FPT Education

FPT UNIVERSITY

# Sign language recognition

by

**Nguyen Vinh Khang**
**Nguyen Nhat Hoang**
**Nguyen Dang Phuc Thinh**
**Dong Nguyen Gia Nguyen**

**THE FPT UNIVERSITY HO CHI MINH CITY**

# ABSTRACT

Sign language is a crucial communication medium for individuals who are deaf or mute. However, the challenge of translating sign language into text or speech remains a significant barrier. This research proposes a Vietnamese Sign Language registration model using BiGRU to analyze hand gestures from video input. We used Mediapipe to create a dataset containing 6489 npy files for 44 labels, each label's folder including 147 files, each file has 126 key points of 3 coordinates( x, y, z) to optimize model accuracy (except for the folder of letter H, which is special, there are 168 files). Experimental results show that the model achieves an accuracy of 97% on the test set. These results demonstrate the effectiveness of using BiGRU combined with Mediapipe for sign language recognition. The proposed model offers a lightweight yet accurate solution, making it suitable for real-time applications. Potential applications include assistive communication tools, educational platforms, and integration with AI-driven virtual assistants to bridge the communication gap for the deaf and mute community. Future work will focus on expanding the dataset, improving model robustness in diverse environments, and exploring deployment on mobile and embedded systems for wider accessibility.

# CONTENTS

# List of Figures



# List of Tables

# 1. INTRODUCTION

In recent years, the combination of deep learning and sign language recognition technology has made breakthroughs, helping to narrow the communication gap between the silent and the community. One of the notable products is the development of the Vietnamese Sign Language Recognition System (Vietnamese Sign Language - VSL), which allows the direct conversion of symbols into text. This technology not only opens up many opportunities in education and translation but also helps security personnel integrate more easily into society.

Knowledge of VSL Recognition and Solutions

- VSL is expected to face many large formulas, requiring a clear strategy to ensure accuracy and efficiency. One of the biggest problems is data. Vietnamese Sign Language is not yet widely available, leading to limited data training. To solve this problem, we have built a dedicated dataset from the Vietnamese sign language dictionary (https://tudienngonngukyhieu.com), ensuring accuracy and suitability for real silver community use.
- Technically, the recognition system needs to achieve high accuracy in the analysis and recognition of movements. We used the BiGRU model to train the data because this model can remember information better than other communication methods, especially effective in low-light conditions or complex camera angles. Experiments on real data show that BiGRU is superior in accuracy, helping to improve the performance of symbolic sign language recognition.
- In addition, Vietnamese have a rich system of signs, creating additional formulas in the work to ensure that the text reflects the exact content of the symbol. We have applied natural language processing algorithms to optimize the work of converting symbols to text most coherently and understandably.

Potential Applications

VSL recognition systems can be developed on many online translation platforms, supporting subsequent communication between farmers and ordinary people. This technology can also be integrated into supported virtualizations, enhancing the user experience of communicating information using language symbols.

Our research and testing not only solve the current formulas but also lay the foundation for new standards of accuracy and controllability of output in sign language recognition. The test results show that the system is capable of converting symbols to text accurately and reliably, opening up many applications in practice.

With the rapid development of artificial intelligence and deep learning, we believe that our recognition technology will become more and more perfect, contributing to building a harmonious and barrier-free society for people to feel secure.

# 2. RELATED WORK

1. **CNN vs YOLO**
    a. CNN excels in feature extraction and image classification but is slow and requires large datasets, making it unsuitable for real-time recognition.
    b. YOLO is fast, accurate, and detects objects in any position, making it ideal for real-time applications.
    c. In sign language recognition, YOLOv5 achieved 88.4% accuracy, outperforming CNN (52.98%).
2. **Sign Recognition Using SqueezeNet**
    a. SqueezeNet optimizes parameters by replacing 3x3 filters with 1x1 filters, reducing input channels, and utilizing the Fire Module.
    b. The combination of Conv1, Fire Modules, and Conv10 improves accuracy while maintaining efficiency.
3. **Applying MediaPipe for Hand Gesture Recognition**
    a. Uses BlazePalm for hand detection and the Hand Landmark model to identify 21 key 3D points.
    b. MediaPipe enables real-time gesture recognition with high accuracy, suitable for contactless interfaces.
4. **Sign Recognition Using ResNet & BiGRU with Bayesian Optimization**
    a. ResNet is used for feature extraction, while BiGRU is employed for hand gesture recognition.
    b. Bayesian Optimization fine-tunes parameters, achieving 99.75% accuracy on the ASL dataset.
5. **DeepSign: Sign Language Recognition with LSTM & GRU**
    a. Combines LSTM & GRU to process sign language videos in the ISL dataset.
    b. Feature extraction is performed using InceptionResNetV2, achieving 97% accuracy on IISL 2020.
    c. Limitations include low-quality videos and varying camera angles.

