International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04756
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

SIGN LANGUAGE DETECTION USING CNN MODEL

MUPPALA NAGA KEERTHI, PRATHAPARAO KALAHYNDAVI

Assistant Professor, 2MCA Final Semester,

Master of Computer Applications,

Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India

Abstract

A wide range of fields, including computer vision, computer graphics, natural language processing, human-computer interaction, linguistics, and Deaf culture, are required to develop successful sign language recognition, generation, and translation systems. The speech and hearing- impaired community use sign language as a medium of their communication. Most people who aren't familiar with sign language find it difficult to communicate without an interpreter. Sign language recognition appertains to track and recognize the meaningful emotion of humanmade with head, arms, hands, fingers, etc. The technique that has been implemented here, describes the gestures from sign language to a spoken language which is easily understood by the listening. The gestures that have been translated include alphabets, words from static images. This becomes more important for the people who completely rely on gestural sign language for communication tries to communicate with a person who does not understand the sign language. Most of the systems that are under use face a recognition problem with the skin tone; by introducing a filter it will identify the symbols irrespective of the skin tone. The aim is to represent features that will be learned by a system known as convolutional neural networks (CNN), which contains four types oflayers: convolution layers, pooling/subsampling layers, nonlinear layers, and fully connected layers.

IndexTerms: Convoutional Neural Networks(cnn), Hand Gesture Recognition, image Segmentaion, OpenCV,

Region of Interest, Real Time Prediction, Feature Extraction, Deep Learning.

1.INTRODUCTION:

Communication connects people by allowing them to convey messages to each other, to express their innerfeelings, andto exchange thoughts, either verbally or non-verbally. Unfortunately, there is a communication gap for the minority of people who are deaf or hard of hearing. Because they can't communicate vocally, they rely on signlanguage. Hearing and speech disabled persons communicate via sign language, which is a natural language. Instead of utilising words, sign language uses hand gestures and movements, body language, and facial expressions to communicate. Sign language is a way of communicating using hand gestures and movements, body language and facial expressions, instead of spoken words [1].

1.1 Existing System

Convolutional Neural Networks (CNN). It primarily targets the recognition and conversion of Indian Sign Language (ISL) into readable text to bridge communication gaps between hearing-impaired individuals and the general public. The system uses a webcam to capture hand gestures, which are pre-processed through resizing, filtering, and segmentation techniques. These images are then passed into a CNN model trained on a dataset comprising 0–9 hand gestures, each with 1500 images [2]. Libraries such as TensorFlow, Keras, OpenCV, NumPy, and Matplotlib are employed to build, train, and visualize the model. The CNN architecture includes convolution, ReLU, pooling, and fully connected layers to detect and classify signs accurately. Image

preprocessing includes background subtraction and ROI (Region of Interest) isolation using OpenCV. After training with early stopping and learning rate reduction, the model predicts gestures in real time. A GUI displays the translated text output from the recognized signs. The system has shown high accuracy with reduced overfitting due to dropout and regularization techniques. While it effectively processes static numeric signs, the current limitation is the inability to capture dynamic gestures or encode positional data. It is a cost-effective, offline-capable solution suitable for educational and assistive communication purposes. Testing methods included unit, manual, integration, and functionality testing. The system's success demonstrates the feasibility of computer vision in assistive tech, although further enhancement is needed to handle full alphabets, words, or continuous sentences [3].

1.1.1 Challenges:

- Difficulty in understanding sign language by the general population limits effective communication.
- Lack of widespread learning and practice of sign language in society.
- Variability in gestures due to differences in individual signing styles and speeds.
- Complexity in capturing and processing dynamic hand gestures accurately.
- Skin tone variations affecting recognition in vision-based systems.
- Limited and imbalanced gesture datasets hinder robust model training.
- Requirement of real-time prediction with high accuracy under varied conditions.
- Challenges in segmenting and isolating hands from complex backgrounds.
- High dependency on lighting conditions and camera quality.
- Difficulty in recognizing gestures that involve subtle finger movements or overlaps.

