Multi-headed Convolutional Neural Network for Neuro Sign Interpretation from Complex Backgrounds

Tabassum Nahid[1], Ayesha Kiran[2], Asra Fatima[3]
[1,2,3] *Assistant Professor, Khaja BandaNawaz University*

***Abstract*—Sign language recognition systems are essential for facilitating communication between hearing-impaired individuals and the general population. However, existing systems often exhibit performance degradation across different skin tones and rely on suboptimal hand-crafted features. This paper presents a Convolutional Neural Network (CNN) based approach for recognizing static sign language gestures (alphabets and words) that addresses these limitations. The proposed system incorporates an adaptive pre processing filter that achieves skin tone-invariant gesture segmentation through colour space transformation and adaptive thresholding. A four-layer CNN architecture automatically learns hierarchical feature representations, eliminating the need for manual feature engineering. Experimental results demonstrate superior recognition accuracy and consistency across diverse skin tone categories compared to traditional methods. The system provides a practical, scalable solution for automated sign language interpretation in educational, healthcare, and public service settings, contributing to improved accessibility and social inclusion for the hearing-impaired community.**

***Index Terms*—Sign language recognition, CNN, deep learning, skin tone invariance, gesture recognition, accessibility.**

I. INTRODUCTION

Sign Language Recognition (SLR) targets on interpreting the sign language into text or speech, so as to facilitate the communication between deaf-mute people and ordinary people. This task has broad social impact, but is still very challenging due to the complexity and large variations in hand actions. Existing methods for SLR use hand-crafted features to describe sign language motion and build classification models based on those features. However, it is difficult to design reliable features to adapt to the large variations of hand gestures. To approach this problem, we propose a novel 3D convolutional neural network (CNN) which extracts discriminative spatial-temporal features from raw video stream automatically without any prior knowledge, avoiding designing features. To boost the performance, multi-channels of video streams, including color information, depth clue, and body joint positions, are used as input to the 3D CNN in order to integrate color, depth and trajectory information. We validate the proposed model on a real dataset collected with Microsoft Kinect and demonstrate its effectiveness over the traditional approaches based on hand-crafted features(Jie Huang et al., 2015).Convolutional neural network (CNN) classification and feature extraction were used to construct an effective American Sign Language recognition model. With a 99.95% accuracy score on the test set and 100% precision, recall, and F1-score values for all classes, the model performed exceptionally well (Kumar et al).We presented an application that can translate sign language into English and allows users to contribute their own unique gestures. The model yields an average accuracy of 83.76 percent with an average confidence of 78 percent for predicted gestures for the 26 English language alphabets and 15 unique hand gestures being used for training (Limaye et al., 2022).The focus of this work is to create a vision based application which offers sign language translation to text thus aiding communication between signers and non-signers. The proposed model takes video sequences and extracts temporal and spatial features from them. We then use Inception, a CNN (Convolutional Neural Network) for recognizing spatial features. We then use a RNN (Recurrent Neural Network) to train on temporal features. The dataset used is the American Sign

