# Interactive Tool for Sign Language Alphabet Acquisition

**Rebeca Costache**
Faculty of Computer Science, "Alexandru Ioan Cuza" University
General Berthelot, No. 16, Iași, Romania
rebeca.costache@student.uaic.ro

**Adrian Iftene**
Faculty of Computer Science, "Alexandru Ioan Cuza" University
General Berthelot, No. 16, Iași, Romania
adiftene@info.uaic.ro

## ABSTRACT
Sign language education remains inadequately supported by existing interactive learning platforms, often lacking inclusive and engaging experiences. This paper presents a human-centered application aimed at addressing this problem by employing the Kinect, Microsoft's gesture-based sensor, to facilitate immersive sign language instruction. Furthermore, integrating the Kinect with Machine Learning (ML) algorithms can boost the pedagogical progression in sign language acquisition, as learning is most efficient when theory is combined with active practice. In the development process of this application, various models were trained on three different datasets in order to obtain the best performance in learning and predicting the American Sign Language (ASL) alphabet. By integrating Human-Computer Interaction (HCI), this application represents the first step in supporting not only members of the Deaf and Hard of Hearing (DHH) communities, but also individuals from broader linguistic, educational, and professional backgrounds who seek to learn sign language for communication and education, or to advocate for Deaf rights and accessibility.

### Author Keywords
Gesture-based interaction; Kinect; neural networks; educational technology; American Sign Language; human-computer interaction.

### ACM Classification Keywords
I.5.1 Models: Neural nets.
K.3.1 Computer Uses in Education: Computer-assisted instruction (CAI).

### General Terms
Human Factors; Languages; Performance.

**DOI:** 10.37789/icusi.2025.2

## INTRODUCTION
Language deprivation refers to the condition of a child that lacks full access to a natural language during critical years of development. Unfortunately, this phenomenon is often present in DHH children who present delayed skills in their first language [4]. Unlike most children who acquire language naturally thorough constant exposure, DHH children face a mismatch between their perceptual abilities and their environment, leading to difficulties in social development and cognitive delays. The solution described in this paper to the phenomenon of language deprivation among DHH communities is the development of Sign Textify: an interactive tool designed to teach users the American Sign Language alphabet by providing a well-rounded and comprehensive learning experience. The tool offers users the flexibility to engage in both practice and assessment modes.

In the practice mode, users are guided with visual references displaying the correct hand positions for each letter and are asked to attempt to replicate the sign. Upon successful execution, the application moves on to the next letter in alphabetical order. In the assessment mode, the users must sign letters without any hints. The letters appear in randomized order to prevent reliance on muscle memory, ensuring a more robust learning experience. Throughout both modes, the tool renders the frames captured by the Kinect sensor [15] on a window in order to give real-time feedback to users. Being able to see themselves in the window like they would in a mirror allows them to self-correct, while also speeding up the educational process.

Many papers, articles, and people talk about the benefits DHH people have if the gaps in communication could be eliminated [5, 7, 9, 14]. Effective communication channels promote accessibility, foster inclusivity and equal opportunities, and remove obstacles that prevent the deaf from engaging with the broader community. Although all these are true, this perspective idealizes hearing individuals, implying that non-hearing people must integrate into hearing communities rather than creating mutual inclusion and understanding. What if we shift our perspective and explore the advantages of being able to communicate with the DHH community? Rather than focusing on inclusion as a one-sided effort, we can recognize that learning sign language and creating meaningful connections enrich both hearing and non-hearing people alike.

## PREVIOUS APPROACHES
Before proposing a new solution to the challenges described in the previous section, it is essential to understand the landscape of existing attempts. Over the years, developers and engineers have explored various methods to bridge the communication gap between sign language users and non-signers. Despite decades of research, no existing system has fully succeeded in making sign language translation seamless, accurate, and accessible, highlighting the need for further innovation. This section reviews the most prominent attempts in easing the process of learning and translation of sign language into natural language, to discover their outcome related to the problem in question, to select the most effective solutions and to avoid less optimal ones.