1.2 Proposed system:

The proposed system is a Python-based sign language recognition model leveraging Convolutional Neural Networks (CNNs) to bridge the communication gap for the hearing and speech impaired [4]. It begins by capturing live hand gestures via a webcam, followed by image pre-processing that includes grayscale conversion, noise filtering, and region of interest (ROI) extraction. The pre-processed frames are passed through a CNN consisting of convolutional, ReLU, pooling, and fully connected layers for feature extraction and classification. A labeled dataset of sign images is used for supervised training, divided into training and testing sets to optimize prediction accuracy. The model is trained to recognize numerical hand gestures (0–9), and after successful validation, it performs real-time prediction on live input. OpenCV is used for camera interfacing and ROI handling, while TensorFlow and Keras drive model creation. Matplotlib assists in visualizing training metrics. The final output is a translated text displayed to the user, facilitating seamless communication. This modular system is scalable and can be extended to full alphabets or gesture-based commands [5].

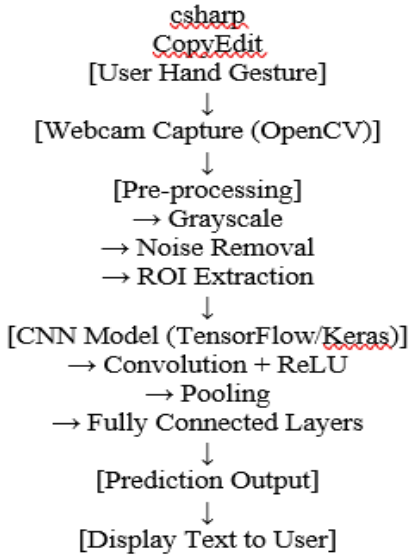


Fig: 1 Proposed Diagram

1.2.1 Advantages:

- **Efficient Conversion**: It provides a fast and accurate method of translating sign language into text or numerical data.
- **High Accuracy**: The CNN model delivers high accuracy for image recognition tasks with minimal training parameters.
- **Automatic Feature Extraction**: It automatically identifies key features without needing manual intervention.
- **Real-Time Prediction**: Capable of recognizing signs in real-time, enhancing user interaction and responsiveness.
- **Bridges Communication Gaps**: Helps connect hearing-impaired individuals with the broader society by enabling smoother communication.
- **Low-Cost Implementation**: Designed as an economical solution using widely available hardware and open-source tools.

2.1 Architecture:

The system starts by capturing live video using a webcam with the help of OpenCV. Each frame from the video is pre-processed by converting it to grayscale, resizing it to a consistent size, and applying Gaussian blur to reduce noise. A specific Region of Interest (ROI) is selected where the hand gesture is likely to appear. Background subtraction is then applied to isolate the hand from the rest of the image [6]. The segmented hand image undergoes thresholding and is resized to match the input dimensions required by the model. This processed frame is then passed to a trained Convolutional Neural Network (CNN) developed using TensorFlow and Keras. The CNN architecture includes Convolutional layers, ReLU activation functions, Pooling layers, and Fully Connected layers to effectively recognize the gesture. The model predicts the hand sign, which corresponds to digits from 0 to 9, and maps it to a textual label. During training, Matplotlib is used to visualize the model's accuracy and loss. Finally, the predicted output is displayed in real-time as translated text overlaid on the video feed [7].