Language Dataset(Bantupalli & Xie, 2018). The user must be able to capture images of the hand gesture using web camera and the system shall predict and display the name of the captured image. We use the HSV colour algorithm to detect the hand gesture and set the background to black. The images undergo a series of processing steps which include various Computer vision techniques such as the conversion to grayscale, dilation and mask operation. And the region of interest which, in our case is the hand gesture is segmented. The features extracted are the binary pixels of the images. We make use of Convolutional Neural Network(CNN) for training and to classify the images. We are able to recognise 10 American Sign gesture alphabets with high accuracy. Our model has achieved a remarkable accuracy of above 90%(Hurroo & Elham,2020). This paper considers an implementation of the classification algorithm for the traffic signs recognition task. Combined with preprocessing and localization steps from previous works, the proposed method for traffic signs classification shows very good results: 99.94 % of correctly classified images (Shustanov & Yakimov, 2017). ntroduction of a Convolutional Neural Network (CNN) model tailored to the identification of hand signs representing the English alphabet. For model training and validation, a dataset comprising 26,000 grayscale images of hand signs was employed. The model architecture embraced a profound CNN design, featuring numerous layers for convolution and pooling, followed by fully connected layers. Employing the Adam optimizer, the training procedure yielded an impressive accuracy of 96.7% when evaluated on the Kaggle dataset (Parashar et al., 2023). Our contribution considers a recognition system using the Microsoft Kinect, convolutional neural networks (CNNs) and GPU acceleration. Instead of constructing complex handcrafted features, CNNs are able to automate the process of feature construction. We are able to recognize 20 Italian gestures with high accuracy. The predictive model is able to generalize on users and surroundings not occurring during training with a cross-validation accuracy of 91.7% (Pigou et al, 2015). An automated ISLR system based on DWT is presented in this paper. DWT sub-band energies extracted from the hand gesture images are used as features along with the area of the segmented hand gesture region. Otsu thresholding and morphological operators are used for the segmentation procedure. DWT is applied up to the 7th level for computing sub-band energy features in order to obtain the best feature set. The nearest neighbour classifier used for the classification provides 99.23% accuracy while using the cosine distance metric (Anand et al., 2016). this paper studies hand locating and sign language recognition of common sign language based on neural network, and the main research contents include: 1. A hand locating network based on the Faster R-CNN is established to recognize the sign language video or the part of the hand in the picture, and the result of recognition is handed over to subsequent processing; 2. A 3D CNN feature extraction network and a sign-language recognition framework based on long and short time memory (LSTM) coding and decoding network are constructed for the sign language images of sequence sequences. The framework can improve the recognition accuracy by learning the context of sign language; 3. In order to solve the problem of RGB sign language image or video recognition in practical problems, this paper combines hand locating network, 3D CNN feature extraction network and LSTM encoding and decoding to build the recognition algorithm. Experimental results show that the recognition rate of this method is up to 99% in common vocabulary data set, which is better than other known methods(He, 2019). A revised end-to-end technology for detecting and recognizing traffic signs in real time. The developed system uses the speed received from the vehicle. This allows you to predict not only the presence of the object, but also the scale and its exact coordinates in the neighboring frame. Thus, the accuracy of detection increases, while the computational complexity remains the same. The classification of localized objects is implemented using convolutional neural networks (CNNs) (Shustanov & Yakimov, 2017).

This paper demonstrates a cost-effective technique to detect American Sign Language (ASL) using an image dataset. Here, “Finger Spelling, A” dataset has been used, with 24 letters (except j and z as they contain motion). The main reason for using this dataset is that these images have a complex background with different environments and scene colors.

To be specific, in order to build a CNN model, four components are typically needed. Convolution is a

pivotal step for feature extraction. The outputs of convolution can be called feature maps. When setting a convolution kernel with a certain size, we will lose information in the border. Hence, padding is introduced to enlarge the input with zero value, which can adjust the size indirectly. Besides, for the sake of controlling the density of convolving, stride is employed. The larger the stride, the lower the density. After convolution, feature maps consist of a large number of features that is prone to causing overfitting problem. As a result, pooling (a.k.a. down-sampling) is proposed to obviate redundancy, including max pooling and average pooling(Li et al., 2022). Another frequently-used optimizer is Adaptive Moment Estimation (Adam) [91] It is essentially an algorithm formed by combining the Momentum and the RMSprop. Adam stores both the exponential decay average of the past square gradients like the Adadelta algorithm and the average exponential decay average of the past gradients like the Momentum algorithm. Practice has proved that Adam algorithm works well on many problems and is applicable to many different convolutional neural network structures. We propose Adam, a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name Adam is derived from adaptive moment estimation(Kingma & Ba, 2017).Two layers of image processing have been used: in the first layer, images are processed as a whole for training, and in the second layer, the hand landmarks are extracted. A multi-headed convolutional neural network (CNN) model has been proposed and tested with 30% of the dataset to train these two layers. To avoid the overfitting problem, data augmentation and dynamic learning rate reduction have been used. With the proposed model, 98.981% test accuracy has been achieved. It is expected that this study may help to develop an efficient human–machine communication system for a deaf-mute society