### Rocket Sign Language [12]

The platform provides videos demonstrating ASL signs, which users can slow down, speed up, or rewind to perfect their signing skills over 7 learning modules, each module consisting of 7 lessons (see Figure 1). It offers a structured course with lifetime access, allowing learners to revisit lessons anytime. At the end of every lesson, an evaluation assesses how many words the user has retained. Each evaluation earns points, which are recorded on a leaderboard showcasing the scores achieved by users over the past 24 hours. Additionally, the platform features a streak count to encourage consistent learning, and it covers essential aspects of Deaf culture and etiquette to ensure learners can communicate respectfully and effectively within the community. Rocket Sign Language offers several advantages that contribute to its accessibility and functionality. It provides structured lesson plans for learning thirteen spoken languages in addition to ASL. Each language includes its own set of learning tools. For ASL, the main tool is a dictionary that links each word to a corresponding video demonstration of its sign. Furthermore, the platform supports mobile access, enhancing its convenience for learners on the go. However, certain limitations affect its pedagogical effectiveness. The app lacks a dedicated practice tab that allows for independent skill rehearsal separate from guided lessons. Additionally, it does not feature real-time evaluation of signing accuracy, relying instead on self-assessment, which may hinder immediate correction and learning feedback.

### Lingvano [8]

The platform offers an engaging and interactive approach to learning, making sign language accessible to everyone, regardless of prior experience. One of the most unique aspects of Lingvano is the “mirror” feature. The idea behind this feature is that when the user is shown a new sign, they can choose to open their laptop camera, in order to see themselves perform the new sign. The image of the new sign is placed next to this mirror, allowing users to refine their gestures in real-time by trying to mimic the sign in the photo and autocorrecting their hand position.

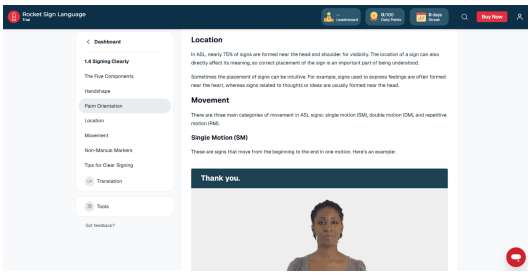


**Figure 1. Dashboard of the Rocket Sign Language web application.**

Furthermore, this application prioritizes hands-on learning through well-organized dialogue exercises (see Figure 2).

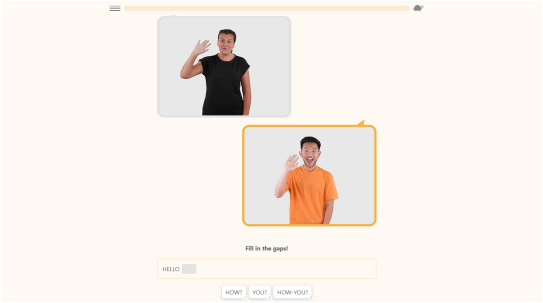


**Figure 2. Dialogue of Chapter 1, Lesson 5 on Lingvano web application.**

Instead of just learning individual signs, users experience conversations that reflect real-world interactions, making the learning process feel more authentic. Similar to Rocket Sign Language, the platform provides insights into Deaf culture and etiquette, ensuring that learners gain a well-rounded understanding of how to communicate effectively and respectfully. As additional advantages, Lingvano is gamified to encourage interaction and motivation and it accommodates three sign languages (American, English, and Australian), making it more broadly accessible than other similar applications. Learners also benefit from an integrated dictionary, a free trial period with flexible billing options, and a mobile app for portability. However, another similarity between Lingvano and Rocket Sign Language is the drawback of not having a real-time automated evaluation of signing accuracy, which means users must rely on self-assessment, without immediate corrective feedback given directly from the platform.

### Sign School [13]

The website features a Sign of the Day, which introduces users to new vocabulary daily, and a Fingerspelling Game for practicing handshapes, depicted in Figure 3. The platform delivers ASL grammar lessons designed to support learners across a range of proficiency levels, from beginner to advanced, ensuring a comprehensive understanding of sentence structure and communication nuances. One of its standout features is the ability to customize learning paths, allowing users to progress at their own pace. Whether used for casual learning, academic purposes, or professional development, Sign School provides a flexible and engaging way to study ASL without requiring formal enrollment. As a not logged-in user, it allows access to three different dictionaries (one with all the signs, one ordered by topic, and one ordered by handshapes), whereas a logged-in user can customize his/her dictionaries (personal and favorites dictionaries). In each lesson, the user has access to “Topics”, which enables creating a custom learning path, “Grammar” explanations, and “Tools” such as the fingerspelling game and the dictionaries. Additionally, the dashboard of the application contains a series of metrics indicating the activity on this platform: the current and record streak, fingerspelling

accuracy and score, lesson accuracy, and the user's level based on points earned.