# 3. PROJECT MANAGEMENT PLAN

**Table 1.** Project Plan

| Task name | Priority | Owner | Start date | End date | Status | Issues |
|---|---|---|---|---|---|---|
| Find documents | High | Nguyen Nhat Hoang | 07/01/2025 | 9/01/2025 | completed | None |
| Review papers | High | Dong Nguyen Gia Nguyen | 10/01/2025 | 14/01/2025 | completed | None |
| Review and analyze public dataset | High | Nguyen Vinh Khang | 15/01/2025 | 18/01/2025 | completed | The lack of data |
| Create and label data | High | Nguyen Dang Phuc Thinh | 19/01/2025 | 31/01/2024 | completed | None |
| Select model | High | Dong Nguyen Gia Nguyen<br><br>Nguyen Dang Phuc Thinh | 01/02/2025 | 05/02/2025 | completed | Many possible models can be used to apply to the project so consumes a lot of time to choose the possible model |
| Compare results | Medium | Nguyen Dang Phuc Thinh | 06/02/2025 | 09/02/2025 | completed | Bi-GRU, LSTM and Bi-LSTM |
| Choose the model and | | Dong Nguyen Gia Nguyen | 10/02/2025 | 20/02/2025 | completed | Bi-GRU |

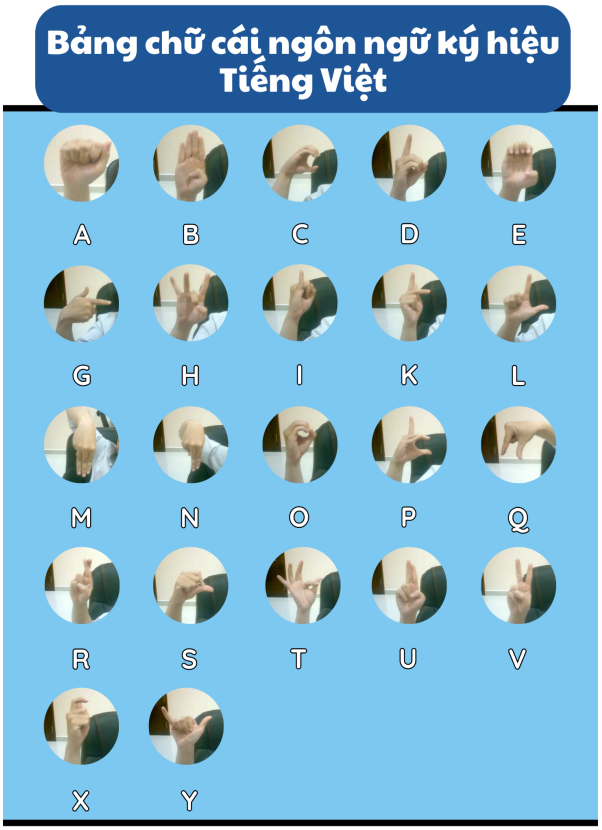
| | | | | | | |
|---|---|---|---|---|---|---|
| training model | | | | | | |
| Develop App | High | Nguyen Vinh Khang<br><br>Nguyen Nhat Hoang | 21/02/205 | 02/03/2025 | completed | Using Tkinter library to create an interface for users |
| Writing report | Medium | Nguyen Nhat Hoang<br><br>Nguyen Vinh Khang<br><br>Dong Nguyen Gia Nguyen<br><br>Nguyen Dang Phuc thinh | 01/03/2024 | 04/03/2024 | completed | None |

