American Sign Language (ASL) Classification

Submitted in partial fulfillment of the requirements of the course of

Image Analysis and Computer Vision

by

Rushi Kantode 60003200117
Yash Bijoor 60003200136

under guidance of

Assistant Professor
Priyanca Gonsalves

Department of Information TechnologyDwarkadas J. Sanghvi College of Engineering,Mumbai– 400 056

2022-23

Abstract

This study evaluates the potential of computer vision to transform sign language gestures into text. For those who are deaf or hearing-impaired, sign language is a vital tool of communication. Yet, in scenarios that require written or verbal forms of communication, it can prove to be a hurdle. To this end, the proposed solution puts to use computer vision algorithms to recognize and decode sign language movements, followed by real-time transformation of this information into written text. Aimed to be available, fast and precise, the system is tried and tested by users with hearing loss. Results of the trials emphasize the effectiveness of computer vision technology in facilitating communication for those whose primary language is sign language. Lastly, the paper deliberates on further areas of exploration, such as its integration into education, healthcare and other environments where communication is vital.

This study investigated a system using Random Forest machine learning algorithm and a CNN model, to recognize sign language gestures. After the dataset of sign language videos was pre-processed to identify pertinent features, they were fed into the model, allowing it to recognize different gestures. Various metrics, including accuracy, speed, and user experience, were employed to evaluate the system, with high accuracy and the capacity for real-time text output found. Additionally, the system is highly user-friendly and needs minimal instruction, making it an excellent choice for a variety of people, especially the deaf and hard of hearing.

Table of Contents

Introduction

Motivation:

The primary impetus for this endeavor stems from the fact that sign language is a major method of communication for millions of deaf and hard of hearing individuals around the globe. However, it can be tricky for deaf and hearing individuals to communicate in situations where written or spoken language is needed, resulting in social seclusion, restricted access to education and work, and poorer quality of life.

Consequently, investigators have investigated various technologies to promote communication between deaf and hearing people, including sign language recognition systems. These systems implement computer vision algorithms to spot and interpret sign language movements, which are then translated into either written or spoken output. Though numerous advancements have been made in this area, developing accurate, competent, and user-friendly sign language recognition systems is still an ongoing area of exploration.

This particular project focuses on building a sign language recognition system that can quickly and accurately interpret sign language movements into written language in real-time. This system is intended to help improve communication and availability for deaf and hard of hearing individuals, specifically in fields that require written language, such as education, healthcare, and employment. By providing a dependable and easy-to-use communication tool, this system has the potential to enhance the lives of deaf and hard of hearing individuals, and bring about more integration in society.

Objective:

The goal of this venture is to construct a framework that can precisely and proficiently decipher sign language gestures into written text through computer vision innovation. To this end, the project intends to formulate a sign language acknowledgment framework that can correctly distinguish and translate sign language gestures with the aid of machine learning techniques such as Random Forest or CNN and fabricate a system that can produce written text output immediately, thus permitting efficient communication between deaf and hearing people.

It should also have user-friendly interface that enables effortless and instinctive use of the system by deaf and hard of hearing individuals.

Literature review

Journal/ Paper No.	Model used	Dataset used	Conclusion
1.	Convolution Neural Network based VGG16 architecture is used as well as a Tensorflow model for image classification (improved the accuracy of the latter by over 4%)	The dataset which has been used is the ASL dataset which has over 87000 images and has been used to train and test the video.	There has been an improvement in accuracy from 94% of CNN to 98.7% by Transfer Learning.
2.	The algorithm is developed on top of a Java-based OpenCV wrapper. SVM is used to train our model and this experimentation deals with using three different array parameters for SVM and comparing the results of each. The three array parameters are Detection Method, Kernel and Dimensionality reduction type. The following are the different types of array parameters that are used for training. • Detection Method - Contour Mask, Canny Edges, Skeleton • Kernel - Linear, Radial Basis Function (RBF) • Dimensionality Reduction Type - None, Principal Component Analysis (PCA)	The dataset used for this paper is the ASL Kaggle dataset [2], which contains 3000 images for every alphabet of the English vocabulary.	This paper compares different techniques and chooses the most optimal approach for creating a vision- based application for sign language to text/speech conversion. The proposed system could efficiently recognize the alphabets from images using a customized SVM model. This project is aimed at societal contribution.

