Chhatrapati Sahuji Maharaj University
Kanpur-208024

A

Project

ON

"Sign Language Recognition and Conversion to Audio using Convolutional Neural Network "

Er.Preeti Singh **Dr. Niraj Kumar**

Er.Anand Kumar Gupta

(Project Guide) **(Project Incharge)**

Submitted By:

Vidushi Shukla (230)

Komal Chaudhary (195)

Anuradha Pal (180)

Alok Kumar Gautam(174)

Sahil (216)

Department Of Electronics And Communication Engineering

University Institute Of Engineering and Technology

CSJM University, Kanpur, 208024.

University Institute Of Engineering & Technology

Chhatrapati Sahuji Maharaj University

Kanpur-208024

CERTIFICATE

This is to certify that the work in this project report entitled "**Sign Language Recognition and Conversion to audio using Convolutional Neural Network ”** submitted by Vidushi Shukla, Komal Chaudhary, Anuradha Pal, Alok Kumar Gautam, Sahil in partial fulfillment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering in the department of Electronics and Communication Engineering, U.I.E.T, C.S.J.M University is a bonafide record of project work carried out by him/her under my/our supervision. The contents of this report, in full or in parts, have not been submitted to any other institution or University for the award of any degree or diploma.

Er.Preeti Singh
Er. Anand Kumar Gupta **Dr.Niraj Kumar**

(Project Guide) **(Project Incharge)**

DECLARATION

We declare that this project report titled "**Sign Language Recognition and Conversion to audio using Convolutional Neural Network**" submitted in partial fulfillment of the degree of B.Tech in (Electronics and Communication Engineering) is a record of original work carried by Vidushi Shukla, Komal Chaudhary, Anuradha Pal, Alok Kumar Gautam, Sahil under the supervision of Er. Preeti Singh and Er.Anand Kumar Gupta, and has not formed the basis of award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Er.Preeti Singh
Er. Anand Kumar Gupta **Dr.Niraj Kumar**
(Project Guide) **(Project Incharge)**

ACKNOWLEDGEMENT

This Progress report is an outcome of the project work which we have gone through in the final year of B.tech program till our Endsem of 8th semester. We would like to express deep sense of gratitude towards Dr. Ajeet Kumar Srivastava, HOD of ECE Department and Dr.Niraj Kumar, Project Incharge who gave us opportunity to present and work on the project titled "**Sign Language Recognition and Conversion to audio using Convolutional Neural Network**".

Special thanks to Er.Anand Kumar Gupta and Er. Preeti Singh, project guide for their time to time very much needed and valuable guidance. Without their continuous mentorship and invaluable insights, which have significantly shaped the direction of this research, we could not work on this project effieciently.

I extend my appreciation to my group members for their feedback, collaboration, and support during the project's critical phases.

Finally, I wish to acknowledge the developers and contributors of the open- source libraries and frameworks, such as MediaPipe, TensorFlow, and OpenCV, whose tools have been instrumental in realizing this project's technical goals.

This project's progress reflects the collective effort and support of this community, and I am truly grateful for their contributions

Vidushi Shukla(230)
Komal Chaudhary(195)
Anuradha Pal (180)
Alok Kumar Gautam(174)
Sahil (216)

ABSTRACT

This report details the development and evaluation of a real-time sign language recognition system leveraging Convolutional Neural Networks (CNNs) for the conversion of hand gestures into spoken language. The system employs a multi-layered approach, initially processing video frames captured by a webcam using Gaussian blur and thresholding for feature extraction. These processed images are then fed into a CNN model trained to predict individual letters of the sign language alphabet. To ensure robust letter detection, a temporal filtering mechanism considers a letter valid only if detected consistently across more than 50 consecutive frames. Furthermore, the system incorporates logic for handling word spacing using a designated blank symbol.

Significant improvements were made to the CNN architecture, including the adoption of three convolutional layers with an increasing number of filters (32, 64, and 128) optimized for grayscale input images of size 128x128x1. Training for 10 epochs resulted in a substantial increase in both training and validation accuracy, achieving near-perfect validation accuracy (almost 99.90%), demonstrating the model's enhanced learning and generalization capabilities compared to a baseline model.

The system also includes a frontend interface, facilitated by a server intermediary, for displaying the recognized text. A confidence-based mechanism (> 0.5) and a temporal majority vote over the last 10 detections are used to finalize detected letters. Users are provided with functionalities to manipulate the formed words, including clearing letters, adding spaces, and saving words. Finally, the recognized text can be converted to speech using the browser's Web Speech API upon user request. This work presents an effective pipeline for sign language recognition and its conversion to audible speech, offering a potential communication aid for individuals who use sign language.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. **Introduction**:

American sign language is a predominant sign language Since the only disability Deaf and Dumb (hereby referred to as D&M) people have is communication related and since they cannot use spoken languages, the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals. D&M people make hands to express use of their different gestures to express their ideas with other people. Gestures are the non-verbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language. A sign language is a language which uses gestures instead of sound to convey meaning combining hand-shapes, orientation and movement of the hands, arms or body, facial expressions and lip-patterns. Contrary to popular belief, sign language is not international. These vary from region to region.

Fingerspelling	Word level sign vocabulary	Non-manual features
Used to spell words letter by letter.	Used for the majority of communication	Facial expressions and tongue, mouth, and boy position

Table -1 Introduction to Sign Language

Sign language is a visual language and consists of 3 major components:

Minimizing the verbal exchange gap among D&M and non-D&M people turns into a desire to make certain effective conversation among all. Sign language translation is among one of the most growing lines of research and it enables the maximum natural manner of communication for those with hearing impairments. A hand gesture recognition system offers an opportunity for deaf people to talk with vocal humans without the need of an interpreter. system is built for the automated conversion of ASL into textual content and speech.

In our project we primarily focus on producing a model which can recognize Fingerspelling based hand gestures in order to form a complete word by combining each gesture. The gestures we aim to train are as given in the image below.