**Table 2.** Source code and data

| Items | Link | Description |
|---|---|---|
| Data | Link | 44 labels of VSL |
| Source code | Link | Code for model and app recognize reality |

# 4. MATERIALS AND METHODS

## 4.1 Materials




**Datasets:**

### 1. Introduction

Our Vietnamese Sign Language dataset contains 6489 npy files for 44 labels. Each label folder has 147 files except the "H" label, which has 168 files. A single file contains 126 key points for 3 coordinates (x, y, z) which use Mediapipe to detect landmarks in hand videos captured from a webcam.

**Figure 1:** Alphabet

### 2. Data Collection

Sign language data is collected using a webcam with the Mediapipe Hands library. The collection process includes:

- Capturing hand movements through a webcam.
- Identifying key landmarks of the hand.
- Normalizing data by using the wrist as the standard reference point.
- Saving data in NumPy array format (".npy").

The dataset includes letters, special characters, and some common gesture phrases:

A, B, C, D, Đ, E, G, H, I, K, L, M, N, O, P, Q, R, S, T, U, V, X, Y, /, ?, \, ~, ., del, space, ^, ', w, Xin chào, Cảm ơn, Tôi, Tên là, 22, Tuổi, gặp bạn, rất vui, sống ở, tạm biệt, Thành phố Hồ Chí Minh.

*Special label:

"/": slash (á, ý, ó, …)

"\": backslash (à, ò, è,...)

".": dot (ạ, ợ, ọ, ẹ,...)

“?”: question (ả, ỏ, ể,...)

“~”: tilde (ã, ễ, ỗ,...)

“`”: (ơ, ư)

“w”: (ă)

“^”: (â, ô, ê)

## 3. Data Preprocessing

After collection, the data undergoes the following processing steps:

- Normalization: Hand landmarks are adjusted based on the wrist position and scaled appropriately.
- Storage: Data is stored as NumPy array (".npy") files corresponding to each label.

## 4. Data Augmentation

Since the raw dataset is not sufficiently large, various augmentation techniques have been applied:

1. Adding noise (applying Gaussian noise to hand landmarks).
2. Translation (randomly shifting hand landmarks within the frame).
3. Scaling (modifying hand size ratio).
4. Rotation (random rotation within [-10, 10] degrees).
5. Flipping (mirroring the hand along the Y-axis).

**Each original sample is expanded approximately threefold through these techniques.**

## 5. Results & Evaluation

- Webcam-captured data maintains high quality due to the use of Mediapipe.
- Augmentation techniques help reduce bias during training.
- The dataset size has significantly increased after augmentation.
- Data is well-organized by label, ensuring ease of management and utilization.

## 6. Conclusion

The data collection and processing workflow has produced a comprehensive, diverse, and well-structured dataset for training the Vietnamese Sign Language recognition model. In the future, this dataset can be expanded further to enhance the model’s richness and accuracy.

# 4.2 Methods

## Data Processing Steps

### Data Loading and Feature Adjustment

The first step in data processing is loading the .npy files and ensuring they have a consistent number of features. Since some files contain only 63 features (one hand) while others have 126 features (both hands), a standardization process is applied:

If a file contains 63 features, an empty array of size 126 is created, and the existing values are copied into the first half of the array. The remaining features are filled with zeros to maintain consistency.

If a file contains more than 126 features, only the first 126 features are retained.

If a file contains less than 126 features (excluding the case of 63 features), it is considered invalid and excluded from the dataset.

This step ensures that every sample in the dataset has the same feature size of 126.

### Splitting Data into Training and Testing Sets

Once the data is standardized, it is divided into training and testing sets. The dataset is split in an 80:20 ratio, meaning 80% of the samples are used for training while the remaining 20% are reserved for testing.

Splitting the data ensures that the model is evaluated on unseen samples, helping assess its real-world performance.

### Encoding Labels

Since the class labels are stored as text (e.g., "A", "B", "C"), they need to be converted into numerical form for training the model. This is done in two steps:

Assigning numerical values to each label:

For example, the labels "A", "B", "C" might be assigned the values 0, 1, 2, respectively.