3.	The algorithm was written in C++ utilizing the computer vision libraries of OpenCV 2.3. The algorithm consists of the following steps 1. **Frame Capture:** Resizing and conversion to YCrCb. 2. **Skin Detection:** Discrete color segmentation and morphological operations. 3. **Segment Hand:** Contour detection and segmentation hand from frame. 4. **Feature Selection:** Histogram normalization and canny edge detection. 5. **Normalize Image:** Reduction to 50 x 50. 6. **Dimensionality Reduction:** Locality preserving projections. 7. **Classification:** Support vector machine.	A total of 32 different ASL signs were trained, with 26 consisting of the ASL alphabet and 6 other custom created commands. A database of 80 to 100 images of each sign were used for training. The hand was rotated variably within a boundary of 15 degrees on all axes from the most frontal representation of the sign	This work has explored the differences between a local and cloud assisted method for automatic American Sign Language detection on a mobile device. A efficient implementation accomplished ASL detection while keeping power constrain low, with generally fast computation and high accuracy.

4.	Algorithm-1: Creating & Splitting the Dataset Input: Created a dataset, which contains images for each sign Output: train and test data 1: Traverse through the directory specified and do 2: if the “train” directory exists 3: delete the directory and contents in it 4: if the “test” directory exits 5: delete the directory and contents in it 6: Create “train” and “test” directory if they don’t exist 7: Traverse through the specified dataset directory where all the images created for each sign reside 8: Traverse through every sub-directory in the dataset directory(there are 29 sub-directories, each containing the images for their corresponding signs) 9: Initialize num to 0.8% of the number of images in the current sub-directory and initialize i to 0 10: if i < num do 11: move the image dataset directory to the “train” directory 12: else do 13: move the image dataset directory to the “test” directory 14: increment i 15: Display the details of number of images in test and train directory Algorithm-2: Training on the split data Input: The train and test data obtained from module 1 Output: Obtaining the trained model i.e. obtaining *.h5* file 1: import required libraries 2: Access Sequential model by *tf.keras.Sequential()*[10] 3: Add the “Conv2D” layers to the convolutional neural network with increasing number of units in each layer, using *tf.keras.layers.Conv2D()* [20] 4: Add a final “Dense” layer with 29 units using *tf.keras.layers.Dense(29, ...)* [16] 5: Compile the model with appropriate metics. 6: Use ImageDataGenerator using *tf.keras.preprocessing.image.ImageDataGenerator()* for training data to add additional images by making zoom_range as 0.2%, shear_range as 0.2%, making horizontal_flip as True and rescaling by 1./255 the existing images [13] 7: Use ImageDataGenerator in keras.preprocessing library for test images by rescaling the existing images by 1./255 [13] 8: Train the model using. 8: Evaluate the model using test data and obtain the accuracy and value_loss of the model using *evaluate()* method for the test data [17]. 9: save the model Algorithm-3: Testing on the trained model	Since none of the datasets on the huge internet are precise or meet our needs, we have built our own dataset using Teachable machine software with 200– 230 photos in .jpg format for each sign that corresponds to each letter of the English alphabet. Here we have created a total of 6414 images in .jpg format and then split them into training data and testing data.	Future improvements will include making it such that it may be used in any context and that the user's surroundings will not affect the outcome of a prediction. The ability to anticipate in all languages might be added. It does not follow that a prediction will always be accurate even if the model has high accuracy

Input: sample images for each sign to test
Output: Observing the behavior of model
1: Load the saved model
2: Convert the keras *.h5* model to *.tflite* model using *tf.lite.TFLiteConverter.from_keras_model()*
3: Traverse through images in the sample images directory
4: Load the image
5: resize the image according to model into (64,64,3) using *resize()* method in cv2 library [21]
6: predict the image and display predicted alphabet
7: Import cv2
8: Use video capture for capturing the frames
9: while True do
10: Capture the frame
11: Resize the image using *resize()* method in cv2 library [21]
12: Predict the image using *predict()* method keras Library and obtain the predicted text.
13: Append the predicted text to the "sequence" string
13: Display the sequence
14: Close the application

Algorithm-4: Developing the web application

Input: trained model, the labels and *index.html*
Output: web application
1: import the flask library
2: access the Flask API using *Flask()* method
3: render the template to the file *index.html* with the help of render_template()[11]
4: Switch on the camera using video capture object [22]
5: While True do
6: get the frame and the predicted sign from *camera.py* (*Camera.py* will capture the frame, predict it and Return the predicted sign and the frame to be displayed on the webpage)
7: Append the sign returned from *camera.py* to sequence string
8: Pass the frame and the sequence to the webpage and display them on the screen
9: Exit when closed the application