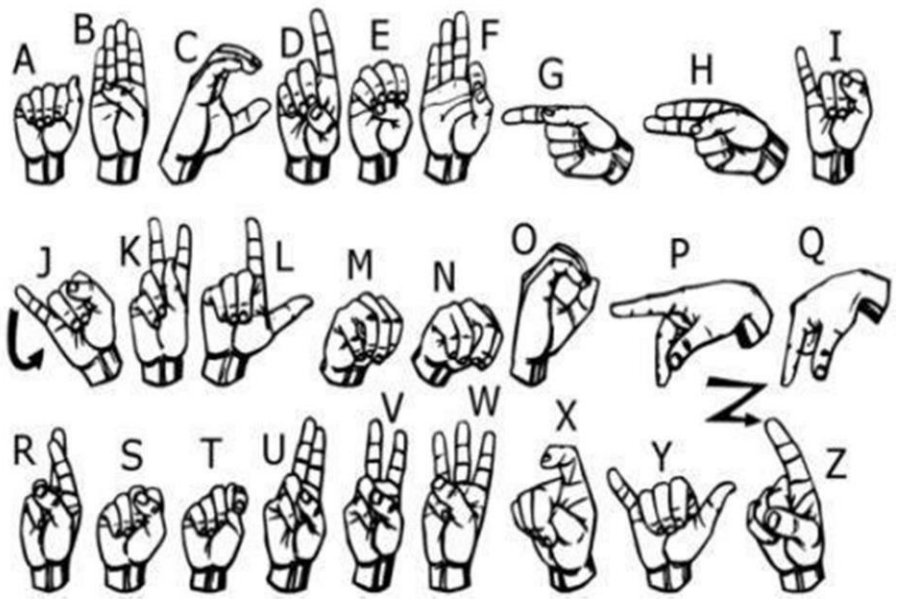


Figure - 1

1.1 Global Statistics on Hearing and Speech Impairment

According to the World Health Organization (WHO), over 430 million people worldwide suffer from disabling hearing loss. It is projected that by 2050, over 700 million people will require hearing rehabilitation services. A significant proportion of these individuals rely on sign language for daily communication, especially those who are pre-lingually deaf.

These individuals often face social exclusion due to communication barriers, and technological tools aimed at bridging this gap can have a transformative impact.

1.2 Historical Development of Sign Language

Sign languages have existed for centuries. One of the earliest recorded uses of sign language for communication was documented in the 5th century BC. In

the 16th century, Spanish monk Pedro Ponce de León used manual signs to teach deaf children.In the 1800s, Thomas Hopkins Gallaudet helped develop American Sign Language (ASL) after collaborating with French Sign Language experts. Today, ASL is one of the most widely used sign languages in the world.

1.3 Importance of Inclusive Technology

Inclusive technologies such as automatic sign language interpreters can democratize communication, promote social inclusion, and provide equitable access to education, healthcare, and employment. With advancements in AI and computer vision, real-time recognition and conversion of signs into spoken language is becoming increasingly viable.

2. **Motivation:**

For interaction between normal people and D&M people a language barrier is created as sign language structure since it is different from normal text. So, they depend on vision-based communication for interaction.

If there is a common interface that converts the sign language to text, then the gestures can be easily understood by non-D&M people. So, research has been made for a vision-based interface system where D&M people can enjoy communication without really knowing each other's language.

The aim is to develop a user-friendly Human Computer Interface (HCI) where the computer understands the human sign language.

There are various sign languages all over the world, namely American Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language, Japanese Sign Language and work has been done on other languages all around the world.

2.1 Real-World Communication Barriers Faced by D&M Individuals

People with hearing and speech disabilities often face systemic challenges in everyday communication. From basic services to high-stakes interactions, they are frequently left out due to the lack of sign language interpreters or inclusive technology. This exclusion is particularly severe in the following sectors:

1. Education

Deaf students often attend mainstream schools where teachers and peers do not understand sign language. Without real-time interpretation, these students struggle to follow lessons, ask questions, or participate in discussions. This leads to reduced academic performance and social isolation. Technology that translates sign language to speech can help bridge this gap and create more inclusive classrooms.

2. Healthcare

In medical environments, effective communication is critical. Deaf patients may struggle to explain symptoms or understand diagnoses if no interpreter is present. Miscommunication can lead to incorrect treatment, lack of consent understanding, and severe patient risk. A sign-to-speech system could enable better communication in emergencies or rural clinics with no interpreters.

3. Employment

Job interviews, workplace meetings, and even day-to-day tasks can become inaccessible to D&M individuals due to language barriers. Employers may hesitate to hire skilled candidates simply due to communication concerns. Introducing assistive tools that provide real-time translation can improve inclusion and productivity in professional environments.

4. Public Services and Law Enforcement

Situations involving law enforcement, disaster relief, or public transportation require instant communication. D&M individuals are at a disadvantage in emergencies when their communication needs are not met. Technological solutions that interpret signs and convert them to audio can play a life-saving role in such scenarios.

5. Daily Social Interactions

Even casual activities like shopping, dining, or traveling become challenging when service staff cannot understand sign language. This affects the mental health and independence of D&M individuals, reinforcing a cycle of exclusion.

2.2 Why Technology Must Fill the Gap

Despite the growing awareness around inclusion, there is a significant shortage of human sign language interpreters, particularly in low-income and rural areas. As such, technology must act as a bridge — ensuring that D&M individuals can express themselves and understand others without relying on intermediaries.By leveraging machine learning, computer vision, and audio synthesis, systems like ours can help remove these communication barriers — making society more inclusive, efficient, and humane.

3. Literature Review:

3.1 EXISTING SYSTEM

In recent years there has been tremendous research done on hand gesture recognition. In Literature survey we have gone through other similar works that are implemented in the domain of sign language recognition. The summaries of each of the project works are mentioned below:

- **A Survey of Hand Gesture Recognition Methods in Sign Language Recognition:**

  Sign Language Recognition (SLR) system, which is required to recognize sign languages, has been widely studied for years.The studies are based on various input sensors, gesture segmentation, extraction of features and classifcation methods. This paper aims to analyze and compare the methods employed in the SLR systems, classi cations methods that have been used, and suggests the most promising method for future research. Due to recent advancement in classification methods, many of the recent proposed works mainly contribute on the classifcation methods, such as hybrid method and Deep Learning. This paper focuses on the classifcation methods used in prior Sign Language Recognition system. Based on our review, HMM- based approaches have been explored extensively in prior research, including its modifcations. This study is based on various input sensors, gesture segmentation, extraction of features and classification methods. This paper aims to analyze and compare the methods employed in the SLR systems, classifications methods that have been used, and suggests the most reliable method for future research. Due to recent advancement in classification methods, many of the recently proposed works mainly contribute to the classification methods, such as hybrid method and Deep Learning. Based on our review, HMM-based approaches have been explored extensively in prior research, including its modifications. Hybrid CNN-HMM and fully Deep Learning approaches have shown promising results and offer opportunities for further exploration.