II. RESEARCH GAP AND NOVELTY STUDY

Existing methods for sign language recognition often rely on sensor-based systems like IoT gloves or require specialized hardware, which may be expensive, uncomfortable, or limited in accuracy. Moreover, many current systems struggle to handle the variability in signing styles, background environments, and lighting conditions. Therefore, there is a critical need for a reliable, accurate, and real-time sign language recognition system that can work with visual input alone, such as video or camera feeds, and translates signs into spoken or written language. Sign Language Recognition (SLR) systems can help non-signers communicate with sign language users, making interactions easier in various settings (e.g., hospitals, schools, workplaces). They could allow both deaf and hearing individuals to converse naturally, translating speech into signs and vice versa. Better Accessibility: SLR could be used in devices like smart phones or smart glasses, providing real-time sign language translation. This could help people who use sign language navigate daily life more independently. In emergencies, SLR could facilitate communication between first responders and individuals who use sign language, improving safety. Learning Sign Language: SLR systems could serve as interactive tools for teaching sign language, helping both deaf and hearing individuals learn more effectively. SLR can help preserve less common or endangered sign languages by making it easier to document and learn them.

B. METHODOLOGY

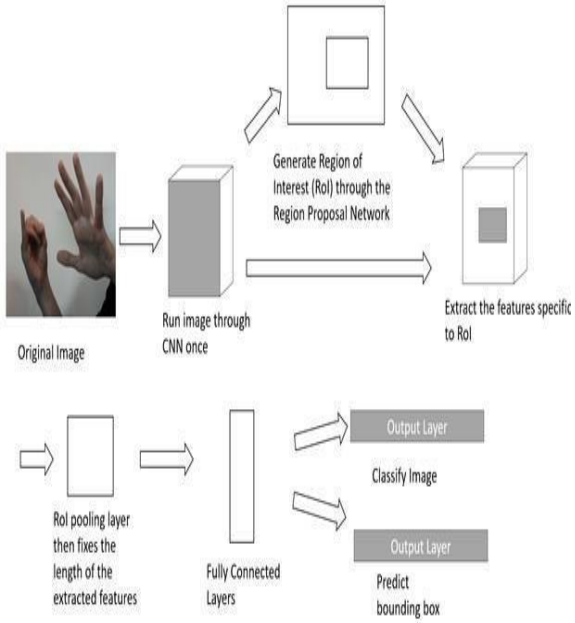

Fig. 1 Sign Language recognition using CNN implementation

A computer-mounted camera that can record real-time videos was used to collect data. When accessing the camera and storing the produced photos into the system after image preprocessing, Open CV library is utilized. When an image has been taken, hand recognition is applied to it to determine the hand's precise location electronic copy available within the entire picture. After being converted to grey scale and having their pixel sizes reduced to 64*64 pixels for computer efficiency, 300 pictures for each hand gesture are kept. During model training and testing, the file system's storage process is programmed to help with gesture classification.

The proposed system for Sign Language Recognition (SLR) utilizes advanced Convolutional Neural Networks (CNNs) to enable accurate interpretation and translation of sign language gestures into text or speech in real-time. The foundation of the system is a diverse dataset that encompasses a wide range of sign language gestures, incorporating various signing styles, regional dialects, and contextual variations. This dataset will be compiled through video recordings of signers in different environments, ensuring representation across diverse lighting conditions and backgrounds. Preprocessing techniques such as normalization, data augmentation (including rotation, scaling, and flipping), and background subtraction will enhance the dataset's quality and robustness, preparing it for effective training. At the core of the recognition system lies a custom-designed CNN architecture specifically tailored for dynamic gesture recognition. The system will be equipped with a real-time recognition pipeline that processes video input from a webcam or camera feed, extracting frames and applying the trained CNN model to identify gestures as they are performed. Once a gesture is recognized, it will be translated into corresponding text or spoken language through a predefined mapping, providing immediate feedback to users. A user-friendly interface will be developed to facilitate easy interaction with the system, displaying recognized signs and their text translations while offering visual feedback for a seamless user experience. The performance of the system will be rigorously evaluated using metrics such as accuracy, precision, recall, and latency.