<table>
<tr>
<td>5.</td>
<td>The process of identifying Sign Language alphabets is distributed as pre-processing the input image, applying a CNN model to it, recognizing it in the form of text and then converting it text to speech. A reverse operation of speech to sign is also performed.
Model: "sequential"
<table>
<tr><th>Layer (type)</th><th>Output Shape</th><th>Param #</th></tr>
<tr><td>conv2d (Conv2D)</td><td>(None, 62, 62, 32)</td><td>896</td></tr>
<tr><td>max_pooling2d (MaxPooling2D)</td><td>(None, 31, 31, 32)</td><td>0</td></tr>
<tr><td>conv2d_1 (Conv2D)</td><td>(None, 29, 29, 32)</td><td>9248</td></tr>
<tr><td>max_pooling2d_1 (MaxPooling2</td><td>(None, 14, 14, 32)</td><td>0</td></tr>
<tr><td>conv2d_2 (Conv2D)</td><td>(None, 12, 12, 64)</td><td>18496</td></tr>
<tr><td>max_pooling2d_2 (MaxPooling2</td><td>(None, 6, 6, 64)</td><td>0</td></tr>
<tr><td>flatten (Flatten)</td><td>(None, 2304)</td><td>0</td></tr>
<tr><td>dense (Dense)</td><td>(None, 256)</td><td>590080</td></tr>
<tr><td>dropout (Dropout)</td><td>(None, 256)</td><td>0</td></tr>
<tr><td>dense_1 (Dense)</td><td>(None, 53)</td><td>13621</td></tr>
</table>
Total params: 632,341
Trainable params: 632,341
Non-trainable params: 0
The parameter of the convolutional layer is given by,
*Parameter = ((K)*S+1)*F*
…where, K is the kernel size, S is Stride and F is Filters
Parameter for pooling layer is zero. Parameter of fully connected layer is given by,
*Parameter = (previouslayer+1)*currentlayer*</td>
<td>The data is taken with different backgrounds, different light settings, and at different times (day and night), in order to ensure that the accuracy of the classification model is maximized. The data is divided in different section for ASL and ISL and these two data directories are further divided in Training and Test data sub- directories. This makes the data easily usable by the CNN model for classification. The total number of images in training dataset were 62967 and the total number of images in validation dataset were 6997</td>
<td>Three CNN models were created for recognizing the alphabets of ASL and ISL. The three layered model gave a validation accuracy of 98.34% with 50 epochs. A real time sign language to speech conversion and speech to sign language conversion was achieved using the project work so that two people can communicate without any difficulty if any one of them knows only sign language and the other one has no knowledge of the sign language. Also, a sign language reading mechanism was developed to read English text in sign language by just capturing an image of the text.</td>
</tr>
</table>

<table>
<tr>
<td>6.</td>
<td>The techniques of image segmentation and feature detection played a crucial role in implementing this system. We formulate the interaction between image segmentation and object recognition in the framework of FAST and SURF algorithms. The system goes through various phases such as data capturing using KINECT sensor, image segmentation, feature detection and extraction from ROI, supervised and unsupervised classification of images with K-Nearest Neighbour (KNN)-algorithms and text-to-speech (TTS) conversion. The combination FAST and SURF with a KNN of 10 also showed that unsupervised learning classification could determine the best matched feature from the existing database. In turn, the best match was converted to text as well as speech.

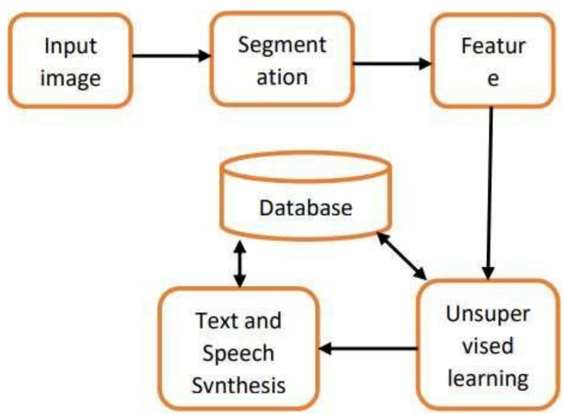



Fig 1: System Overview</td>
<td>Unsure … 1200 – 6000 training sample dataset</td>
<td>The introduced system achieved a 78% accuracy of unsupervised feature learning. The success of this work can be attributed to the effective classification that has improved the unsupervised feature learning of different images. The pre-determination of the ROI of each image using SURF and FAST, has demonstrated the ability of the proposed algorithm to limit image modelling to relevant region within the image.</td>
</tr>
</table>