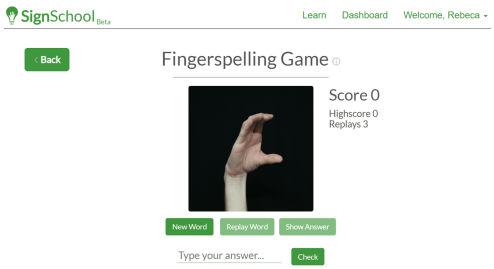


**Figure 3. Interactive fingerspelling game screen from Sign School.**

The application empowers teachers to design custom lessons tailored to their students' needs and enables users to select difficulty levels appropriate to their skill. Similar to Lingvano, Sign School includes a mirrored camera feature for self-monitoring. Users can also share sign lessons with other users, encouraging collaborative learning. Additional features include a spelling-focused practice hub, full mobile access, and no cost to use. Consistent with the other applications described in this section, one significant drawback is the absence of real-time accuracy evaluation, leaving users to depend solely on self-assessment without immediate corrective feedback coming in an automated manner from the platform.

### Pocket Sign [11]

The gamified platform allows access to an abundance of learning resources. With new lessons released each week, the user can gain points after each lesson, which can then be used to buy special "superpowers" such as retrying a sign, skipping a sign, or activating a tool that reveals the correct answer. Additional resources include a dictionary that maps words to their sign language representations through short videos, and fingerspelling games which need access to the laptop camera to check if the user is signing correctly (see Figure 4).

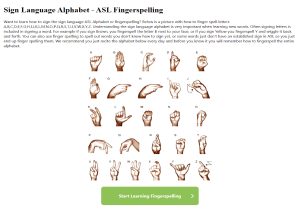


**Figure 4. Fingerspelling game of the Pocket Sign web application.**

If a sign is recognized as correct, the game automatically advances to the next item; if not, it remains on the current sign until the user performs it accurately. As advantages, the application offers a free introductory lesson along with the option for a 14-day free trial, making it accessible for new users. It includes complimentary fingerspelling games that help reinforce learning through interactive practice. Notably, these games provide real-time evaluation of signing accuracy, allowing users to receive immediate feedback and adjust their performance accordingly. Nevertheless, the platform would benefit significantly from enhancements to its user interface and user experience design to make it more visually appealing and intuitive.

## SIGN TEXTIFY

The solution described in this paper to the lack of educational resources for learning sign languages application is Sign Textify. It is designed as two mini-games, allowing users to choose between a practice session and an assessment session. In practice mode, the user is prompted to sign the letters of the American alphabet, starting with A and finishing with Z, and is provided with a support image that presents the corresponding sign. If the user correctly signs the letter, the application proceeds to the following letter, displaying a "well done" message on the screen for a few seconds. If the sign is not correctly performed, the application waits until the user gets it right. Additionally, the user is given as an escape route a skip button, because each student has their own learning pace, and assigning responsibility for the pace to the student fosters autonomy. In the assessment part, the letters of the alphabet are shuffled, and no aid is offered except the skip button. This allows the user to assess how well they have learned.

### System Context

Sign Textify is designed as a server-client application to provide both translation capabilities and an interactive learning experience. As illustrated in Figure 5, the system is composed of three main components: the server, the client, and the Kinect sensor, each of which will be described in detail in the following subsections. The user's first interaction in Sign Textify is with the Kinect sensor, as the signs will be performed in front of it. The .NET client continuously waits for color feed from the sensor and sends it to the Python Server for processing. The server is responsible for extracting hand and finger position coordinates, and if the pattern of these coordinates is recognized, it displays the resulting letter on the screen. The user can see the outcome in real-time and adjust the positions of the hands and fingers to either try again or sign the next required sign. If the user gets stuck, the interface features a skip button that allows them to jump to the following letter, as depicted in Figure 6.

#### *Kinect*

The Kinect for Xbox 360, as shown in Figure 7, also known as Kinect version 1, is a motion-sensing device developed by Microsoft that enables interaction through body movements and voice commands. Equipped with depth sensors, skeletal tracking, gesture recognition, and voice control, it allows users to interact with applications hands-free. In software development, Kinect has been employed for human-computer interaction, virtual reality, robotics, and machine vision [1, 2, 3, 6, 10].