- **Communication between Deaf-Dumb People and Normal People:**

  Chat applications have become a powerful mediathat assist people to communicate in different languages witheach other. There are lots of chat applications that are useddifferent people in different languages but there are not such achat application

that has facilitate to communicate with signlanguages. The developed system isbased on Sinhala Sign language. The system has included fourmain components as text messages are converted to sign messages, voice messages are converted to sign messages, signmessages are converted to text messages and sign messages areconverted to voice messages. Google voice recognition API hasused to develop speech character recognition for voice messages.The system has been trained for the speech and text patterns by usingsome text parameters and signs of Sinhala Sign language isdisplayed by emoji. Those emoji and signs that are included inthis system will bring the normal people more close to the disabled people. This is a 2 way communication system but it uses pattern of gesture recognition which is not very realiable in getting appropriate output.

Figure-2 Talking in sign language

- **A System for Recognition of Indian Sign Language for Deaf People using Otsu's Algorithm:**

  In this paper we proposed some methods, through which the recognition of the signs becomes easy for peoples while communication. And the result of those symbols signs will be converted into the text. In this project, we are capturing hand gestures through webcam and convert this image into gray scale image. The segmentation of gray scale image of a hand gesture is performed using Otsu thresholding algorithm. Total image level is divided into two classes one is hand and other is background. The optimal threshold value is determined by computing

the ratio between class variance and total class variance. To find the boundary of hand gesture in image Canny edge detection technique is used. In Canny edge detection we used edge based segmentation and threshold based segmentation. Then Otsuʿs algorithm is used because of its simple calculation and stability. This algorithm fails, when the global distribution of the target and background vary widely.

- **Intelligent Sign Language Recognition Using Image Processing :**

  Computer recognition of sign language is an important research problem for enabling communication with hearing impaired people. This project introduces an efficient and fast algorithm for identification of the number of fingers opened in a gesture representing an alphabet of the Binary Sign Language. The system does not require the hand to be perfectly aligned to the camera. The project uses image processing system to identify, especially English alphabetic sign language used by the deaf people to communicate. The basic objective of this project is to develop a computer based intelligent system that will enable dumb people significantly to communicate with all other people using their natural hand gestures. The idea consisted of designing and building up an intelligent system using image processing, machine learning and artificial intelligence concepts to take visual inputs of sign languageʿs hand gestures and generate easily recognizable form of outputs. Hence the objective of this project is to develop an intelligent system which can act as a translator between the sign language and the spoken language dynamically and can make the communication between people with hearing impairment and normal people both effective and efficient. The system is we are implementing for Binary sign language but it can detect any sign language with prior image processing.

- **Sign Language Recognition Using Image Processing:**

  One of the major drawback of our society is the barrier that is created between disabled or handicapped persons and the normal person. Communication is the only medium by which we can share our thoughts or convey the message but for a person with disability (deaf and dumb) faces difficulty in communication with normal person. For many deaf and dumb people , sign language is the basic means of communication. Sign language recognition (SLR) aims to interpret sign

languages automatically by a computer in order to help the deaf communicate with hearing society conveniently. Our aim is to design a system to help the person who trained the hearing impaired to communicate with the rest of the world using sign language or hand gesture recognition techniques. In this system, feature detection and feature extraction of hand gesture is done with the help of SURF algorithm using image processing. All this work is done using MATLAB software. With the help of this algorithm, a person can easily trained a deaf and dumb.

- **Sign Language Interpreter using Image Processing and Machine Learning :**

  Speech impairment is a disability which affects one's ability to speak and hear. Such individuals use sign language to communicate with other people. Although it is an effective form of communication, there remains a challenge for people who do not understand sign language to communicate with speech impaired people. The aim of this paper is to develop an application which will translate sign language to English in the form of text and audio, thus aiding communication with sign language. The application acquires image data using the webcam of the computer, then it is preprocessed using a combinational algorithm and recognition is done using template matching. The translation in the form of text is then converted to audio. The database used for this system includes 6000 images of English alphabets. We used 4800 images for training and 1200 images for testing. The system produces 88% accuracy.

- **Hand Gesture Recognition based on Digital Image Processing using MATLAB:**

  This research work presents a prototype system that helps to recognize hand gesture to normal people in order to communicate more effectively with the special people. Aforesaid research work focuses on the problem of gesture recognition in real time that sign language used by the community of deaf people. The problem addressed is based on Digital Image Processing using Color Segmentation, Skin Detection, Image Segmentation, Image Filtering, and

Template Matching techniques. This system recognizes gestures of ASL (American Sign Language) including the alphabet and a subset of its words.

- **Gesture Recognition System**

  Communication plays a crucial part in human life. It encourages a man to pass on his sentiments, feelings and messages by talking, composing or by utilizing some other medium. Gesture based communication is the main method for Communication for the discourse and hearing weakened individuals. Communication via gestures is a dialect that utilizations outwardly transmitted motions that consolidates hand signs and development of the hands, arms, lip designs, body developments and outward appearances, rather than utilizing discourse or content, to express the individual's musings. Gestures are the expressive and important body developments that speaks to some message or data. Gestures are the requirement for hearing and discourse hindered, they pass on their message to others just with the assistance of motions. Gesture Recognition System is the capacity of the computer interface to catch, track and perceive the motions and deliver the yield in light of the caught signals. It enables the clients to interface with machines (HMI) without the any need of mechanical gadgets. There are two sorts of sign recognition methods: image- based and sensor- based strategies. Image based approach is utilized as a part of this project that manages communication via gestures motions to distinguish and track the signs and change over them into the relating discourse and content.

3.2 PROPOSED SYSTEM

Our proposed system is sign language recognition system using convolution neural networks which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained model. Thus our system is more robust in getting exact text labels of letters. With the help of literature survey, we realized that the basic steps in hand gesture recognition are: -

- Data acquisition
- Data pre-processing
- Feature extraction
- Gesture classification