Proposed Methodology/Approach

Problem Definition

This project endeavors to bridge the communication gap between deaf and hard of hearing individuals and the hearing population, particularly in contexts that require the use of written or spoken language. As sign language is the main mode of communication for these individuals, people who don't understand sign language find it hard to interact with them. Thus, the system we intend to build will leverage computer vision technology to recognize and interpret sign language gestures and transform them into written texts in real-time. By offering a simple yet effective way of communication, our system aims to increase inclusivity and better the quality of life for deaf and hard of hearing individuals. We ultimately hope to facilitate the participation of such individuals in daily life scenarios such as education, healthcare, and employment.

Scope

The scope of this project is to develop a sign language recognition system that can accurately and efficiently translate sign language gestures into written text using computer vision technology. The system will be designed to recognize and interpret a wide range of sign language gestures, allowing for effective communication in various domains.

The system will be developed using machine learning algorithms such as Random Forest or CNN, and will be trained on a large dataset of sign language videos. The system will be designed to generate written text output in real-time, allowing for efficient communication between deaf and hearing individuals.

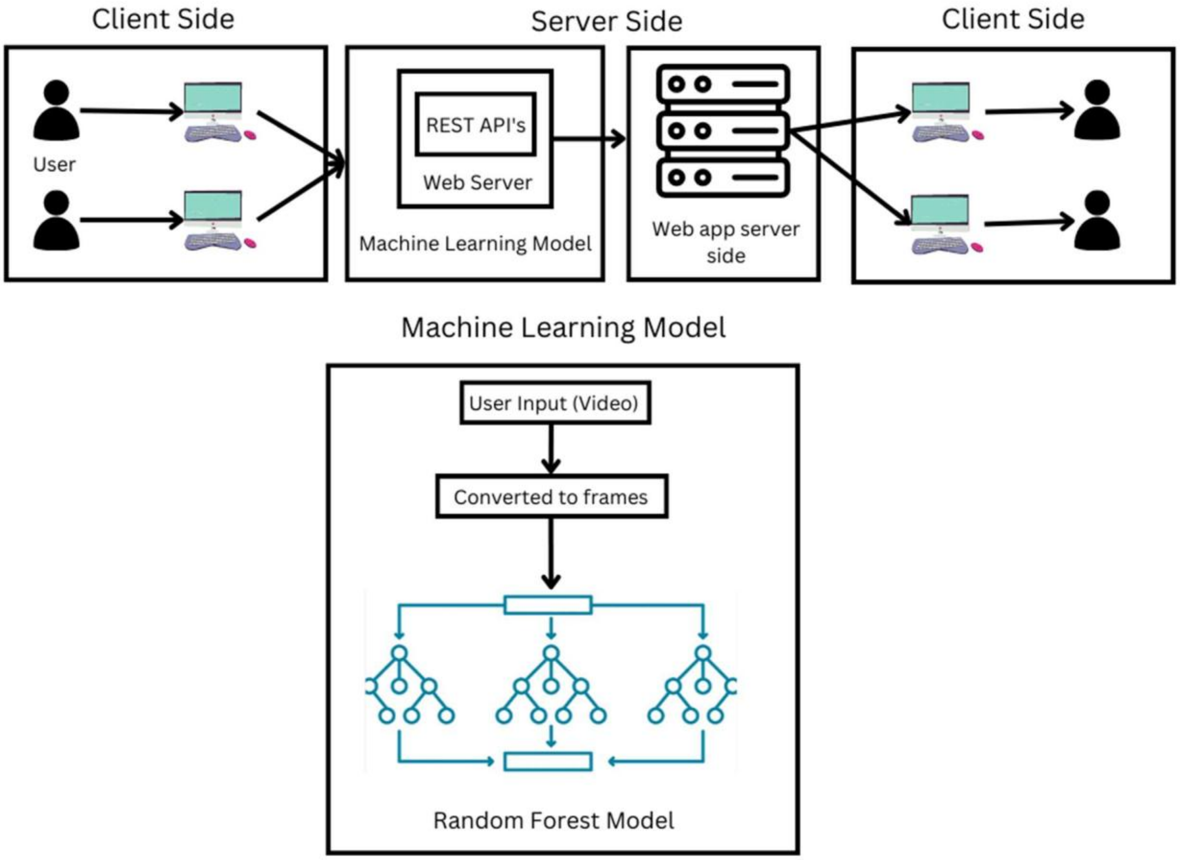
Proposed Approach

Here's a proposed approach for implementing a Random Forest model for American Sign Language (ASL) recognition using video:

- Data collection: Collect a large dataset of ASL videos showing different people signing the alphabet letters. The videos should be recorded from different angles, with varying lighting conditions, and with different backgrounds. Ensure that the dataset is balanced and contains an equal number of samples for each alphabet letter.
- Data preprocessing: Preprocess the videos to extract relevant features that can be used by the Random Forest model. For example, you can extract hand and finger position, motion, and trajectory information from the videos. Additionally, you can use techniques such as background subtraction and skin detection to isolate the hands from the background.
- Feature engineering: Perform feature engineering to transform the raw video features into a format that can be used by the Random Forest model. This may involve techniques such as dimensionality reduction, feature selection, and normalization.
- Training the Random Forest model: Train the Random Forest model using the preprocessed and engineered features. Use techniques such as cross-validation and hyperparameter tuning to optimize the performance of the model.

- Testing and evaluation: Test the performance of the model on a separate testing dataset. Evaluate the model's accuracy, precision, recall, and F1 score. Use techniques such as confusion matrix and ROC curve analysis to gain insights into the model's performance.
- Deployment: Deploy the Random Forest model as an application that can be used to recognize ASL alphabets from live video feeds or pre-recorded videos.
- Continuous improvement: Monitor the performance of the model and continuously improve it by collecting new data, refining the preprocessing and feature engineering techniques, and tuning the model's hyperparameters.
- In summary, the proposed approach involves collecting a dataset of ASL videos, preprocessing the videos to extract relevant features, engineering the features to be used by the Random Forest model, training the model, testing and evaluating the model, deploying the model, and continuously improving the model.