Support Vector Machines (SVMs): Traditional Machine Learning Algorithms Support Vector Machines (SVMs) are used for classification and can be trained to distinguish between different sign language gestures based on extracted features. Deep CNN has been used for four class MI EEG signals classification. CNN with two convolutional layers was used to classify EEG signals acquired during Right-hand and Left-hand MI tasks with an average accuracy of 86.4% and was compared with SVM that gave accuracies of 77.17%, 82.6% and 81.25% using power, Common Spatial Pattern (CSP) and autoregressive (AR) parameters as features extracted from EEG signals(A et al., 2020).

K-Nearest Neighbors (KNNs): KNNs are used for classification based on the proximity of data points to their neighbors, which can be applied to identify sign language gestures by comparing their features to known examples.

Import Required Python Libraries: Start by importing all necessary libraries such as TensorFlow or PyTorch for deep learning, OpenCV for image processing, NumPy for numerical operations and Matplotlib for data visualization. These libraries provide the tools needed to handle images, build models, and train them effectively. Collect Images: Gather a dataset of images representing various sign language gestures. This can be done by using publicly available datasets or capturing images manually. A diverse dataset improves the model’s ability to generalize across different users, lighting conditions, and backgrounds. Preprocess Images: Resize all images to a consistent size and normalize pixel values to a standard scale (usually 0 to 1). This step ensures the model receives uniform input, improving learning efficiency. Additional preprocessing like data augmentation (rotation, flipping, etc.) can help improve robustness.Build CNN Model: Design a Convolutional Neural Network (CNN) architecture tailored for image classification. Include layers such as convolutional layers to extract features, pooling layers to reduce dimensionality, and dense layers for final classification. The CNN is responsible for learning visual patterns associated with each sign.Train the Model: Feed the preprocessed images into the CNN and train it using a labeled dataset. The model learns to map input gestures to their corresponding labels by minimizing prediction errors over multiple iterations using optimization algorithms like Adam.

Test the Model: Evaluate the trained model on a separate test dataset to assess its accuracy and

performance. Use metrics like accuracy, precision, recall, and confusion matrices to analyze how well the model recognizes each sign and to detect any weaknesses. Predict the Sign: Once trained and validated, use the model to predict the sign language gesture from new, unseen images. The output can be displayed as text or converted to speech, enabling real time gesture recognition and communication support for the hearing-impaired.

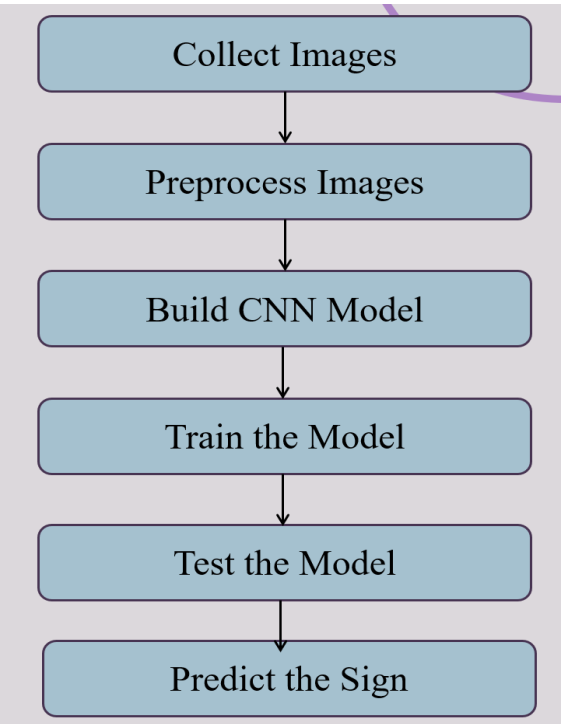

Fig.2 Training Model

C. Training Data

1. Cv2.VideoCapture (0) Initializes the webcam (device 0) for real-time video capture, allowing frame-by-frame reading from the camera.

