportions after the data extraction process.


(a) Sign for "M" with overlapping fingers.

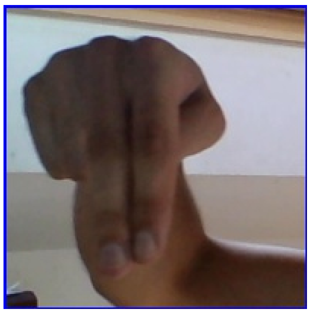

(b) Sign for "N" with overlapping fingers.

Figure 6: Examples of "M" and "N" hand signs that are challenging for landmark detection.

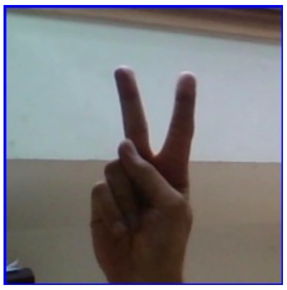


Figure 7: Hand gesture for the letter K.

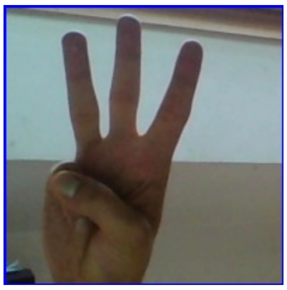


Figure 8: Hand gesture for the letter W.

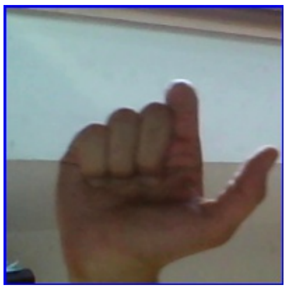


Figure 9: Hand gesture for the letter T.

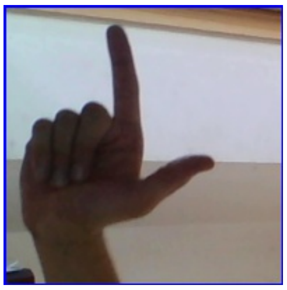
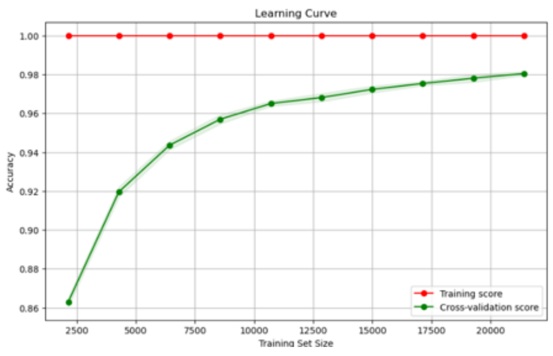
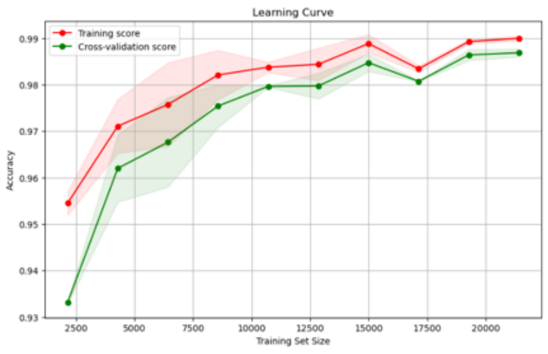


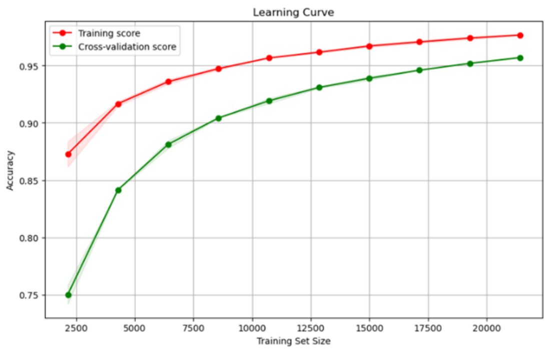
Figure 10: Hand gesture for the letter L.