One-hot encoding the labels:

Instead of using a single numerical value, each label is converted into a vector representation.

For instance, if there are 44 different classes, the label 2 would be represented as a 44-dimensional vector with 1 at index 2 and 0 elsewhere.

This step is essential for training the model using categorical classification techniques.

### Standardizing Sequence Length

Since different samples contain different numbers of frames, all sequences must be adjusted to a fixed length of 30 frames. This ensures that the deep learning model receives inputs of uniform shape.

If a sequence has more than 30 frames, only the first 30 frames are retained.

If a sequence has less than 30 frames, additional frames filled with zeros are added at the end to reach 30 frames.

This process helps maintain consistency and prevents the model from facing difficulties due to varying sequence lengths.

# Model

This section provides the mathematical details behind the GRU, BiGRU, Dense layers, Loss function, and Optimization algorithm used in the model.

**Data Preprocessing**

## Masking

- **Purpose**:
  To handle padded inputs (with a value of 0) and prevent them from affecting the learning process.

$$\text{mask}(x_t) = \begin{cases} \text{True}, & \text{if } x_t = \mathbf{0} \\ \text{False}, & \text{if } x_t \neq \mathbf{0} \end{cases}$$

- **Formula**:
  For each timestep xtx_txt in the sequence:

**Sequence Processing with Bidirectional GRU**

The model uses two Bidirectional GRU layers to capture information from both past and future contexts in the sequence.

**First Bidirectional GRU Layer (256 Units)**

- **Basic GRU Equations**:
  For each timestep t, a GRU cell computes:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) && \text{(update gate)} \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) && \text{(reset gate)} \\ \tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) && \text{(candidate hidden state)} \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t && \text{(updated hidden state)} \end{aligned}$$

**Bidirectional Operation**:
Two GRUs operate in opposite directions:

$$\overrightarrow{h}_t = \text{GRU}_{\text{forward}}(x_t, \overrightarrow{h}_{t-1})$$
$$\overleftarrow{h}_t = \text{GRU}_{\text{backward}}(x_t, \overleftarrow{h}_{t+1})$$

Their outputs are concatenated:

$$h_t = \text{concat}(\overrightarrow{h}_t, \overleftarrow{h}_t)$$

With `return_sequences=True`, the output at every timestep is kept.

**Batch Normalization**:
Applied to the outputs to stabilize and accelerate training:

$$\hat{h}_t = \gamma \frac{h_t - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

### 2.2. Second Bidirectional GRU Layer (128 Units)

- **Purpose**:
  To summarize the sequence into a single feature vector.
- **Process**:
  This layer works similarly but returns only the final hidden states:

$$\overrightarrow{h}_{\text{final}} = \text{GRU}_{\text{forward}}(\{h_t\}_{t=1}^{T})$$
$$\overleftarrow{h}_{\text{final}} = \text{GRU}_{\text{backward}}(\{h_t\}_{t=T}^{1})$$

They are concatenated to form:

$$h_{\text{final}} = \text{concat}(\overrightarrow{h}_{\text{final}}, \overleftarrow{h}_{\text{final}})$$

**Batch Normalization** is applied afterward for stabilization.

## Fully Connected (Dense) Layers

After the GRU layers extract sequence features, fully connected layers with GELU activation further process these features for classification.

### Dense Layers with GELU Activation

- **Dense Layer Formula**:
  For each dense layer, the output is computed as:

$$z = \text{activation}(Wx + b)$$

**GELU Activation Function**:

An approximate form of GELU is given by:

$$\text{GELU}(x) = 0.5\,x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right)$$

This function allows a smooth flow of both positive and negative values, which helps in avoiding issues like "dead neurons" common in ReLU.

**Additional Layers:**

Batch Normalization is applied after each Dense layer to standardize outputs.

Dropout is used to prevent overfitting by randomly disabling neurons during training.

L2 Regularization penalizes large weights to further reduce overfitting.

### Output Layer

- **Output**:
  The final Dense layer outputs a vector of logits corresponding to the number of classes.