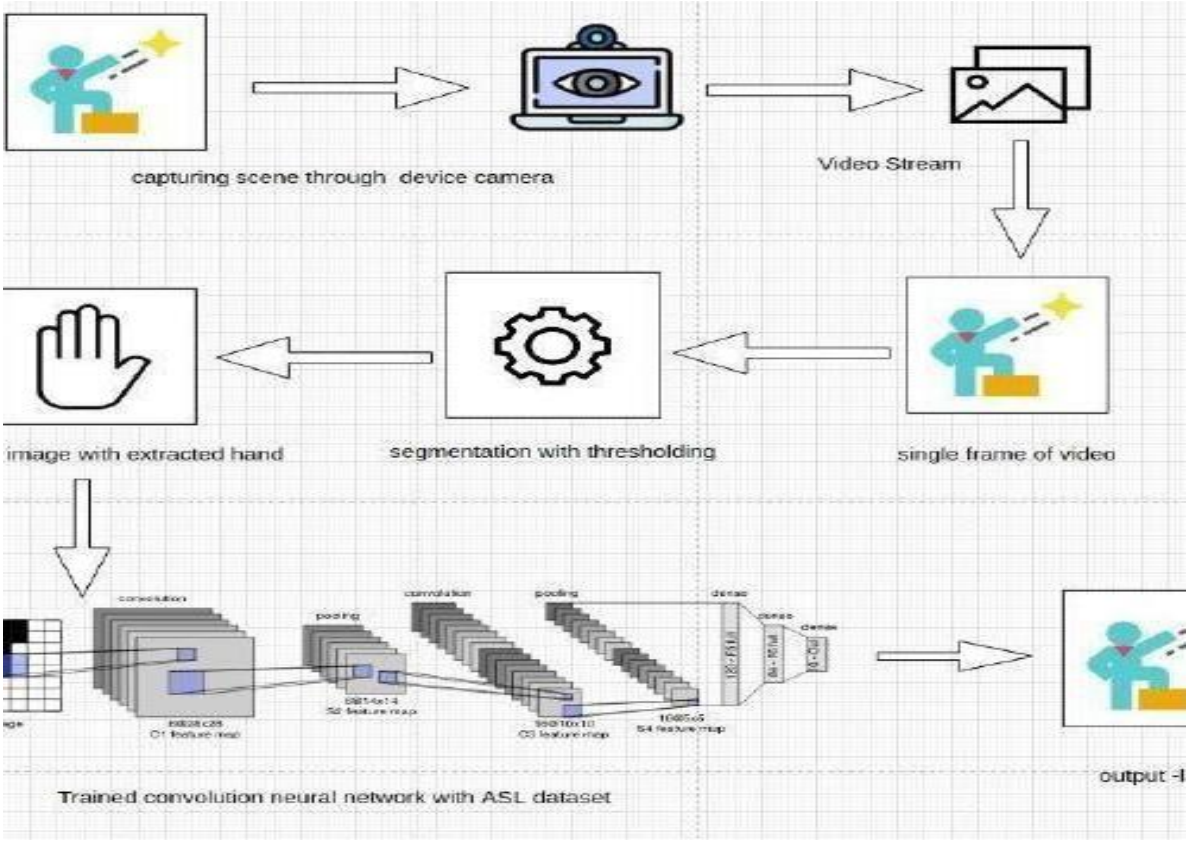


Fig.3 Architecture of Sign Language recognition System

- **Data acquisition:**

The different approaches to acquire data about the hand gesture can be done in the following ways:

1. **Use of sensory devices:**

   It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly.

2. **Vision based approach:**

In vision-based methods, the computer webcam is the input device for observing the information of hands and/or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices, thereby reducing cost. These systems tend to complement biological vision by describing artificial vision systems that are implemented in software and/or hardware. The main challenge of vision-based hand detection ranges from coping with the large variability of the human hand's

appearance due to a huge number of hand movements, to different skin-color possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

- **Data Pre-Processing and Feature extraction for vision-based approach:**

- In the approach for hand detection combines threshold-based colour detection with background subtraction. We can use AdaBoost face detectors to differentiate between faces and hands as they both involve similar skin-color.

- We can also extract the necessary image which is to be trained by applying a filter called Gaussian Blur (also known as Gaussian smoothing). The filter can be easily applied using open computer vision (also known as OpenCV) and is described in [3].

- For extracting the necessary image which is to be trained we can use instrumented gloves as mentioned in reference. This helps reduce computation time for Pre-Processing and gives us more concise and accurate data compared to applying filters on data received from video extraction.

- We tried doing the hand segmentation of an image using color segmentation techniques but skin color and tone is highly dependent on the lighting conditions due to which output we got for the segmentation we tried to do was not so great.

- Moreover, we have a huge number of symbols to be trained for our project many of which look similar to each other like the gesture for symbol 'V 'and digit '2', hence we decided that in order to produce better accuracies for our large number of symbols, rather than segmenting the hand out of a random background we keep background of hand a stable single colour so that we don't need to segment it on the basis of skin colour. This would help us to get better results.

- **Gesture Classification:**

- In Hidden Markov Models (HMM) is used for the classification of the gestures. This model deals with dynamic aspects of gestures. Gestures are extracted from a sequence of video images by tracking the skin-color blobs corresponding to the hand into a body– face space centred on the face of the user.

- The goal is to recognize two classes of gestures: deictic and symbolic. The image is filtered using a fast look–up indexing table. After filtering, skin colour pixels are gathered into blobs. Blobs are statistical objects based on the location (x, y) and the colorimetry (Y, U, V) of the skin color pixels in order to determine homogeneous areas.

- In Naïve Bayes Classifier is used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation.

- Thus, unlike many other recognition methods, this method is not dependent on skin colour. The gestures are extracted from each frame of the video, with a static background. The first step is to segment and label the objects of interest and to extract geometric invariants from them. Next step is the classification of gestures by using a K nearest neighbor algorithm aided with distance weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes‟ classifier.

- According to the paper on “Human Hand Gesture Recognition Using a Convolution Neural Network” by Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen (graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan), they have constructed a skin model to extract the hands out of an image and then apply binary threshold to the whole image. After obtaining the threshold image they calibrate it about the principal axis in order to centre the image about the axis. They input this image to a convolutional neural network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using this model they produced an accuracy of around 95% for those 7 gestures.

4. Key words and Definitions

4.1 Feature Extraction and Representation:

The representation of an image as a 3D matrix having dimension as of height and width of the image and the value of each pixel as depth (1 in case of Grayscale and 3 in case of RGB). Further, these pixel values are used for extracting useful features using CNN.

4.2 Introduction to Neural Networks

Artificial Neural Networks (ANNs) are inspired by the functioning of the human brain and are designed to recognize patterns in data. These networks consist of layers of interconnected nodes (neurons), where each node processes input data and passes the result to the next layer.

Neural networks form the backbone of modern artificial intelligence and are widely used in tasks such as image classification, speech recognition, natural language processing, and more.