System Design



Implementation

About the Dataset:

The dataset was taken from Kaggle[7]. The data set is a collection of images of alphabets from the American Sign Language, separated in 29 folders which represent the various classes. The training data set contains 87,000 images which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z and 3 classes for *SPACE*, *DELETE* and *NOTHING*.
These 3 classes are very helpful in real-time applications, and classification.
The test data set contains a mere 29 images, to encourage the use of real-world test images.

Asl.py:

```
Asl.py ×
Asl.py > ...
import numpy as np
import cv2
import mediapipe as mp
import joblib
import scipy.stats as st
import warnings
warnings.filterwarnings("ignore")
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.corpus import wordnet
from nltk.tokenize import word_tokenize

print(joblib.__version__)

# Define the ASL labels
asl_labels = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X'

# Initialize variables for sentence
sentence = []
last_sent = None

# Load trained classifier
clf = joblib.load('rf_model.joblib')

# Initialize hand tracking model
hands = mp.solutions.hands.Hands(
    static_image_mode=False,
    max_num_hands=1,
```

```python
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5)
mpDraw = mp.solutions.drawing_utils
# Initialize video capture device
cap = cv2.VideoCapture(0)
prediction = []

while True:
    # Read a frame from the video capture device
    ret, frame = cap.read()
    frame = cv2.flip(frame, 1)

    if not ret:
        print('Unable to read frame from video capture device')
        break

    # Convert the frame to RGB
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Use the hand tracking model to detect hand landmarks
    results = hands.process(image)

    # Extract the hand landmarks
    if results.multi_hand_landmarks:
        landmarks = results.multi_hand_landmarks[0].landmark

        # Convert the hand landmarks to a flattened NumPy array
        landmarks_arr = np.array([(lmk.x, lmk.y, lmk.z) for lmk in landmarks]).flatten()

        # Classify the ASL sign using your trained classifier
        asl_class = clf.predict([landmarks_arr])[0]
        prediction.append(asl_class)
        mode_class = st.mode(prediction[-9:])[0][0]
        asl_label = asl_labels[mode_class]

        if asl_label != last_sent:
            if asl_label == 'space':
                sentence.append(' ')
                last_sent = asl_label
            elif asl_label == 'del':
                if sentence:
                    sentence.pop()
            else:
                sentence.append(asl_label)
                last_sent = asl_label

        # Concatenate the words in the sentence
        sentence_str = ''.join(sentence)

        # Correct the sentence
        words = word_tokenize(sentence_str)
        corrected_words = []
        for word in words:
            syns = wordnet.synsets(word)
            if syns:
                corrected_words.append(syns[0].lemmas()[0].name().replace('_', ' '))
            else:
                corrected_words.append(word)
```

```python
        corrected_sentence = ' '.join(corrected_words)

        # Draw the ASL label and corrected sentence on the frame
    try:
        cv2.putText(frame, asl_label, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 0), 4)
        cv2.putText(frame, sentence_str, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
    except:
        pass

    # Show the frame
    cv2.imshow('ASL Classification', frame)

    # Check for key press to exit
    k = cv2.waitKey(20) & 0xff
    if k ==27:
        break

# Release the video capture device and close
cap.release()
cv2.destroyAllWindows()


print(sentence_str)
```