2. HandDetector(maxHands=1) Creates a hand detector object from cvzone that uses MediaPipe to detect up to one hand in the video frames.

3. Classifier("Model/keras_model.h5","Model/labels.txt") Loads a pre-trained Keras classification model and its label file for recognizing specific hand gestures.

4. cap.read() Captures a single frame from the webcam to process in each iteration of the loop.

5. detector.findHands(img) Detects hands in the current frame and returns the hand landmarks and the modified image.

6. hand['bbox'] Extracts the bounding box (x, y, width, height) around the detected hand for cropping the image.

7.np.ones((imgSize,imgSize,3),np.uint8)*255 Creates a blank white image of size 300x300 pixels used as the base for standardized input to the model.

8. cv2.resize() Resizes the cropped hand image to fit within the white image while maintaining aspect ratio.

9.classifier.getPrediction(imgWhite,draw=False) Uses the pre-trained model to predict the hand gesture from the standardized input image.

10. cv2.rectangle() Draws rectangles for UI display—one around the hand and one as a background for the predicted label.

11. cv2.putText() Displays the name of the predicted gesture label on the frame above the hand.

12. cv2.imshow() Opens and updates the real-time display windows for the original image, cropped hand, and white canvas.

13. cv2.waitKey(1) Waits 1 millisecond for a key event, allowing the display to update in real time.

d. DATA COLLECTION

1. cv2.VideoCapture(0) Opens the default webcam to capture real-time video frames.

2. Hand Detector (maxHands=1) Initializes the hand detection module from cvzone to detect one hand at a time in each frame.

3. np.ones((imgSize,imgSize,3),np.uint8)*255 Creates a blank white 300x300 image used as a standard background for resizing hand crops.

4. cap.readq() Reads the current frame from the webcam feed for processing.

5. detector.findHands(img) Detects hands in the frame and returns both the modified image with annotations and hand details.

6. hand['bbox'] Retrieves the bounding box (x, y, width, height) coordinates around the detected hand.

7. cv2.resize() Resizes the cropped hand image proportionally to fit the 300x300 white image depending on aspect ratio.

8. AspectRatioHandling() Maintains hand image proportions during resizing — different logic depending on whether the hand is taller or wider.

9. cv2.imshow() Displays the cropped hand image, white canvas (resized version), and the original frame with real-time updates.

cv2.waitKey(1)

Keepsthewindowresponsive and checks forkeypresseseverymillisecond.
If_key==ord("s"):
Checksifthe's'keyispressed;ifso,savesthecurrenthandgestureimagetothespecified folder.
cv2.imwrite()
Savesthestandardizedimagetothe"Okay"folderwithauniquefilenameusingthe current time.
print(counter)
Displaysthenumberofimagessavedsofarfor dataset tracking.

III. **Implementation**

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier("Model/keras_model.h5" , "Model/labels.txt")
offset = 20
imgSize = 300
counter = 0
labels = ["Hello","I love you","No","Okay","Please","Thankyou","Yes"]
while True:
success, img = cap.read()
imgOutput = img.copy()
hands, img = detector.findHands(img)
 if hands:
 hand = hands[0]
 x, y, w, h = hand['bbox']
imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
imgCrop = img[y-offset:y + h + offset, x-offset:x + w + offset]
imgCropShape = imgCrop.shape
aspectRatio = h / w
 if aspectRatio> 1:
 k = imgSize / h
wCal = math.ceil(k * w)
imgResize = cv2.resize(imgCrop, (wCal, imgSize))
imgResizeShape = imgResize.shape
wGap = math.ceil((imgSize-wCal)/2)
imgWhite[:,wGap: wCal + wGap] = imgResize
prediction , index = classifier.getPrediction(imgWhite, draw= False)
print(prediction, index)
else:
 k = imgSize / w
hCal = math.ceil(k * h)
imgResize = cv2.resize(imgCrop, (imgSize, hCal))
imgResizeShape = imgResize.shape
hGap = math.ceil((imgSize - hCal) / 2)
imgWhite[hGap: hCal + hGap, :] = imgResize
prediction , index = classifier.getPrediction(imgWhite, draw= False)
      cv2.rectangle(imgOutput,(x-offset,y-offset-70),(x -offset+400, y - offset+60-50),(0,255,0),cv2.FILLED)