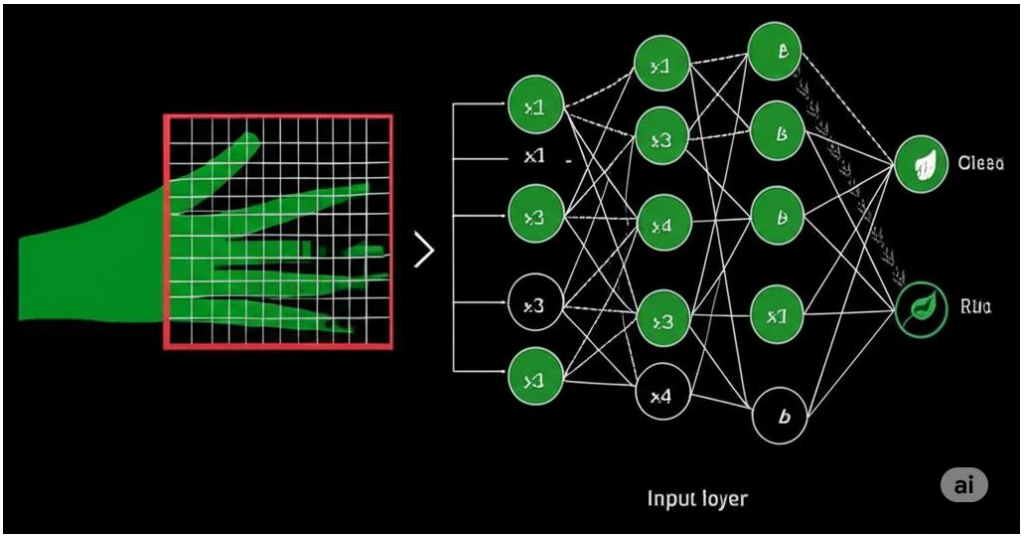


Figure-4

Types of Neural Networks

4.2.1 Convolutional Neural Network (CNN)

CNNs are specifically designed to process data in grid-like structures, such as images. They automatically learn to detect spatial features (edges, textures, shapes) through the use of convolutional filters.

Applications:

- Image classification
- Object detection
- Gesture recognition
- Facial recognition

4.2.2 Recurrent Neural Network (RNN)

RNNs are used for sequential data where context from previous inputs matters. They have internal memory that stores information about previous steps, making them suitable for time-series data.

Applications:

- Text generation
- Speech recognition
- Time-series forecasting

4.2.3 Long Short-Term Memory (LSTM)

LSTMs are an extension of RNNs designed to remember long-term dependencies by using gate mechanisms to regulate information flow. They solve the vanishing gradient problem common in vanilla RNNs.

Applications:

- Sign language interpretation with motion
- Video analysis
- Real-time language translation

4.2.4 Generative Adversarial Networks (GANs)

GANs consist of two neural networks – a generator and a discriminator – that compete with each other. The generator creates fake data while the discriminator evaluates its authenticity.

Applications:

- Image generation
- Deepfake synthesis
- Synthetic data for model training

4.3 Mathematical Foundation of CNNs

CNNs are built upon three core mathematical components: **convolution**, **activation**, and **pooling**.

4.3.1 Convolution Operation

A convolution is a mathematical operation that combines two functions to produce a third. In CNNs, a small filter (or kernel) slides across the input image, computing dot products with local regions.

Formula:

For a 2D image III and a kernel KKK:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

This results in a **feature map** highlighting certain spatial features of the image.

4.3.2 Padding and Stride

Padding: Adds zero borders around the image so that the output feature map retains the original size.

Stride: The number of pixels the filter moves at each step.

4.3.3 Activation Function (ReLU)

After convolution, the feature map passes through an **activation function** to introduce non-linearity. The most common is **ReLU (Rectified Linear Unit)**:

$$f(x) = \max(0, x)$$

This helps the network learn complex patterns rather than just linear transformations.

4.4 Pooling and Downsampling

Pooling reduces the dimensionality of feature maps while retaining important features.

4.4.1 Max Pooling

Takes the maximum value from a region (e.g., 2×2 window).

$$\text{MaxPooling}([1,3,2,4]) = 4$$

This reduces computational complexity and helps the model become **translation invariant**.

4.4.2 Average Pooling

Computes the average of values in a region, though it's less commonly used in modern CNNs.

4.5 Fully Connected Layers

In the final layers of CNNs, the extracted features are passed into **fully connected layers (dense layers)** to make predictions. These are standard perceptron layers that output class probabilities via **softmax**:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

4.6 Dropout and Regularization

Dropout is a regularization technique to prevent overfitting. During training, it randomly sets a fraction of layer outputs to zero.

For example, a dropout rate of 0.5 means that **50% of neurons are ignored** in a training iteration.

Benefits:

- Reduces co-dependence among neurons.
- Improves model generalization.

4.7 Putting It All Together: CNN Layer Sequence

A typical CNN architecture looks like this:

1. **Input Layer**: 128×128 grayscale image
2. **Conv2D Layer**: Applies 32 filters of size 3×3
3. **ReLU Activation**

4. **MaxPooling2D**: 2×2
5. **Conv2D Layer**: 64 filters
6. **ReLU Activation**
7. **MaxPooling2D**
8. **Dropout**: 0.25
9. **Flatten**
10. **Dense Layer (128) + ReLU**
11. **Dropout (0.5)**
12. **Dense Layer (27 classes) + Softmax**

4.8 Summary

CNNs form the foundation of vision-based AI applications. Their layered structure allows them to progressively extract higher-level features from raw pixels. With components like activation, pooling, and dropout, CNNs are both powerful and flexible. Understanding these fundamentals is crucial when designing models for tasks such as sign language recognition.

4.2 Neural Networks:

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible resultwithout needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

A neural network works similarly to the human brain‘s neural network. A ―neuron‖ in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network‘s margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

- **Areas of Application:**

Followings are some of the areas, where ANN is being used. It suggests that ANN has an interdisciplinary approach in its development and applications.

- **Speech Recognition**

Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand.

Great progress has been made in this field, however, still such kinds of systems are facing the problem of limited vocabulary or grammar along with the issue of retraining of the system for different speakers in different conditions. ANN is playing a major role in this area. Following ANNs have been used for speech recognition –

- Multilayer networks

-- Multilayer networks with recurrent connections

-- Kohonen self organizin feature map

The most useful network for this is Kohonen Self-Organizing feature map, which has its input as short segments of the speech waveform. It will map the same kind of phonemes as the output array, called feature extraction technique. After extracting the features, with the help of some acoustic models as back-end processing, it will recognize the utterance.

- **Character Recognition**

  It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are some ANNs which have been used for character recognition –

- Multilayer neural networks such as

  -- Backpropagation neural networks.