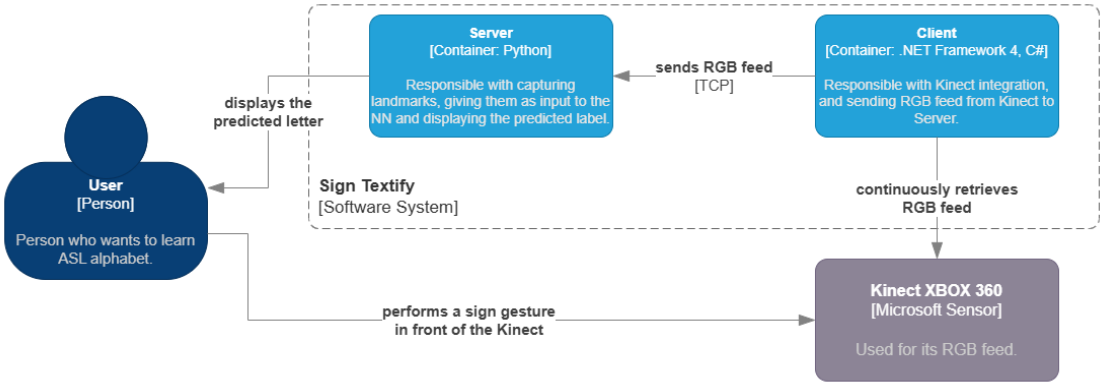


**Figure 5. Context diagram to showcase main interactions between the user and the software system.**

In Sign Textify, the Kinect is used as a camera. Initially chosen for its skeleton tracking library, the Kinect did not prove to be as helpful as intended. It was clear that the library lacks the expressiveness required to accurately distinguish between different signs. This limitation arises from the fact that it tracks only two key points on each hand: the wrist and the hand as a whole (see Figure 8).

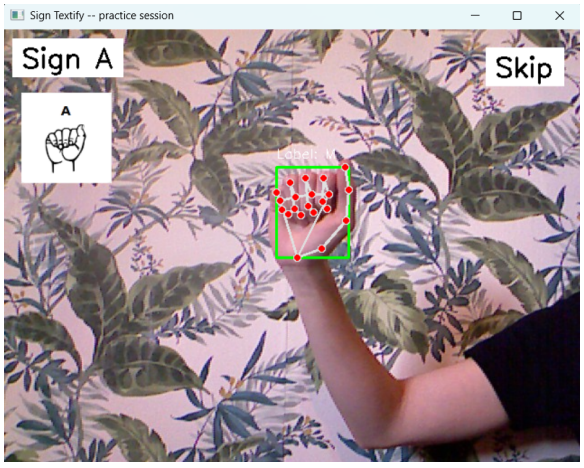


**Figure 6. Window of a practice session in Sign Textify, prompting the user to sign letter A.**

**Figure 7. Kinect Xbox 360 sensor from Microsoft.**

*Client*

The primary purpose of the client is to connect to the Kinect sensor and capture RGB frames, which are then delivered to the server for processing. The client is a C# console application that streams color frames from the Kinect sensor to a remote server. Upon startup, the application initializes the Kinect sensor, then it attempts to establish a Transmission Control Protocol (TCP) connection to a server, and once connected, the application listens for new color frames from the Kinect. For each frame, the client captures the raw image data, compresses it into JPEG (Joint Photographic Experts Group) format, and transmits it to the server, enabling efficient network usage and faster data transfer while maintaining reasonable image quality. For an application that aims to translate signs into letters in real-time, speed is crucial to minimize latency. The application uses a custom communication protocol to transmit image data from the client to the server. Each message sent consists of a 5-byte header followed by the payload (the compressed image data). The first byte indicates the data type, where a value of 1 represents a Red Green Blue (RGB) frame and a value of 2 means an infrared depth frame. The next 4 bytes specify the size of the payload in bytes that allows the server to know exactly how many bytes to read for the image data that follows.