      cv2.putText(imgOutput,labels[index],(x,y-30),cv2.FONT_HERSHEY_COMPLEX,2,(0,0,0),2)
      cv2.rectangle(imgOutput,(x-offset,y-offset),(x + w + offset, y+h + offset),(0,255,0),4)

      cv2.imshow('ImageCrop', imgCrop)
      cv2.imshow('ImageWhite', imgWhite)

   cv2.imshow('Image', imgOutput)
   cv2.waitKey(1)
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset = 20
imgSize = 300
counter = 0
folder = "Data/Okay"
while True:
   success, img = cap.read()
   hands, img = detector.findHands(img)
   if hands:
      hand = hands[0]
      x, y, w, h = hand['bbox']
imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
```

```
imgCrop = img[y-offset:y + h + offset, x-offset:x + w
+ offset]
imgCropShape = imgCrop.shape

aspectRatio = h / w

        if aspectRatio> 1:
            k = imgSize / h
wCal = math.ceil(k * w)
imgResize = cv2.resize(imgCrop, (wCal, imgSize))
imgResizeShape = imgResize.shape
wGap = math.ceil((imgSize-wCal)/2)
imgWhite[:,wGap: wCal + wGap] = imgResize

        else:
            k = imgSize / w
hCal = math.ceil(k * h)
imgResize = cv2.resize(imgCrop, (imgSize, hCal))
imgResizeShape = imgResize.shape
hGap = math.ceil((imgSize - hCal) / 2)
imgWhite[hGap: hCal + hGap, :] = imgResize

        cv2.imshow('ImageCrop', imgCrop)
        cv2.imshow('ImageWhite', imgWhite)

    cv2.imshow('Image', img)
    key = cv2.waitKey(1)
    if key == ord("s"):
        counter += 1

cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',
imgWhite)
        print(counter)
```

IV. INTERPRETATION OF RESULTS

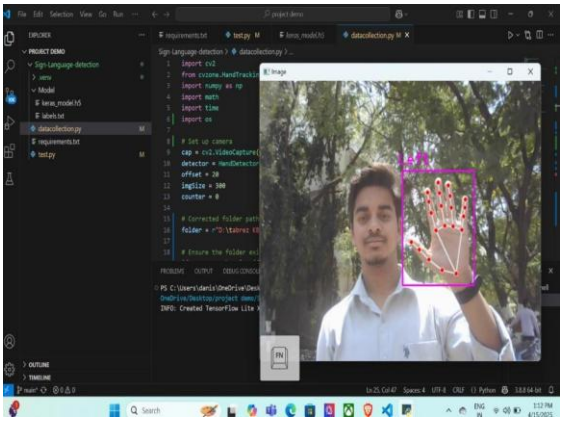

Fig.3DataCollectionfor“Hello”Sign

Gather a dataset of image in Fig 3 representing various sign language gestures such as hello, yes, please etc. This can be done by using publicly available datasets or capturing images manually diverse dataset improves the model’s ability to generalize across different users, lighting conditions and backgrounds. Design a Convolution Neural Network (CNN) architecture tailored for image classification

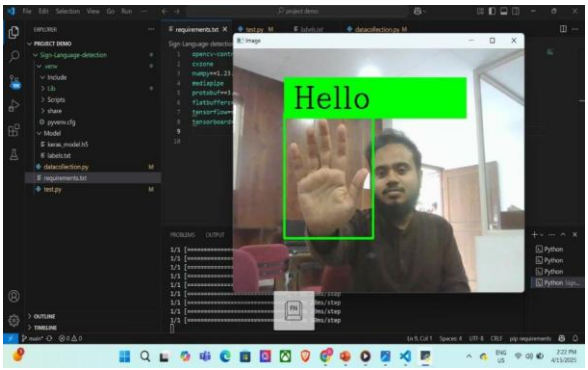

Fig.4 Trained model“Hello” Sign

Once trained and validated, use the CNN model to predict the sign language gesture from new, unseen images. Feed the Preprocessed images into the CNN and train it using a labeled dataset. The CNN is responsible for learning visual patterns associated with each sign. The model learns to map input gestures to their corresponding labels by minimizing prediction errors. The output can be displayed as text or converted to speech, enabling real- time gesture recognition such as “Hello” Sign and communication support for the hearing-impaired as shown in Fig 4

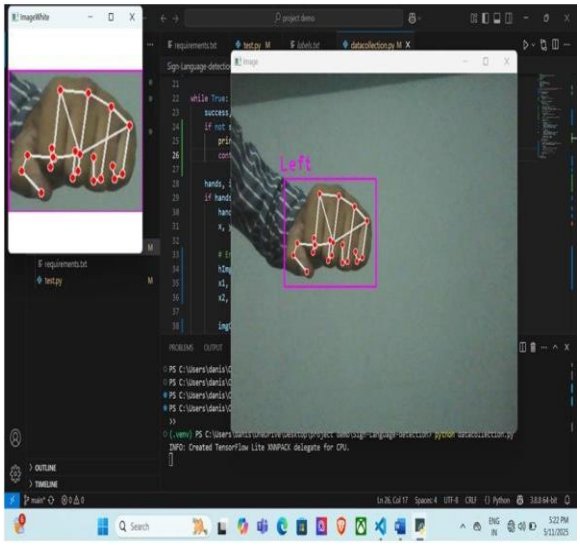

Fig.5 Data collection for “yes” sign

Gather a dataset of image in Fig 5 representing various sign language gestures such as yes, please etc. This can be done by using publicly available datasets or capturing images manually diverse dataset improves the model's ability to generalize across different users, lighting conditions, and backgrounds.

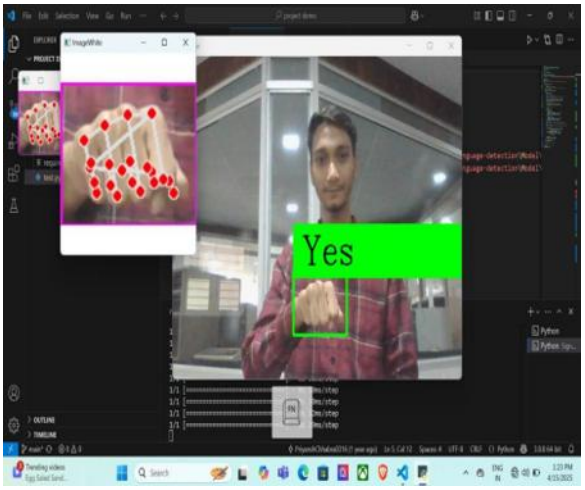

Fig.6 Trained model of "yes" sign

Once trained and validated, use the CNN model to predict the sign language gesture from new, unseen images. Feed the Preprocessed images into the CNN and train it using a labeled dataset. The CNN is responsible for learning visual patterns associated with each sign. The model learns to map input gestures to their corresponding labels by minimizing prediction errors. The output can be displayed as text or converted to speech, enabling real- time gesture recognition such as "Hello" Sign and communication support for the hearing-impaired as shown in Fig 6

VI. FUTURE SCOPE

The future of Neurosign Interpretation in sign language recognition holds immense potential for technological and social advancement. One major area of growth lies in expanding the system to support dynamic and continuous sign recognition, allowing it to understand full sentences or conversations, rather than isolated gestures. Integration with Natural Language Processing (NLP) an enhance the system's ability to provide context-aware translations, making communication more fluid and accurate. Additionally, incorporating multimodal data—such as facial expressions, body posture, and motion sensors—can further improve recognition accuracy and better capture the nuances of sign language.

There is also significant scope for applying this technology in mobile and wearable platforms, enabling real-time communication to on-the-go through smart phones, AR glasses, or smart gloves. In educational settings, it can serve as a powerful tool for teaching and learning sign language. Moreover, the system can be integrated into public service kiosks, healthcare interfaces, and virtual assistants, improving accessibility in diverse environments.

With advances in AI and edge computing, future versions of Neurosign Interpretation can operate with lower latency and higher speed, even on low-power devices. Lastly, with broader datasets representing different regional sign languages and dialects, the system can be scaled globally, fostering inclusivity and empowering millions of hearing-impaired individuals around the world.