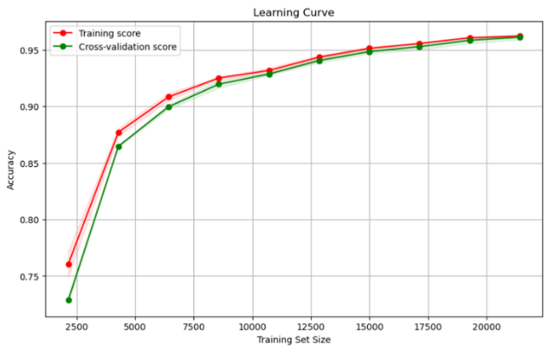
(a) RFC Learning Curve



(b) MLP Learning Curve

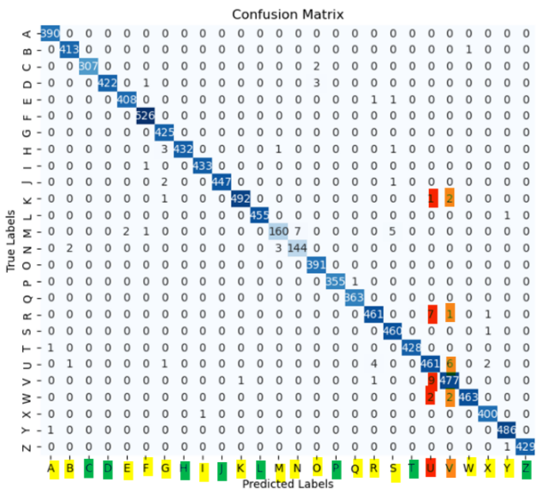


(c) KNN Learning Curve

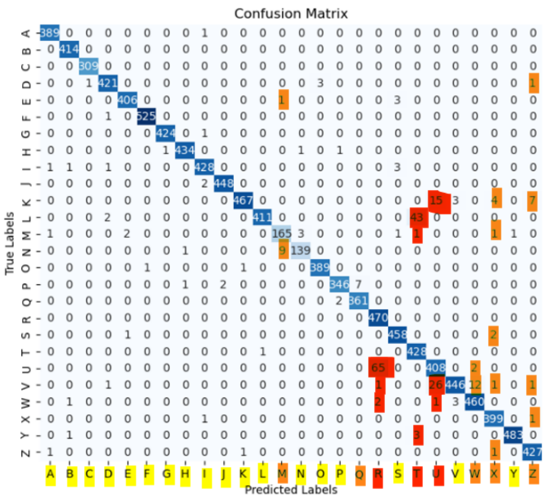


(d) SVM Learning Curve

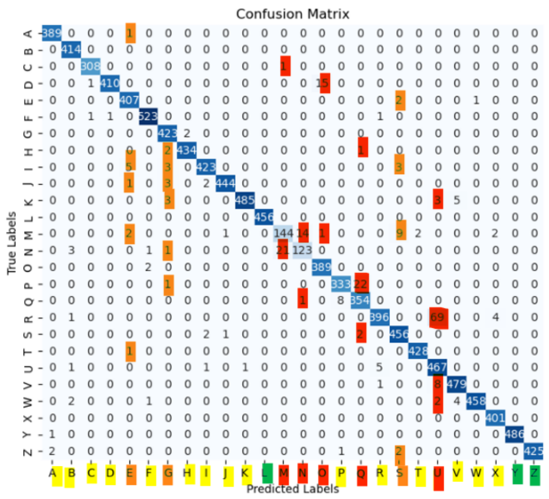
Figure 11: Comparison of Learning Curves for RFC, MLP, KNN, and SVM models.