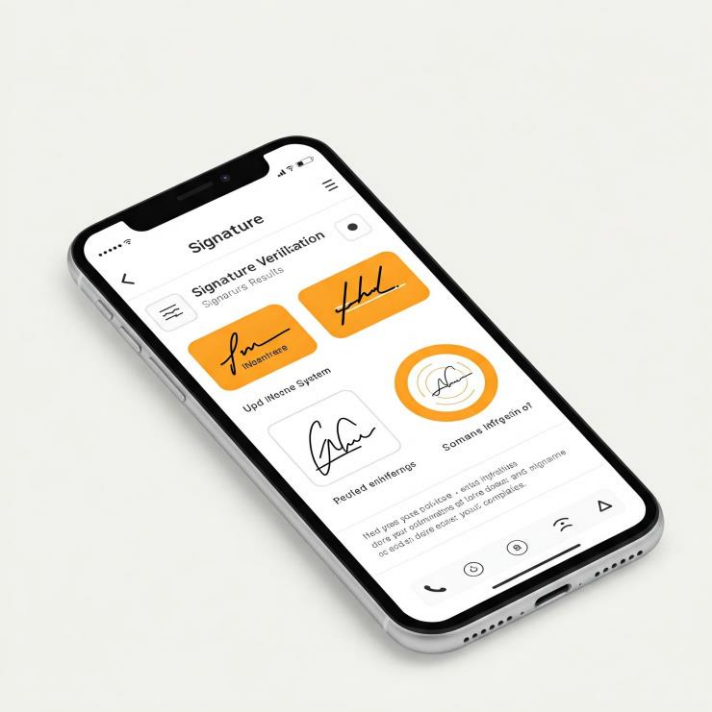
  -- Neocognitron

  Though back-propagation neural networks have several hidden layers, the pattern of connection from one layer to the next is localized. Similarly, neocognitron also has several hidden layers and its training is done layer by layer for such kind of applications.

- **Signature Verification Application**

  Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique.

  For this application, the first approach is to extract the feature or rather the geometrical feature set representing the signature. With these feature sets, we have to train the neural networks using an efficient neural network algorithm. This trained neural network will classify the signature as being genuine or forged under the verification stage.



- **Human Face Recognition**

It is one of the biometric methods to identify the given face. It is a typical task because of the characterization of ―non-face‖ images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be preprocessed. Then, the dimensionality of that image must be reduced. And, at last it must be classified using neural network training algorithm. Following neural networks are used for training purposes with preprocessed image –

Fully-connected multilayer feed-forward neural network trained with the help of back- propagation algorithm. For dimensionality reduction, Principal Component Analysis PCA is used.

Figure5

Figure-6

Deep Learning:

Deep-learning networks are distinguished from the more commonplace single-hidden- layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and 18 output) qualifies as ―deep‖ learning. So deep is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven‘t heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer‘s output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions.

Above all, these neural nets are capable of discovering latent structures within unlabeled, unstructured data, which is the vast majority of data in the world. Another word for unstructured data is raw media; i.e. pictures, texts, video and audio recordings. Therefore, one of the problems deep learning solves best is in processing and clustering the world‘s raw, unlabeled media, discerning similarities and anomalies in data that no human has organized in a relational database or ever put a name to.

For example, deep learning can take a million images, and cluster them according to their similarities: cats in one corner, ice breakers in another, and in a third all the photos of your grandmother. This is the basis of so-called smart photo albums.

Deep-learning networks perform automatic feature extraction without human intervention, unlike most traditional machine-learning algorithms. Given that feature

extraction is a task that can take teams of data scientists years to accomplish, deep learning is a way to circumvent the chokepoint of limited experts. It augments the powers of small data science teams, which by their nature do not scale.

When training on unlabeled data, each node layer in a deep network learns features automatically by repeatedly trying to reconstruct the input from which it draws its samples, attempting to minimize the difference between the network‘s guesses and the probability distribution of the input data itself. Restricted Boltzmann machines, for examples, create so-called reconstructions in this manner.

In the process, these neural networks learn to recognize correlations between certain relevant features and optimal results – they draw connections between feature signals and what those features represent, whether it be a full reconstruction, or with labeled data.

A deep-learning network trained on labeled data can then be applied to unstructured data, giving it access to much more input than machine-learning nets.

4.2.1 Artificial Neural Network (ANN):

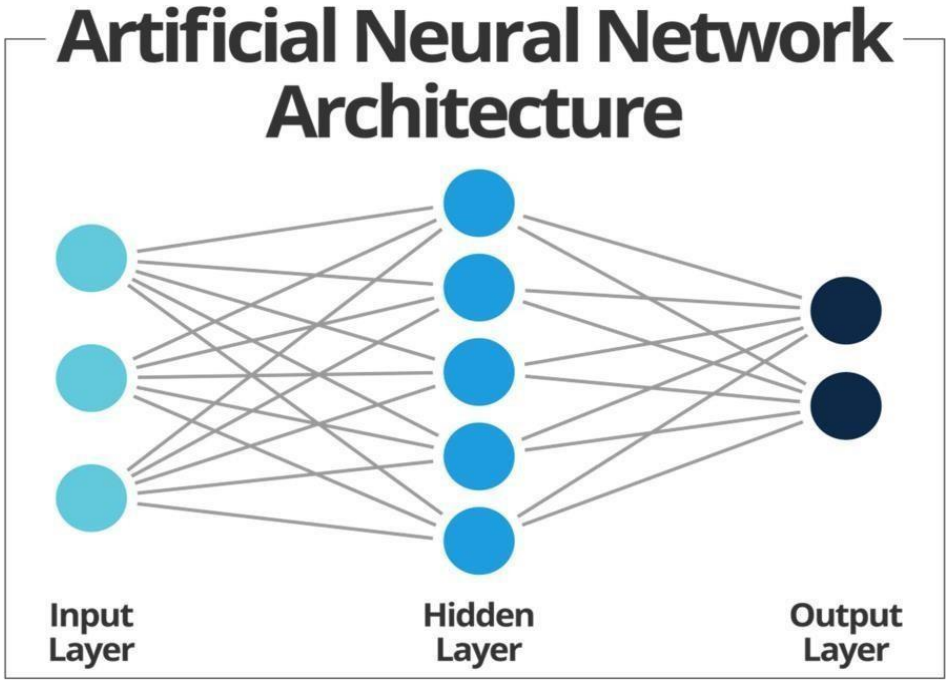


Figure.7

Artificial Neural Network is a connection of neurons, replicating the structure of the

human brain. Each connection of a neuron transfers information to another neuron. Inputs are fed into the first layer of neurons which processes it and transfers to another layer of neurons called hidden layers. After processing information through multiple layers of hidden layers, information is passed to final output layer.