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04756
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

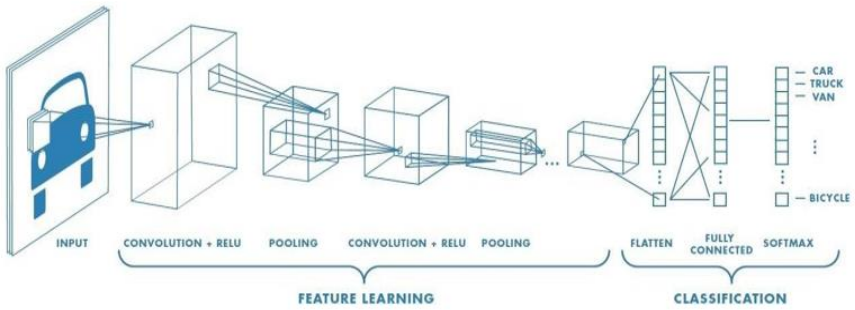


Fig:2 Architecture

UML DIAGRAMS

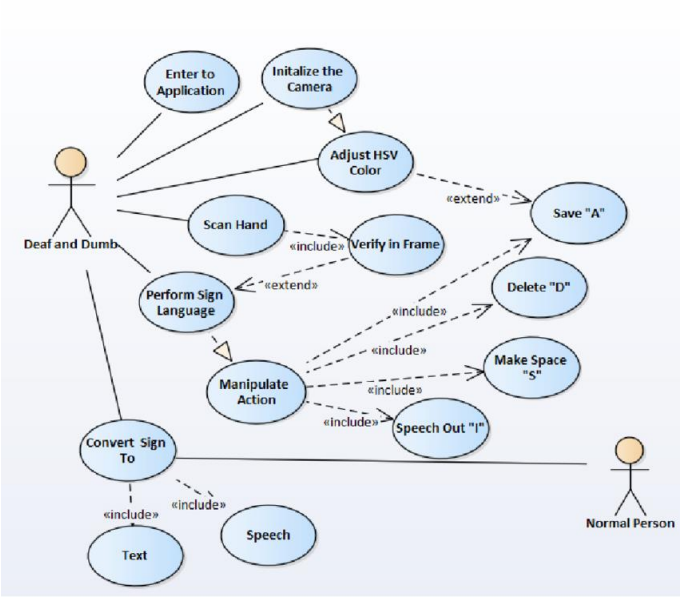


Fig:use case diagram

2.2 Algorithm:

The algorithm employed for sign language detection in this project is based on Convolutional Neural Networks (CNNs), which are ideal for image classification tasks [8]. It begins with the collection of a labeled dataset of hand gestures, which is then split into training, testing, and validation sets. The input images are pre-processed through grayscale conversion, resizing, and filtering to reduce noise. The CNN architecture is composed of four main layers**:** convolutional, ReLU activation, pooling, and fully connected layers. The convolutional layers extract essential features by applying filters to the image, while ReLU layers remove negative values to introduce non-linearity. Pooling layers downsample the feature maps to reduce dimensionality and prevent overfitting. The fully connected layers interpret these features to classify the gestures into predefined categories. Training uses ImageDataGenerator for data augmentation and Keras Sequential API to construct the model. The model is optimized using categorical cross-entropy loss and adaptive learning techniques like ReduceLROnPlateau and EarlyStopping. After training, the model predicts gestures in real-time from webcam input. A Region of Interest

(ROI) is defined to extract hand gestures, which are thresholded, resized, and passed to the CNN model. Finally, the predicted output is displayed as text, translating the gesture into human-readable form. This approach ensures efficient and accurate sign recognition with minimal preprocessing [9].

2.3 Techniques:

The project uses Convolutional Neural Networks (CNN) to recognize hand gestures representing sign language. It starts by collecting a labeled dataset of hand signs, primarily using Microsoft Kinect (v1)**.** Pre-processing techniques such as aspect ratio correction**,** cropping**,** scaling**,** and noise removal are applied to ensure consistent image input. The CNN architecture includes layers like convolution**,** ReLU, pooling, and fully connected layers for feature extraction and classification [10]. Tools like OpenCV are employed for camera access and hand segmentation using contours. Libraries such as NumPy, TensorFlow, Keras, and Matplotlib support data handling, model creation, and result visualization. Segmentation helps isolate the hand region, while thresholding refines the focus area. The training pipeline uses an 80:20 train-test split, and ImageDataGenerator from Keras streamlines batch loading. CNN model optimization includes techniques like ReduceLROnPlateau and EarlyStopping to avoid overfitting. After model training, real-time hand detection is implemented through live webcam input. Bounding boxes mark the Region of Interest (ROI), and accumulated background subtraction distinguishes the hand from the background. Predictions are made frame-by-frame, converting detected gestures into numeric or alphabetic text output. The final system enables efficient, real-time communication support for the hearing and speech impaired [11].