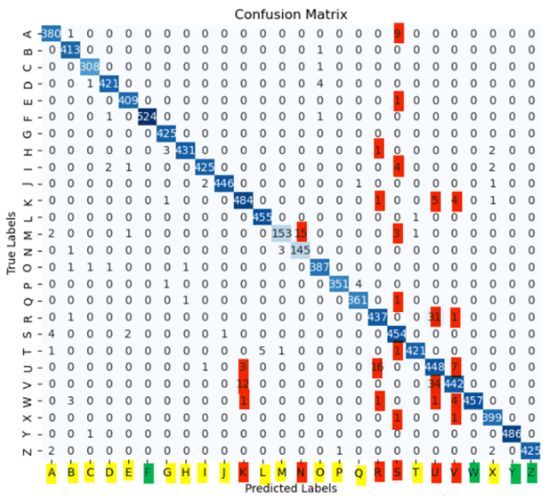
(a) RFC Confusion Matrix



(b) MLP Confusion Matrix

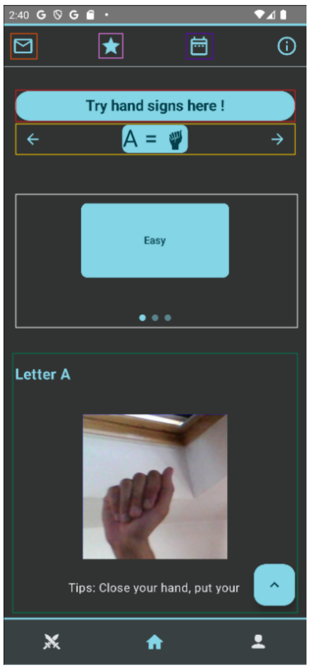


(c) SVM Confusion Matrix

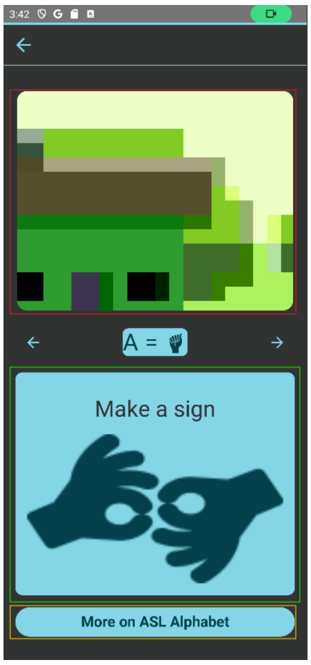


(d) KNN Confusion Matrix

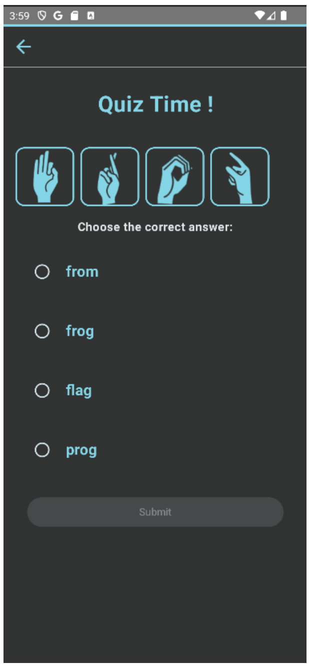
Figure 12: Comparison of Confusion Matrices for RFC, MLP, SVM, and KNN models.The color scheme indicates the magnitude of errors: red represents 15 or more errors, orange represents 10 to 14 errors, yellow represents fewer than 10 errors, and green indicates no errors.



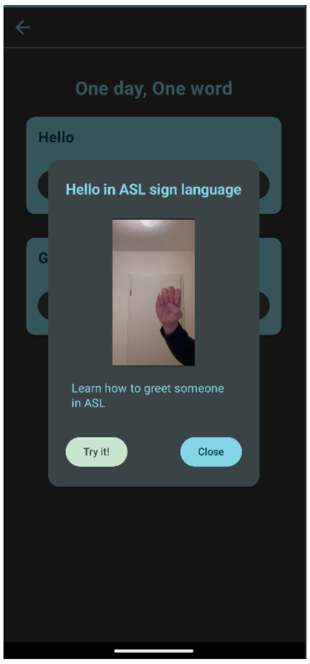
(a) Home Screen showcasing the main features with color indicators.



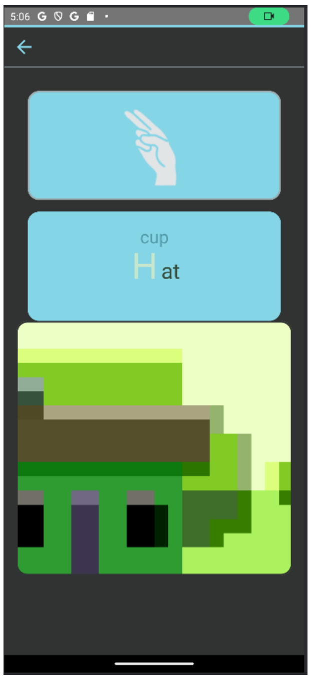
(b) Practice Screen where the user can finger spell.



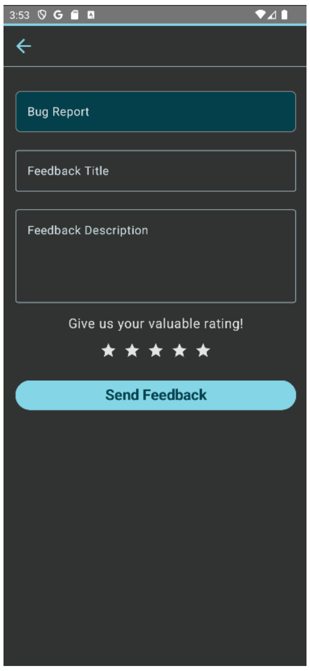
(c) Quiz Screen where the user selects the word corresponding to the successive signs presented.



(d) Quest Screen where the user learns and reproduces daily words through a guided video.

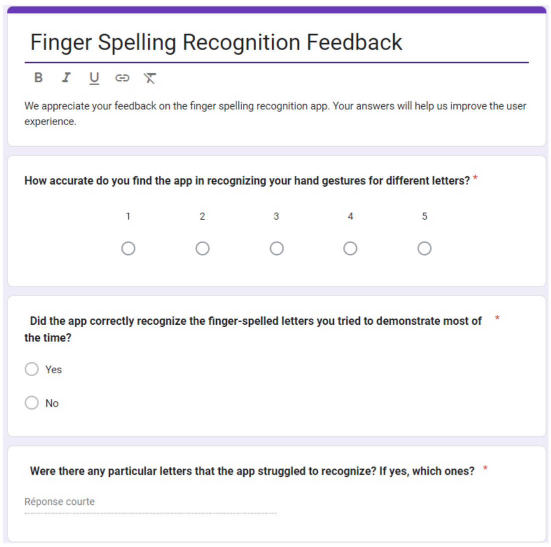


(e) Exercise Screen displaying the current letter to finger spell, highlighted in uppercase, with the next word to finger spell shown faintly in the background.