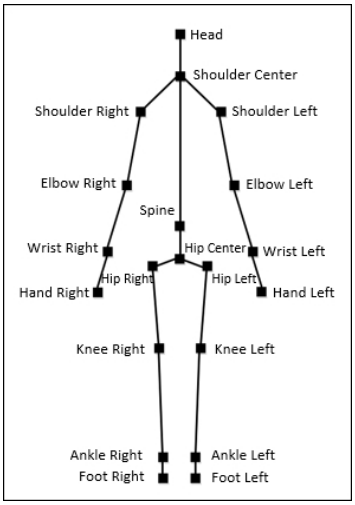


**Figure 8. The joints of the skeleton tracking system from the Kinect.**

*Server*

The server plays a crucial role in processing frames received from the client. Its primary responsibilities include detecting the presence of a hand in each frame, extracting 21 key landmarks of the palm using Media Pipe, and running inference on those extracted coordinates. Based on the analysis, the server then generates and displays a bounding box around the detected hand, presents the precise coordinates of the identified landmarks, and provides the inference result in the form of a label. The Hand Tracking module has the function of drawing the bounding box of the hand, the connections between the landmarks, and the predicted label on a given frame (see Figure 6). This is done using OpenCV functions for rendering rectangles and putting text on the window, and Media Pipe for drawing the connecting lines between the points of all the landmarks. Each frame is processed by another function from this module, which converts the image to RGB format, processes the image using the Hands class from Media Pipe, and returns an array of tuples consisting of the x, y, and z coordinates if a hand is recognized in the picture; otherwise, it returns nothing. The next important module of the server is the Inference module that manages interactions with the model, including loading the model, normalizing landmarks to fit the requested size and type for its input, and predicting labels. All trained models were stored in Open Neural Network Exchange (ONNX) format, an open standard for the representation of deep learning or machine learning models. The normalization of landmarks involves converting the coordinates from a flat array into a 2-dimensional NumPy array (one row and multiple columns), to feed it as input to the model. After this step, the normalized coordinates are sent to the model, and the predicted label is forwarded to the main loop of the application, which determines whether the user has correctly signed the current letter.

## Datasets

For training the neural network of the application, data was taken from an ASL Alphabet dataset on Kaggle, which is composed of both training and test batches. The training dataset is an extensive collection of labeled images representing all letters of the alphabet: 3000 images per sign, 29 signs (including signs for “space”, “delete”, and “nothing”), which means a total of 87,000 color images of hands signing each letter. Each letter has its folder, containing thousands of images captured under various lighting conditions, backgrounds, and hand orientations to offer a diverse range of options. For test, the dataset has one image per letter, resulting in exactly 29 images. To create the most efficient model and minimize training time, the “nothing” label was ignored, as it was not necessary for this application. Thus, the final model predicts a total of 28 different possible labels, and the dataset used is composed of 84,000 images. A new dataset was created consisting of coordinate arrays of the 21 3-dimensional landmarks extracted using Media Pipe. This new dataset was stored using Hierarchical Data Format version 5 (HDF5), a method for storing large amounts of data in a single file. HDF5 files can contain multidimensional arrays, tables, images, and metadata, all organized in a hierarchical structure similar to folders and files in a filesystem, enabling fast loading and easy access during model training and evaluation.

Due to preprocessing, the dataset suffered a loss in the number of training and test images, as Media Pipe’s model failed in recognizing the hand in all images. In other words, the training dataset is a matrix with only 63,673 rows (out of 84,000 total number of images) and 63 columns (3 coordinates for each of the 21 landmarks). To gain a better understanding of how many images were lost after preprocessing, a script was created to generate a markdown file containing statistics on the data. Firstly, Figure 9 shows a comparison between the original number of training images and the final number of successfully preprocessed images. Extracting the landmarks resulted in the loss of exactly 20,327 train instances. This means that approximately 75.80% of the original train dataset remained after preprocessing. The statistics revealed that the test set was not as fortunate: only 51.85% of the original test dataset was kept. Secondly, for greater precision, the script also generated statistics for each letter, including the number of lost instances after preprocessing and the percentage of the original data that remained for that letter (see Figure 10).

a) Original train data

| Label 1 | Count 1 | Label 2 | Count 2 | Label 3 | Count 3 |
|---|---|---|---|---|---|
| A | 3000 | J | 3000 | space | 3000 |
| B | 3000 | K | 3000 | T | 3000 |
| C | 3000 | L | 3000 | U | 3000 |
| D | 3000 | M | 3000 | V | 3000 |
| del | 3000 | N | 3000 | W | 3000 |
| E | 3000 | O | 3000 | X | 3000 |
| F | 3000 | P | 3000 | Y | 3000 |
| G | 3000 | Q | 3000 | Z | 3000 |
| H | 3000 | R | 3000 | | |
| I | 3000 | S | 3000 | | |

b) Preprocessed train json data

| Label 1 | Count 1 | Label 2 | Count 2 | Label 3 | Count 3 |
|---|---|---|---|---|---|
| A | 2187 | J | 2578 | space | 1625 |
| B | 2207 | K | 2700 | T | 2349 |
| C | 1988 | L | 2527 | U | 2516 |
| D | 2463 | M | 1565 | V | 2548 |
| del | 1701 | N | 1276 | W | 2456 |
| E | 2308 | O | 2265 | X | 2158 |
| F | 2876 | P | 2042 | Y | 2585 |
| G | 2440 | Q | 2093 | Z | 2351 |
| H | 2393 | R | 2541 | | |
| I | 2384 | S | 2551 | | |