2.4 Tools:

The project leveraged a set of powerful Python libraries and tools for implementing sign language detection. NumPy was used for handling multidimensional arrays, especially for storing pixel values from images. TensorFlow formed the core of the machine learning framework, offering support for building and training deep learning models. On top of TensorFlow, Keras provided a high-level API to quickly construct the CNN architecture using its Sequential model. OpenCV was essential for real-time image processing, enabling access to the webcam, drawing contours, and isolating foreground elements. For visualization, Matplotlib helped in plotting accuracy and loss graphs during training. The system was built using Python 3.8.x, run on environments like Google Colab, and tested with a decent hardware setup (Intel i3 or above, 4GB+ RAM, 3MP+ camera). The CNN model was trained with datasets of hand gestures representing numbers 0-9. ImageDataGenerato**r** in Keras streamlined the dataset loading and preprocessing process. The team also utilized cv2.accumulateWeighted() and cv2.drawContours() for background modeling and gesture extraction. During live testing, bounding boxes were created to define Regions of Interest (ROI) for hand segmentation [12]. The system combined real-time video capture with classification to predict gestures. All necessary packages were installed using pip, such as pip install [13].

2.5 Methods:

The project adopts a supervised learning approach, using labeled image datasets to train the model to recognize hand gestures. The dataset is divided into training, testing, and validation sets to optimize performance [14]**.** Image pre-processing involves cropping, scaling, and removing background noise to focus on the Region of Interest (ROI). Segmentation techniques are applied to isolate the hand from the background using contours. A Convolutional Neural Network (CNN**)** model is used, consisting of convolution, ReLU, pooling, and fully connected layers to extract features and classify gestures. The CNN minimizes the need for manual feature extraction, improving accuracy. The model is trained using Keras and TensorFlow, with data augmentation via ImageDataGenerator. ReduceLROnPlateau and EarlyStopping callbacks are used during training to avoid overfitting. A live webcam feed captures hand gestures, and the processed image is fed into the trained model for real-time prediction. The final output is displayed as a recognized sign, closing the communication gap for the hearing-impaired. numpy, tensorflow, keras, opencv-python, and matplotlib. With careful implementation and the right blend of tools, the system achieved high accuracy in gesture recognition.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04756
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

III. METHODOLOGY

3.1 Input:

To predict a sign, a dataset of hand gestures representing digits (0–9) was created with 1500 images per class. These images were divided into training (80%) and testing (20%) sets. Each image underwent pre-processing, including grayscale conversion, noise filtering, and resizing. A Convolutional Neural Network (CNN) was used with layers like convolution, ReLU, pooling, and fully connected layers. The model was trained using Keras and TensorFlow, leveraging tools like ImageDataGenerator for data augmentation. During prediction, the webcam captures live input, identifies the region of interest (ROI), and extracts hand contours [15]. The thresholded image is reshaped and fed into the CNN model. The predicted sign is then displayed as text using a predefined label dictionary

```
cam = cv2.VideoCapture(0)
num_frames =0
while True:
    ret, frame = cam. read () frame =
    cv2.flip(frame, 1) frame_copy =
    frame. copy ()
    roi = frame [ROI_top: ROI_bottom, ROI_right: ROI_left] gray_frame =
    cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)gray_frame =
    cv2.GaussianBlur(gray_frame, (9, 9), 0)
    if num  frames < 70:
        cal_accum_avg (gray_frame, accumulated_weight)
        cv2.putText(frame_copy, "FETCHING BACKGROUND...PLEASE WAIT", (80, 400),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)
    else:
        hand = segment_hand(gray_frame)
            thresholded, hand_segment = hand


            cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1, (255, 0,
 0),1)
            cv2.imshow("Thesholded Hand Image", thresholded)
            thresholded = cv2.resize(thresholded, (64, 64))
            thresholded = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2RGB)
        thresholded=np. reshape (thresholded, (1, thresholded. shape [0], thresholded. shape [1],3)) pred = model.
            predict(thresholded)
            cv2.putText(frame_copy, word_dict [np. argmax(pred)], (170, 45),
 cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
     cv2.rectangle(frame_copy, (ROI_left, ROI_top), (ROI_right, ROI_bottom), (0,0,0), 3)


     num_frames += 1
     cv2.putText(frame_copy, "DataFlair hand sign recognition_ _ _", (10, 20),
 cv2.FONT_ITALIC, 0.5, (51,255,51), 1)
     cv2.imshow("Sign Detection", frame_copy)
```