Interface Design:

Proposed interface designs which could be ideal for our project are-

- User input: The interface should have a feature that allows users to input the video footage of a person signing an alphabet letter. This can be in the form of a video recording from a camera or uploaded video footage.
- Processing feedback: The interface should provide feedback to the user on the processing of the input video. For example, there can be a progress bar or a loading animation to show that the system is processing the input.
- Results display: Once the processing is complete, the interface should display the recognized alphabet letter to the user. This can be in the form of a text or image output, such as a picture of the recognized letter.
- User interface design: The interface should have an aesthetically pleasing design with a clear and easy-to-read font. It should also have a simple color scheme that is easy on the eyes. The buttons and icons should be large enough to be easily clickable or tappable.

Conclusion

In conclusion, a sign language recognition system which accurately and quickly interprets sign language hand movements into written text utilizing computer vision technology can greatly advance communication and access for those with hearing and speech impairments. By offering a dependable and simple-to-use communication instrument, this system has the potential to enhance inclusiveness in society and enhance the lifestyle of individuals who depend on sign language as their main type of communication. With the use of AI algorithms like Random Forest and CNN and a comprehensive database of sign language recordings, the system can identify and comprehend an expansive range of sign language gestures, providing successful communication across many areas. In addition, a user-friendly interface that is accessible and adjustable to various user requirements is also important to guarantee that the system meets the necessities of its designated users. To determine if the system is adequate, the evaluation will take into account various metrics, such as accuracy, speed, and ease of use, guaranteeing that it fulfills or surpasses industry standards. The scope of the project is focused on developing a sign language to text system, however the prospective consequences of such a system cannot be downplayed.

References

1. Thakar, Shubham et al. "Sign Language to Text Conversion in Real Time using Transfer Learning." *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)* (2022): 1-5.
2. K. Tiku, J. Maloo, A. Ramesh and I. R., "Real-time Conversion of Sign Language to Text and Speech," *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2020, pp. 346-351, doi: 10.1109/ICIRCA48905.2020.9182877.
3. P. Hays, R. Ptucha and R. Melton, "Mobile device to cloud co-processing of ASL finger spelling to text conversion," 2013 IEEE Western New York Image Processing Workshop (WNYIPW), Rochester, NY, USA, 2013, pp. 39-43, doi: 10.1109/WNYIPW.2013.6890987.
4. H. S. S. K. Vezzu, S. Nalluri, K. Swetha and V. SaiKumar, "Hand Sign Recognition using Image Processing," 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2023, pp. 288-293, doi: 10.1109/IDCIoT56793.2023.10053506.
5. A. Rustagi, Shaina and N. Singh, "American and Indian Sign Language Translation using Convolutional Neural Networks," 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2021, pp. 646-651, doi: 10.1109/SPIN52536.2021.9566105.
6. Akano, Victoria & Olamiti, Adejoke. (2018). Conversion of Sign Language To Text And Speech Using Machine Learning Techniques. JOURNAL OF RESEARCH AND REVIEW IN SCIENCE. 5. 58-65. 10.36108/jrrslasu/8102/50(0170).
7. Kaggle dataset: https://www.kaggle.com/datasets/grassknoted/asl-alphabet