- **Softmax Activation**:
  The softmax function converts these logits into probabilities:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \quad i = 1, 2, \ldots, C$$

# Loss Function and Optimization

## Loss Function: Categorical Crossentropy

- **Formula**:
  This loss function measures the difference between the predicted probability distribution and the true distribution:

$$\mathcal{L} = -\sum_{i=1}^{C} y_{\text{true},i} \log(y_{\text{pred},i})$$

## Optimizer: Adam

- **Adam Update Equations**:
  Adam combines momentum and adaptive learning rates. Its updates are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

# Output

The hand sign language recognition system is an application that combines Tkinter, OpenCV, and MediaPipe to detect and display hand signs in real-time. The system offers multiple features to assist users, including text conversion from hand signs, hand landmark visualization, a sign language alphabet chart, and instructional videos, making it easier for users to learn and practice sign language.

**1. Real-Time Hand Sign Recognition & Video-Based Sign Detection**

- The system uses a live camera feed to detect and track hand movements.
- Supports sign recognition from pre-recorded videos.
- Identifies hand landmarks to analyze the shape and position of fingers.
- Displays the recognized hand sign directly on the interface, allowing users to follow along easily.

**2. Text Display & Simulated Voice Output**

- Converts hand signs into letters, words, or complete phrases.
- Includes a simulated voice reading feature for recognized text.
- Helps users construct coherent sentences using sign language.

**3. Hand Landmark Visualization**

- The system overlays hand landmarks directly onto the video feed to show finger positions.

- Helps users observe and adjust hand movements to improve recognition accuracy.

**4. Sign Language Alphabet & Video Tutorials**

- Displays a visual sign language alphabet chart, allowing beginners to learn quickly.
- Provides video tutorials demonstrating common words and phrases in sign language.
- Helps users practice and enhance their communication skills effectively.

**5. Space, Punctuation & Character Deletion Handling**

- Supports sign-based text input, including:
  - Adding spaces between words.
  - Deleting characters in case of mistakes.
  - Saving and clearing text for re-entry when needed.
- Enhances the user experience and makes input more flexible.

**6. Word Merging & Sentence Optimization**

- Prevents word repetition during continuous recognition.
- Avoids character jumping issues when detecting phrases or sign language sequences.

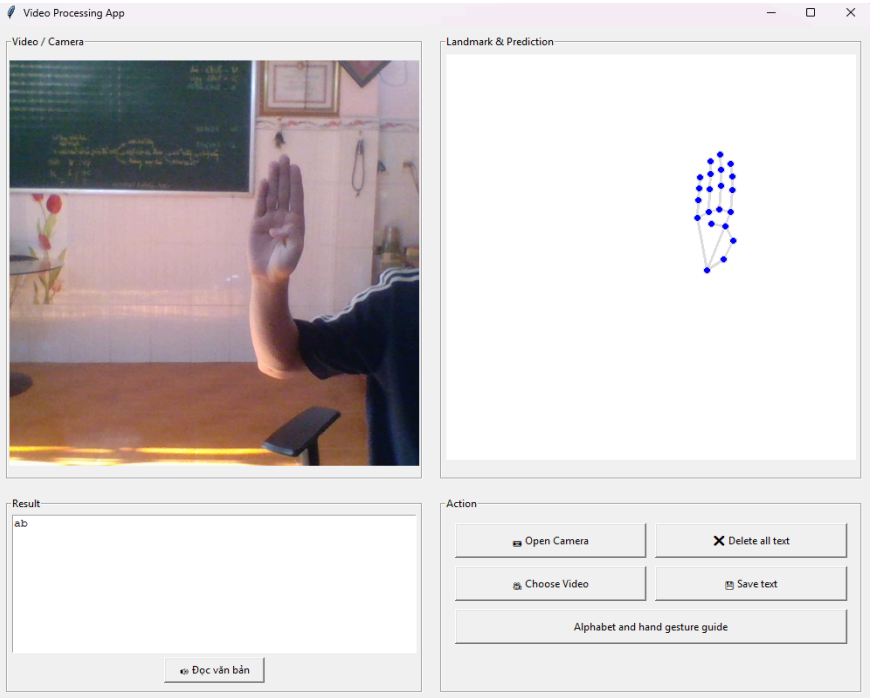



Figure 2: Project app

# 5. RESULTS

### Dataset and Experimental Setup

The predictive model was developed using a dataset of 6,489 npy files. Each file contains 126 key points (x, y, z coordinates) extracted from hand gestures via MediaPipe. The model architecture is based on a Bidirectional Gated Recurrent Unit (BiGRU) network, designed to capture the temporal dynamics of sign language gestures.