Fig: input of predicted sign

3.2 Method of Process:

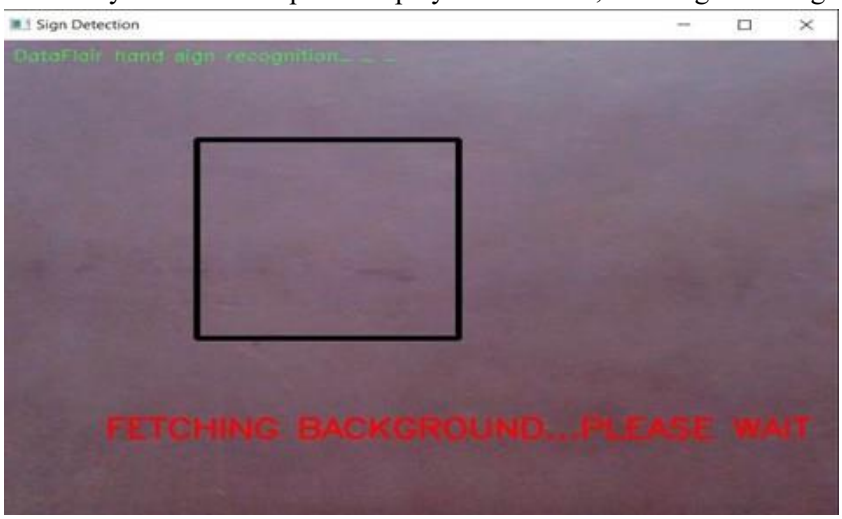
The process begins by collecting a labeled dataset of hand gestures, specifically using images representing numbers 0–9. These images are divided into training, testing, and validation sets in an 80:20 ratio. Pre-processing steps involve cropping images to a consistent aspect ratio, scaling them, and converting them to grayscale to ensure uniformity. Image segmentation is then performed to isolate the region of interest (ROI), typically the hand, using contour detection techniques. The core of the project employs a Convolutional Neural Network (CNN), comprising convolution, ReLU, pooling, and fully connected layers to extract key features from gesture images. The CNN is trained using the Keras library with TensorFlow as a backend, where accuracy is monitored using validation sets. The model incorporates callbacks like early stopping and learning rate reduction to enhance performance. Once trained, the model is saved and used in real-time gesture recognition. A webcam captures live input, and the ROI is extracted and fed into the trained model. The system then predicts the gesture and displays

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04756
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

the result. OpenCV is utilized for real-time image processing, while Matplotlib visualizes training metrics. Accuracy and loss graphs are generated to evaluate performance. The process also involves calculating accumulated background averages to detect motion. Finally, gesture prediction is displayed in a user-friendly interface, effectively translating signs into readable text [16].