(f) Feedback Screen where users share their feedback about the app.

Figure 13: Screens of the Signify App, demonstrating various features: Home Screen, Practice Screen, Quiz Screen, Quest Screen, Exercise Screen and Feedback Screen.



Finger Spelling Recognition Feedback

We appreciate your feedback on the finger spelling recognition app. Your answers will help us improve the user experience.

How accurate do you find the app in recognizing your hand gestures for different letters? *

1	2	3	4	5
○	○	○	○	○

Did the app correctly recognize the finger-spelled letters you tried to demonstrate most of the time? *

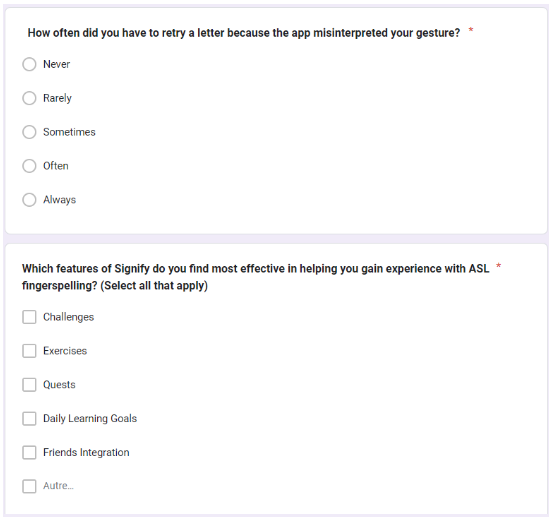
○ Yes

○ No

Were there any particular letters that the app struggled to recognize? If yes, which ones? *

Réponse courte

Figure 14: First Survey Questions



How often did you have to retry a letter because the app misinterpreted your gesture? *

○ Never

○ Rarely

○ Sometimes

○ Often

○ Always

Which features of Signify do you find most effective in helping you gain experience with ASL fingerspelling? (Select all that apply) *

☐ Challenges

☐ Exercises

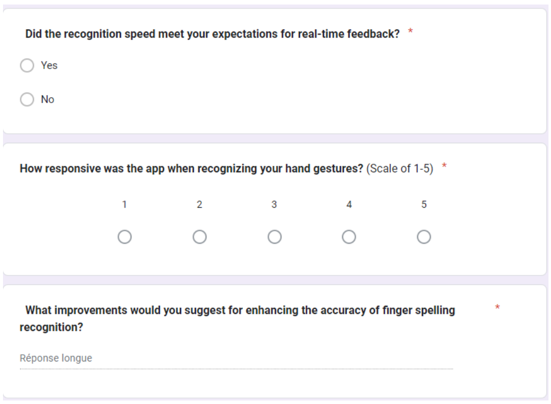
☐ Quests

☐ Daily Learning Goals

☐ Friends Integration

☐ Autre...

Figure 15: Second Survey Questions



Did the recognition speed meet your expectations for real-time feedback? *

○ Yes

○ No

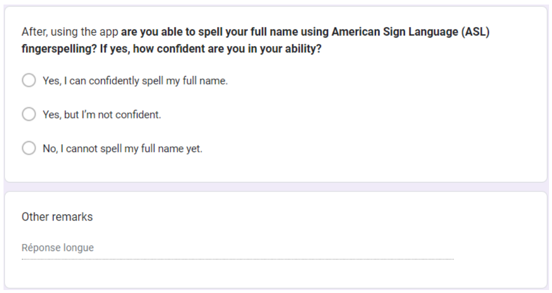
How responsive was the app when recognizing your hand gestures? (Scale of 1-5) *

1	2	3	4	5
○	○	○	○	○

What improvements would you suggest for enhancing the accuracy of finger spelling recognition? *

Réponse longue

Figure 16: Third Survey Questions



After, using the app **are you able to spell your full name using American Sign Language (ASL) fingerspelling? If yes, how confident are you in your ability?**

○ Yes, I can confidently spell my full name.

○ Yes, but I'm not confident.

○ No, I cannot spell my full name yet.

Other remarks

Réponse longue

Figure 17: Fourth Survey Questions

Figure 18: Survey Questions used to test the user experience: Ten total questions visualized.