VII. CONCLUSION

The Neurosign Interpretation system represents a significant advancement in the field of sign language recognition, offering an innovative approach that bridges the communication gap between hearing-impaired individuals and the broader community. By integrating deep learning techniques such as Convolutional Neural Networks (CNNs) with powerful tools like Media Pipe and Open CV, the system is capable of accurately interpreting complex sign language gestures in real-time. The use of diverse and context-rich datasets ensures that the model is robust, inclusive, and adaptable to various signing styles and environments. Furthermore, the ability of the system to translate gestures into text or speech provides a practical and scalable solution for enhancing accessibility. While challenges such as gesture variability, environmental conditions, and real-time performance still exist, the Neurosign Interpretation system lays a strong foundation for future improvements. Continued research, expanded datasets, and optimization of neural network models will further improve accuracy, making this technology an essential tool in assistive communication and inclusive human-computer interaction.

ACKNOWLEDGMENT

I thank all the staff of research Centre, Khaja BandaNawaz University, Kalaburagi.

REFERENCES

[1] A, Janani., M, Sasikala., Chhabra, H., Shajil, N., & Venkatasubramanian, G. (2020). Investigation of deep convolutional neural network for classification of motor imagery fNIRS signals for BCI applications. Biomedical Signal Processing and Control, 62, 102133. https://doi.org/10.1016/j.bspc.2020.102133

[2] Anand, M. S., Kumar, N. M., & Kumaresan, A. (2016). An Efficient Framework for Indian Sign Language Recognition Using Wavelet Transform. Circuits and Systems, 7(8), 1874–1883. https://doi.org/10.4236/cs.2016.78162

[3] Bantupalli, K., & Xie, Y. (2018). American Sign Language Recognition using Deep Learning and Computer Vision. 2018 IEEE International Conference on Big Data (Big Data), 4896–4899. https://doi.org/10.1109/BigData.2018.8622141

[4] He, S. (2019). Research of a Sign Language Translation System Based on Deep Learning. 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), 392–396. https://doi.org/10.1109/AIAM48774.2019.00083

[5] Jie Huang, Wengang Zhou, Houqiang Li, & Weiping Li. (2015). Sign Language Recognition using 3D convolutional neural networks. 2015 IEEE International Conference on Multimedia and Expo (ICME), 1–6. https://doi.org/10.1109/ICME.2015.7177428

[6] Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization (arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

[7] Kumar, R., Singh, S. K., Bajpai, A., & Sinha, A. (n.d.). Mediapipe and CNNs for Real-Time ASL Gesture Recognition.

[8] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. IEEE Transactions on Neural Networks and Learning Systems, 33(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

[9] S. Parashar, D., Thakur, S., Raju, K. B., Madhavi, G. B., & Sharma, K. (2023). A Deep Learning-Based Approach for Hand Sign Recognition Using CNN Architecture. Revue d'Intelligence Artificielle, 37(4), 937–943. https://doi.org/10.18280/ria.370414

[10] Pigou, L., Dieleman, S., Kindermans, P.-J., & Schrauwen, B. (2015). Sign Language Recognition Using Convolutional Neural Networks. In L. Agapito, M. M. Bronstein, & C. Rother (Eds.), Computer Vision—ECCV 2014 Workshops (pp. 572–578). Springer International Publishing. https://doi.org/10.1007/978-3-319-16178-5_40

[11] Shustanov, A., & Yakimov, P. (2017). CNN Design for Real-Time Traffic Sign Recognition. Procedia Engineering, 3rd International Conference "Information Technology and Nanotechnology", ITNT-2017, 25-27 April 2017, Samara, Russia, 201, 718–725. https://doi.org/10.1016/j.proeng.2017.09.594