3.3 Output:

The sign prediction system is designed to convert real-time hand gestures into text using a trained CNN model. A webcam captures live video, from which a specific Region of Interest (ROI) is extracted and processed. The background is first learned using accumulated average frames, isolating the hand as the main foreground. Pre-processing includes grayscale conversion, Gaussian blurring, and thresholding to highlight the hand. Contours are detected to segment the hand, creating a binary threshold image as input. The image is resized to 64x64 pixels and reshaped for the CNN model. This processed input is passed into the saved model, which then predicts the digit (0–9) represented by the gesture. The prediction is mapped to its corresponding label using a predefined dictionary. The final output is displayed on-screen, showing the recognized digit alongside the live video feed.



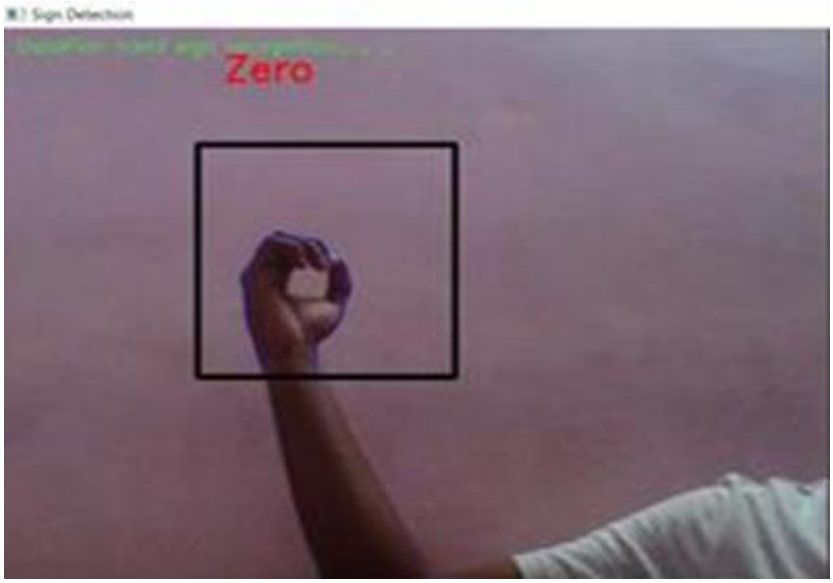


Fig: predict the sign zero

IV. RESULTS:

The project successfully divided a dataset of 0–9 hand signs into training and testing sets, trained a CNN model, and achieved strong accuracy in recognizing hand gestures. Visualizations showed consistent improvements in model accuracy and a decrease in loss over epochs. Real-time prediction using a webcam effectively detected and translated gestures into text, demonstrating high responsiveness and precision. The model used OpenCV for image capture and Keras with TensorFlow backend for training and prediction. This approach proved efficient and accessible for bridging the communication gap for hearing-impaired individuals [17].

V.DISCUSSION:

The project demonstrates a robust approach to bridging the communication gap between hearing-impaired individuals and the general public using a CNN-based sign language detection system. By leveraging computer vision and deep learning, it effectively captures and classifies hand gestures from real-time video streams. The model was trained on a substantial dataset of digit gestures (0–9) and achieved impressive accuracy through efficient preprocessing and CNN architecture. Techniques like image segmentation, ROI extraction, and contour detection were instrumental in isolating hand gestures. Real-time implementation was enabled using OpenCV and TensorFlow, allowing dynamic prediction of signs on webcam input. Despite its success, the system faced challenges like orientation sensitivity and need for abundant training data [18]. The model's scope is currently limited to numeric gestures but shows potential for full alphabet and word recognition. Future enhancements may include extending to full ISL vocabulary and integrating speech synthesis for complete sign-to-speech translation [19].