**Figure 9. The number of original train data for each letter (a) and preprocessed train data for each letter (b).**


**Figure 10. The results of preprocessing for label A (a) and label B (b).**

## Models

Throughout the project's implementation, multiple models were trained to increase test accuracy and improve overall performance. The general structure of the fully connected, feedforward deep neural networks implemented had 63 neurons on the input layer, 128 and 64 neurons on the following two hidden layers, and 28 neurons as the output (corresponding to 28 letters, excluding the "nothing" label in the original dataset). As activation functions, ReLU was used for the hidden layers and Softmax for the output layer to produce a probability distribution for all 28 possible labels. The Adam optimization algorithm was applied to increase training efficiency, leveraging its adaptive learning rate and momentum-based approach. The cross-entropy loss function was chosen for measuring the difference between the predicted and actual probabilities, ensuring accurate model adjustments during training. Additionally, a learning rate scheduling algorithm was used to dynamically adjust the learning rate when model improvements stagnate. By decreasing the learning rate when performance hits a plateau, the model avoids unnecessary oscillations and improves convergence.

### *Performance Metrics*

It was brought to my attention by Professor Sebastian Ciobanu that this accuracy is not a reliable metric for evaluating models trained on unbalanced data, such as the dataset used for Sign Textify. In scenarios where one class is significantly more common than others, a model can achieve high accuracy by favoring the dominant class while neglecting the minority class. The class with the most samples in the dataset of coordinates is K, with 2,700 samples, and the class with the fewest samples is N, with only 1,276 samples (1,424 fewer samples than K). This proves that the dataset is considerably unbalanced and needs a weighted form of evaluation. The F1-score provides a more balanced evaluation, as it considers both precision (how many predicted positives are correct) and recall (how many actual positives were correctly identified). Its formula is:

$$F1 = \frac{Precision \times Recall}{Precision + Recall} \times 2$$

where

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

and

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

### *Neural Network Error Analysis*

The error analysis conducted on this model revealed four groups of signs that are very similar to each other. The first two signs discovered that are similar are T and X, where the key difference is the position of the thumb: for T, the thumb is outside the palm, whereas for X, it rests on the fist, as seen in Figure 11. M and N represent another pair of similar signs (see Figure 12). The difference between these two signs lies again in the placement of the thumb: for the letter M, the thumb is tucked under three fingers (index, middle, and ring), whereas for the letter N, the thumb must be placed under only two fingers (index and middle). Letters A and S are both "fist" shapes (see Figure 13). A is a closed fist with the thumb resting alongside the curled fingers, visible from the side. S is also a fist, but the thumb wraps in front of the fingers, similar to how the thumb lies in the letter X. In both, the thumb makes the difference. The last cluster consists of 4 signs: R, V, K, and U (see Figure 14). Letter R is done by crossing the index and middle fingers, with the thumb wrapped in front of the other curled fingers. For V, the index and the middle fingers form a V, with the thumb out of view. For letter K, the fingers spread into the same V shape, while the thumb reaches up to touch the middle finger. Lastly, for letter U, the same two fingers are held straight and together, with the thumb tucked neatly against the side. Generally, the models trained showed difficulties in recognizing all the signs in a similar group. The final network correctly classifies M from the first group, S from the second, and R, V, and K from the last group.