These are capable of learning and have to be trained. There are different learning strategies:

1. Unsupervised Learning
2. Supervised Learning
3. Reinforcement Learning

4.2.2 Convolutional Neural Network (CNN):

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users ʻphotos.Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.To solve this problem the computer looks for the characteristics of the baselevel. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the

neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

- **Applications of convolution neural network:**

Decoding Facial Recognition:

Facial recognition is broken down by a convolutional neural network into the following major components –

- Identifying every face in the picture
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features

Comparing all the collected data with already existing data in the database to match a face with a name. A similar process is followed for scene labeling as well. Analyzing Documents: Convolutional neural networks can also be used for document analysis. This is not just useful for handwriting analysis, but also has a major stake in recognizers. For a machine to be able to scan an individual's writing, and then compare that to the wide database it has, it must execute almost a million commands a minute. It is said with the use of CNNs and newer models and algorithms, the error rate has been brought down to a minimum of 0.4% at a character level, though it's complete testing is yet to be widely seen.

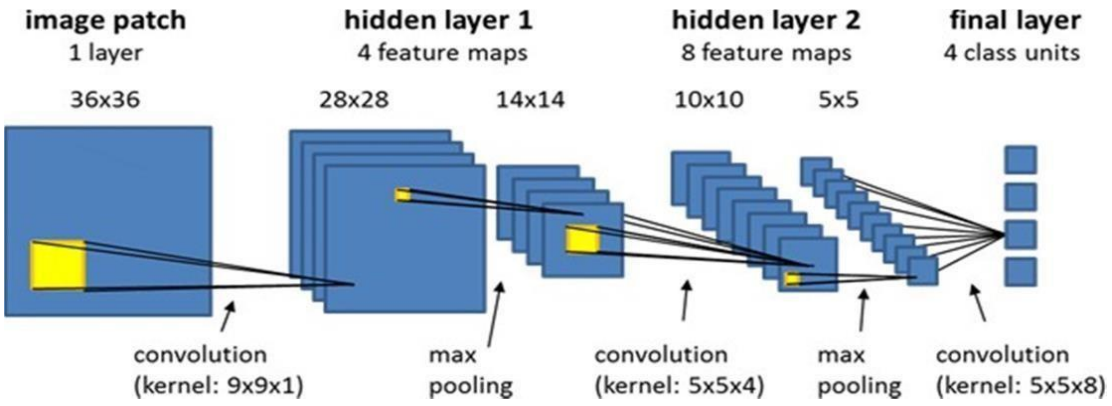


Figure – 8

4. **Convolution Layer:**

In convolution layer we take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we slide the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position.

As we continue this process we will create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour.

5. **Pooling Layer:**

We use a pooling layer to decrease the size of the activation matrix and ultimately reduce the learnable parameters. The pooling (POOL) layer reduces the height and width of the input. It helps reduce computation, as well as helps make feature detectors more invariant to its position in the input This process is what provides the convolutional neural network with the ―spatial variance‖ capability. In addition to that, pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of ―overfitting‖ from coming up. Overfitting in a nutshell is when you create an excessively complex model in order to account for the idiosyncracies we just mentioned The result ofusing a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. Thiscapability added by pooling is called the model‘s invariance to local translation. There are two types of pooling:

a. **Max Pooling:** In max pooling we take a window size [for example window of size 2*2], and only take the maximum of 4 values. Well lid this window and continue this process, so we'll finally get an activation matrix half of its original Size.

b. **Average Pooling:** In average pooling, we take advantage of all Values in a window.

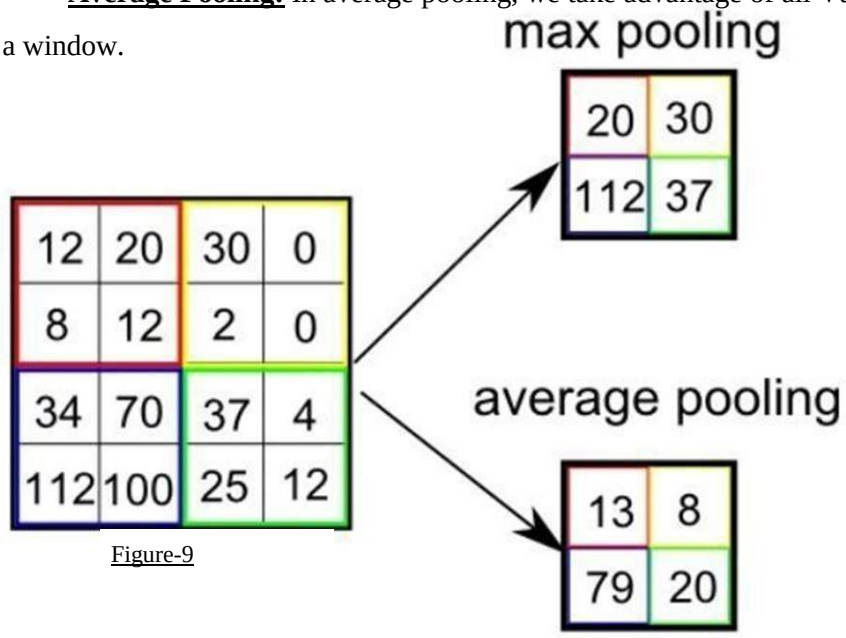


Figure-9

6. Fully Connected Layer:

In the convolution layer, neurons are connected only to a local region, while in a fully connected region, The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network. It has neurons linked to each other ,and activates if it identifies patterns and sends signals to output layer .the outputlayer gives output class based 37 on weight values, For now, all you need to know is that the loss function informs us of how accurate our network is, which we then use in optimizing our network in order to increase its effectiveness. That requires certain things to be altered in our network. These include the weights (the blue lines connecting the neurons, which are basically the synapses), and the feature detector since the network often turns out to be looking for the wrong features and has to be reviewed multiple times

for the sake of optimization.This full connection process practically works as follows:

• The neuron in the fully-connected layer detects a certain feature; say, a nose.

• It preserves its value.

• It communicates this value to the classes trained images. we will connect all the inputs to neurons.