VI. CONCLUSION

In conclusion, this project offers an innovative and cost-effective solution to bridge the communication gap between the hearing-impaired and the general population through automatic sign language detection. By utilizing Convolutional Neural Networks (CNNs), the system effectively recognizes and translates static hand gestures into readable text with high accuracy. Leveraging a well-structured dataset and real-time prediction, it demonstrates strong potential for enhancing accessibility and inclusivity for speech and hearing-impaired individuals. The implementation is practical, with a user-friendly interface and reliable performance. Furthermore, it opens possibilities for future advancements such as dynamic gesture recognition, facial expression interpretation, and sign-to-voice translation. With continued improvements in model architecture and training data, the system’s precision can be further enhanced. Overall, this project highlights the power of deep learning and computer vision in empowering marginalized communities and promoting inclusive human-computer interaction.

VII. FUTURE SCOPE:

The project holds vast potential for future enhancement by expanding gesture recognition beyond alphabets and numbers to full sentences and dynamic phrases. Incorporating facial expression analysis and sign-to-speech conversion could significantly improve accessibility. Integrating multilingual sign language support would broaden its global applicability. Enhancing model accuracy through larger, more diverse datasets and transfer learning techniques can lead to real-time, robust deployment. Ultimately, this system could evolve into an inclusive communication tool for the deaf and hearing-impaired, fostering greater social integration [20].

VIII. ACKNOWLEDGEMENT:

Muppala Naga Keerthi working as an Assistant Professor in Master of Computer Applications in Sanketika Vidya Parishad Engineering College, Visakhapatnam, Andhra Pradesh, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE with 14 years of experience in Computer Science, and member in IAENG. Her areas of interests in C, JAVA, Data Structures, DBMS, Web Technologies, Software Engineering and Data Science

Prathaparao Kalahyndavi is pursuing her final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Python, CNN model P Kalahyndavi has taken up her PG project on SIGN LANGUAGE DETECTION USING CNN MODEL and published the paper in connection to the under the guidance of M Naga Keerthi, Assistant Professor, Master of Computer Applications, SVPEC.

REFERENCES

[1]AN Article Reference of grapevine
https://www.sciencedirect.com/science/article/pii/S2212977416300205
[2] AN Article Reference of the methodology
https://www.sciencedirect.com/science/article/abs/pii/S0951524096000018
[3] AN Article Reference of hearing impairments
https://www.tandfonline.com/doi/abs/10.1080/14992020500060370
[4] AN Article Reference of speech-impaired
https://www.sciencedirect.com/science/article/abs/pii/S0165587603003033
[5] AN Article Reference of deaf individuals
https://psycnet.apa.org/record/1994-26294-001
[6] AN Web Reference of flex sensor
https://onlinelibrary.wiley.com/doi/abs/10.1002/smll.201602790
[7] AN Web Reference of alphabet signs
https://www.mdpi.com/1424-8220/21/17/5856
[8] AN Web Reference of impairments

https://journals.sagepub.com/doi/abs/10.1177/0018720817708397?journalCode=hfsa
[9] AN Web Reference of prototype
https://www.sciencedirect.com/science/article/abs/pii/S0956713508003307
[10] AN Web Reference of regardless
https://onlinelibrary.wiley.com/doi/abs/10.1002/iroh.201601851
[11] A Book Reference of Image scaling
https://dl.acm.org/doi/abs/10.5555/984432
[12] A Book Reference of digital image
https://dl.acm.org/doi/abs/10.5555/4901
[13] A Book Reference of upscaling
https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.267
[14] A Book Reference of resolution enhancement
https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5377/0000/Resolution
[15] A Book Reference of magnification
https://ieeexplore.ieee.org/abstract/document/607494/
[16] AN Article Reference of Image segmentation
https://peerj.com/articles/453/?report=reader&utm_source=TrendMD&utm_campaign=PeerJ_
[17] AN Article Reference of pixel segments
https://www.hindawi.com/journals/jhe/2018/3640705/
[18] AN Web Reference of Convolutional Neural Network
https://dl.acm.org/doi/abs/10.1145/2567948.2577348
[19] AN Web Reference of Grayscale images
https://dl.acm.org/doi/abs/10.1145/1597990.1598049
[20] A Book Reference of a pooling layer
https://link.springer.com/chapter/10.1007/978-3-030-21005-2_23