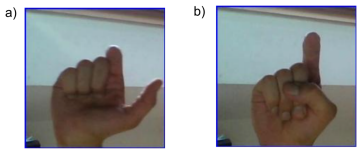


**Figure 11. Images from the original test dataset representing letters T (a) and X (b).**

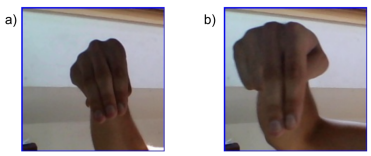


**Figure 12. Images from the original test dataset representing letters M (a) and N (b).**

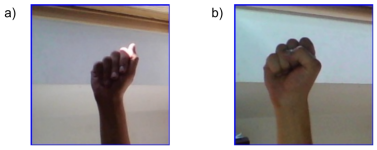


**Figure 13. Images from the original test dataset representing letters A (a) and S (b).**

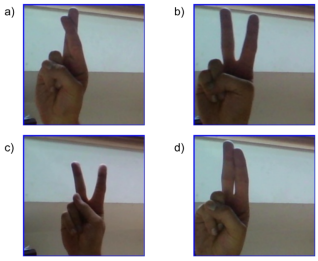


**Figure 14. Images from the original test dataset representing letters R (a), V (b), K (c), and U (d).**

In addition to revealing these similarities between specific handshape groups, the error analysis also showed a limitation in the dataset that significantly impacted the model's performance. The model demonstrated poor generalization when recognizing signs executed with the right hand. This limitation was attributed to a bias in the dataset: all training (and testing) images featured left-hand signs exclusively. As a result, the model lacked samples of right-hand configurations, leading to misclassifications and reduced accuracy in scenarios involving right-handed signs. The last discovery revealed by this error analysis was another bias in the dataset, which involved the interaction with the sensor. The labels predicted for signs performed closer to the Kinect changed as the hand moved further away from it. In other words, the model's performance is sensitive to the distance between the Kinect and the hand. A distance of about 50 centimeters ensured maximum accuracy in predicting letters, while any distance greater than 70 centimeters resulted in wrong predictions. The reason behind this issue is that the training dataset contains images taken at approximately the same distance from the camera. Before addressing the biases and problems revealed by the error analysis, two models were developed using conventional machine learning algorithms to verify the correctness of the neural network implementation and rule out any potential coding errors. The first model employed was an AdaBoost (Adaptive Boosting) classifier, which produced results similar to those of the neural network. To further validate the consistency of the data, an ID3 (Iterative Dichotomiser 3) decision tree was trained and achieved perfect accuracy on the training set. These outcomes indicate that there were no inconsistencies in the dataset and no errors in the implementation of the neural network, reinforcing the conclusion that the model's performance issues were primarily due to the biases revealed by the error analysis.

### *Augmentation Methods*

To remove the biases, two data augmentation methods were used: one targeted the left-hand bias by introducing mirrored right-hand images, and the other addressed class imbalance by augmenting samples from letters that were frequently misclassified. *Firstly*, adding flipped versions of the training and testing images resulted in a lower overall performance, correctly recognizing even fewer signs. One potential explanation for this decline in accuracy is that the dataset size nearly doubled while the architecture and capacity of the network remained unchanged. Although the test accuracy appeared greater, due to the lower training accuracy, the performance did not improve. *Secondly*, to add more images of the incorrectly classified letters, a WPF (Windows Presentation Foundation) application using the .NET Framework 4.7.2 was created. This type of application is widely used for rich user interfaces (UIs) in Visual Studio, as it uses XAML (eXtensible Application Markup Language) for UI layout and C# for logic. In the context of Sign Textify, this application was used to automate the process of taking photos for the dataset augmentation using the Kinect sensor as the camera. The interface prompts the user to choose the letter that s/he wants to augment, the number of images s/he wants to augment with, and the path where the images will be stored. The experiment consisted of augmenting letters A, N, T, U, and Z with 250 images each. After preprocessing, N was again the letter with the fewest samples: only 138 out of 250 images. This model successfully recognized 4 out of 5 augmented signs (A, T, U, and Z). However, this improvement came at the cost of other letters: it failed to recognize B, H, N, P, R, V, W, and X. This resulted in an overall decrease in performance (only 20/28 letters correctly predicted), suggesting that the updated model may has unintentionally biased it towards the augmented letters. The model continued to struggle with the letter N. It means that this letter remains the most challenging sign for the models to learn. The plotted training and testing F1 scores of the model trained on this augmented dataset are shown in Figure 15. *In the end*, the final model of Sign Textify achieved a test F1 score of 0.8095, meaning that 24 letters were correctly predicted. The four letters the model does not recognize are A, N, T, and U.

## USE CASES

This section outlines key use cases where the ASL alphabet learning application can be effectively applied across educational, professional, and accessibility-focused domains.

**Educational Settings -** Educators introducing ASL in elementary or middle school settings often begin with the alphabet. The application can support interactive and engaging activities that promote fingerspelling literacy and inclusivity at a foundational level.

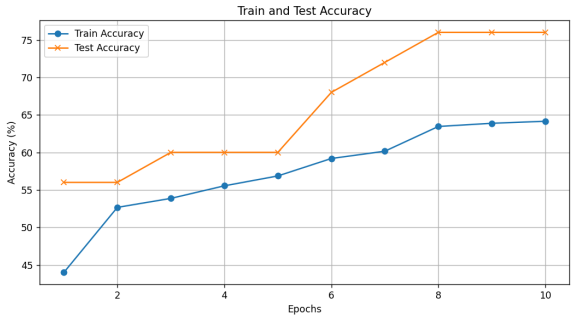


**Figure 15. Training and testing F1 scores for the model trained on the dataset with letters A, N, T, U, and Z augmented.**

**Independent Learning and Practice -** Family members of Deaf individuals can use the app to learn the ASL alphabet as a stepping stone toward basic interaction. Even limited knowledge of fingerspelling empowers loved ones to communicate names, objects, and personal expressions more meaningfully.