### Performance Metrics

The model was evaluated on a dedicated test set, and the following metrics were recorded:

- **Accuracy:** 98%

### Compare to results

| | **Accuracy** | **Loss** |
|---|---|---|
| **BiGRU** | **around 98%** | **0.26** |
| **BiLSTM** | **around 95%** | **0.53** |
| **GRU** | **around 93%** | **0,68** |
| **LSTM** | **around 80%** | **1.11** |

**Table 3 Compared results**

### Summary

The experimental results confirm that our predictive model for hand sign language recognition is highly effective, achieving a 98% accuracy on the test set. The model's strong performance, even under varying real-world conditions, highlights its potential for integration into real-time assistive communication tools for the deaf and mute community.

# 6. DISCUSSION

Authors should discuss the results and how they can be interpreted from the perspective of previous studies and of the working hypotheses. The findings and their implications should be discussed in the broadest context possible. Future research directions may also be highlighted.

# 7. CONCLUSIONS AND PERSPECTIVES

**Conclusions:**
This study proposed a Vietnamese Sign Language recognition model using BiGRU combined with Mediapipe to analyze hand gestures from video input. By leveraging a dataset of 6489 npy files for 44 labels. Our model achieved a quite good accuracy of 97% on the test set. The results demonstrate the effectiveness of our approach in Vietnamese sign language recognition, offering a lightweight yet highly accurate solution suitable for real-time applications. This study contributes to the development of assistive technologies aimed at enhancing communication for individuals who are deaf or mute.

**Perspectives**
Although our model achieves high accuracy, several challenges remain that open avenues for future research. First, expanding the dataset with more labels about many different topics. Second, real-world deployment requires optimization for mobile and embedded systems to ensure low-latency performance. Additionally, integrating Natural Language Processing (NLP) models to enhance context understanding and improve translation accuracy could further enhance the system's usability. Future work will also focus on incorporating facial expressions and body movements, which play a crucial role in sign language communication. Ultimately, this research lays the foundation for developing AI-powered assistive tools that can bridge the communication gap and promote inclusivity for the deaf and mute community.

# 8. REFERENCES

[1] ISanket Bankar, Tushar Kadam, Vedant Korhale, Mrs. A. A. Kulkarni,"Real Time Sign Language Recognition Using Deep Learning",04 Apr 2022, IRJET-V9I4167.pdf

[2] Võ Đức Hoàng, "NHẬN DIỆN BẢNG CHỮ CÁI NGÔN NGỮ KÝ HIỆU TIẾNG VIỆT SỬ DỤNG MÔ HÌNH HỌC SÂU",28-29/9/2023, 74BB_FAIR2023_paper_238.pdf

[3] Indriani Moh.Harris Ali Suryaperdana Agoes, " Applying Hand Gesture Recognition for User Guide Application Using MediaPipe", 2021, (PDF) Applying Hand Gesture Recognition for User Guide Application Using MediaPipe

[4] Fadwa Alrowais, Radwa Marzouk, Fahd N. Al-Wesabi, and Anwer Mustafa Hila, "Hand Gesture Recognition for Disabled People Using Bayesian Optimization with Transfer Learning", 2023, (PDF) Hand Gesture Recognition for Disabled People Using Bayesian Optimization with Transfer Learning.

[5] Deep Kothadiya, Chintan Bhatt, Krenil Sapariya, Kevin Patel, Ana-Belén Gil-González and Juan M.Corchado, "Deepsign: Sign Language Detection and Recognition Using Deep Learning", 3 June 2022, Deepsign: Sign Language Detection and Recognition Using Deep Learning.