**Cognitive and Motor Skill Development -** For children and adults developing fine motor skills or learning hand-eye coordination, practicing the ASL alphabet offers both cognitive challenge and physical engagement. It's also used in therapy for individuals with developmental delays or motor disorders.

**Basic Accessibility Training -** Customer service representatives, retail staff, or healthcare providers can use the app for basic accessibility training: memorizing the alphabet allows them to fingerspell essential words like names, dates, or room numbers if needed.

**Language & Linguistic Studies -** Linguistics students or language enthusiasts interested in visual languages can use the ASL alphabet as an entry point into studying manual phonology and non-verbal communication systems.

**Gamified Learning Modules -** The ASL alphabet is ideal for gamification: apps can include quick matching games, gesture challenges, or accuracy-based scoring using camera input—perfect for short, motivational learning bursts.

## CONCLUSION

Research on previous attempts revealed a legitimate need for more interactive tools, both for learning and translating sign languages. This is why Sign Textify was designed to include practice by encouraging users to perform the signs themselves actively. To achieve this, several types of models have been tested, such as AdaBoost, ID3, and NN. However, neural networks proved to be the most appropriate model type for the task of sign recognition because of their highly customizable architecture. During model validation, two augmenting strategies were compared: increasing the number of samples to all letters equally and selectively augmenting only the misclassified letters. The results indicated that the targeted approach produced better performance than the indiscriminate augmentation. Tracking the hand was one of the most important steps in Sign Textify. Various tracking technologies were evaluated for this purpose, including Kinect's Skeleton Tracking, OpenCV, and Media Pipe. The Kinect SDK lacked the necessary precision, while OpenCV proved overly complex for the intended application. Ultimately, Media Pipe was the most suitable solution due to its balance between accuracy and usability.

## REFERENCES

1. Abdelkader, H.E. and El Said, W.K. 2019. An Assistive System based on Kinect Sensor to Help Students with Disabilities in Educational Institutions. Int. J. Eng. Res. Technol. 8, 4 (Apr. 2019), 606–610.
2. Boutsika, E. 2014. Kinect in Education: A Proposal for Children with Autism. Proc. Comput. Sci. 27 (2014), 123–129.
3. Candra, H., Yuniati, U., and Chai, R. 2024. The Application of Virtual Reality Using Kinect Sensor in Biomedical and Healthcare Environment: A Review. In Proc. 4th ICEBEHI 2023, Springer, Singapore, 15–38.
4. Hall, M.L., Hall, W.C., and Caselli, N.K. 2019. Deaf children need language, not (just) speech. First Language 39, 4 (2019), 367–395.
5. Heyko, D. 2021. Supporting d/Deaf and Hard of Hearing Employees in Their Workplaces Through Technology, Design, and Community. M.Sc. Thesis. University of Guelph.
6. Hsu, J. 2011. The Potential of Kinect in Education. Int. J. Inf. Educ. Technol. 1, 59 (2011), 1–6.
7. Laws, L.J. 2024. The Shared Experiences of Silenced Struggles of a Deaf and Hard-of-Hearing Community in North Carolina Seeking Mental Health Services. Liberty University, School of Behavioral Sciences.
8. Lingvano. 2025. Learn American Sign Language and start building bridges!
9. Lleras-Muney, A. 2002. The Relationship Between Education and Adult Mortality in the United States. National Bureau of Economic Research, paper no. 8986.
10. Lun, R. and Zhao, W. 2018. Kinect Applications in Healthcare. In Encyclopedia of Information Science and Technology, Fourth Edition, IGI Global.
11. Pocket Sign. 2025. ASL Sign Language Learning App.
12. Rocket Languages. 2025. Let's get you signing.
13. SignSchool. 2025. Learn American Sign Language.
14. Stevenson, J., Kreppner, J., Pimperton, H., Worsfold, S., and Kennedy, C. 2015. Emotional and behavioural difficulties in children and adolescents with hearing impairment: a systematic review and meta-analysis. Eur. Child Adolesc. Psychiatry 24 (2015), 477–496.
15. Zhang, Z. 2012. Microsoft Kinect Sensor and Its Effect. IEEE MultiMedia 19, 2 (2012), 4–10.