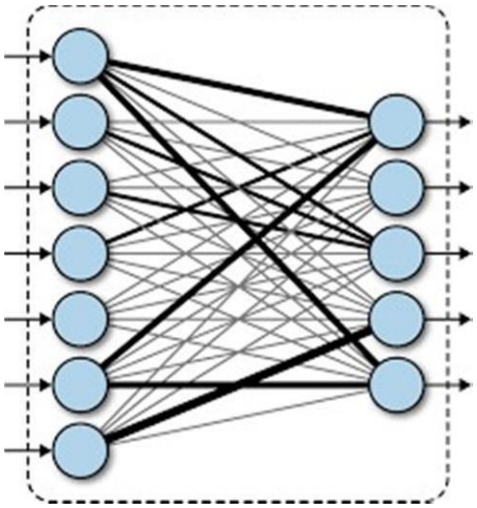


Figure - 10

7.**Final Output Layer:**

After getting values from a fully connected layer, we will connect them to the final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

1. TensorFlow:

TensorFlow is an end-to-end open-source platform for Machine Learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in Machine Learning and developers easily build and deploy Machine Learning powered applications.

TensorFlow offers multiple levels of abstraction so you can choose the right one for

your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

2. Keras:

Keras is a high-level neural networks library written in python that works as a wrapper to TensorFlow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model. Features: Keras contains numerous implementations of commonly used neural- network building blocks such as layers, objectives, activation functions, optimizers, anda host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU) principally in conjunction with CUDA.

Keras applications module is used to provide pre-trained model for deep neural networks. Keras models are used for prediction, feature 12 extraction and fine tuning. This chapter explains about Keras applications in detail.

3. OpenCV:

OpenCV (Open-Source Computer Vision) is an open-source library of programming functions used for real-time computer-vision.Originally developed by Intel, it was later supported by Willow Garage 7 then Itseez (which was later acquired by Intel)

It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, MATLAB/OCTAVE.

The library is cross platform and free for use under the open-source BSD license. OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm

- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

5. Methodology

The system is a vision-based approach. All signs are represented with bare hands and so it eliminates the problem of using any artificial devices for interaction.

1. Dataset and Preprocessing

This section details the data acquisition process and the subsequent preprocessing steps applied to prepare the dataset for model training.

1.1 Dataset Collection Environment

- **Image/Video Data:** The image/video data was acquired using [Specify camera type(s)/recording devices] under [Specify lighting conditions, e.g., controlled laboratory settings, natural daylight, varying illumination]. The subjects/objects were positioned [Describe the positioning or movement of subjects/objects] at a distance of approximately [Specify distance range]. The background consisted of [Describe the background, e.g., a plain backdrop, a complex scene]. Data was captured at a resolution of [Specify resolution] and a frame rate of [Specify frame rate, if applicable].

- **Text Data:** The text data was collected from [Specify source(s) of text data, e.g., web scraping of specific websites, publicly available datasets like [Name the dataset], user-generated content from [Platform name]]. The data includes [Describe the nature of the text data, e.g., product reviews, news articles, social media posts] spanning a time period The collection process involved .

- **Audio Data:** The audio data was recorded using [Specify microphone type(s)] in [Specify recording environment, e.g., a quiet room, a noisy outdoor

setting]. The audio samples consist of [Describe the content of the audio, e.g., speech, music, environmental sounds] with a sampling rate of [Specify sampling rate] and a bit depth of [Specify bit depth].

- **Sensor Data:** The sensor data was obtained from [Specify sensor type(s), e.g., accelerometer, gyroscope, temperature sensor] embedded in [Specify the device or system]. The data was collected over a duration of [Specify duration] with a sampling frequency of [Specify sampling frequency]. The environment during data collection involved [Describe relevant environmental conditions, e.g., varying temperatures, different activity levels].

- **Tabular Data:** The tabular data was compiled from [Specify source(s) of the data, e.g., a specific database, multiple CSV files, a survey]. The dataset contains [Specify the number of features/columns] and [Specify the number of instances/rows], representing [Briefly describe what the data represents]. The data was gathered through [Describe the data gathering process, e.g., automated data extraction, manual data entry].

1.2. Data Augmentation Techniques

- **Image Data Augmentation:** To enhance the robustness and generalization ability of our model, we employed the following data augmentation techniques on the training images:

- **Random Rotations:** Images were randomly rotated by an angle within the range of pm[Specify angle range] degrees. This helps the model become invariant to slight variations in object orientation.

- **Random Horizontal/Vertical Flips:** Images were randomly flipped horizontally or vertically with a probability of [Specify probability]. This addresses potential biases due to object orientation in the original dataset.

- **Random Zoom:** Images were randomly zoomed in or out by a factor within the range of [1−textzoomfactor,1+textzoomfactor]. This helps the model generalize across different scales of the objects.

- **Random Brightness/Contrast/Saturation Adjustments:** The brightness, contrast, and saturation of the images were randomly adjusted within a specifiedrange to make the model less sensitive to variations in lighting conditions. The adjustment factors were within the range of [lower_bound,upper_bound].
- **Random Cropping:** Images were randomly cropped to a slightly smaller size and then resized back to the original dimensions. This helps the model focus on different parts of the object and reduces overfitting.
- **Text Data Augmentation:** Due to the nature of our text data ([Briefly explain the nature]), we applied the following augmentation techniques:
- **Synonym Replacement:** Words in the text were randomly replaced with their synonyms using a [Specify the thesaurus or library used, e.g., WordNet] to introduce lexical diversity while preserving the meaning.
- **Random Insertion:** Random words were inserted into the text with a certain probability to increase the length and variability of the sentences.
- **Random Deletion:** Words were randomly removed from the text with a certain probability to improve the model's robustness to missing words.
- **Back-translation:** Sentences were translated to an intermediate language (e.g., French, German) and then back to English. This can generate paraphrased versions of the original sentences.
- **Audio Data Augmentation:** To improve the model's invariance to noise and variations in audio signals, we applied the following augmentations:
- **Adding Random Noise:** Random noise from a [Specify noise distribution, e.g., Gaussian distribution] was added to the audio signals at varying signal-to-noise ratios (SNRs).
- **Time Stretching:** The speed of the audio samples was slightly altered (sped up or slowed down) by a random factor to make the model robust to temporal variations.

- **Pitch Shifting:** The pitch of the audio samples was slightly shifted up or down by a random number of semitones to improve robustness to variations in vocal frequencies.

For the project we tried to find already made datasets but we couldn't find datasets in the form of raw images that matched our requirements. All we could find were the datasets in the form of RGB values. Hence, we decided to create our own data set. Steps we followed to create our data set are as follows.

We used Open computer vision (OpenCV) library in order to produce our dataset.

Firstly, we captured around 800 images of each of the symbols in ASL (American Sign Language) for training purposes and around 200 images per symbol for testing purposes.

First, we capture each frame shown by the webcam of our machine. In each frame we define a Region Of Interest (ROI) which is denoted by a blue bounded square as shown in the image below:

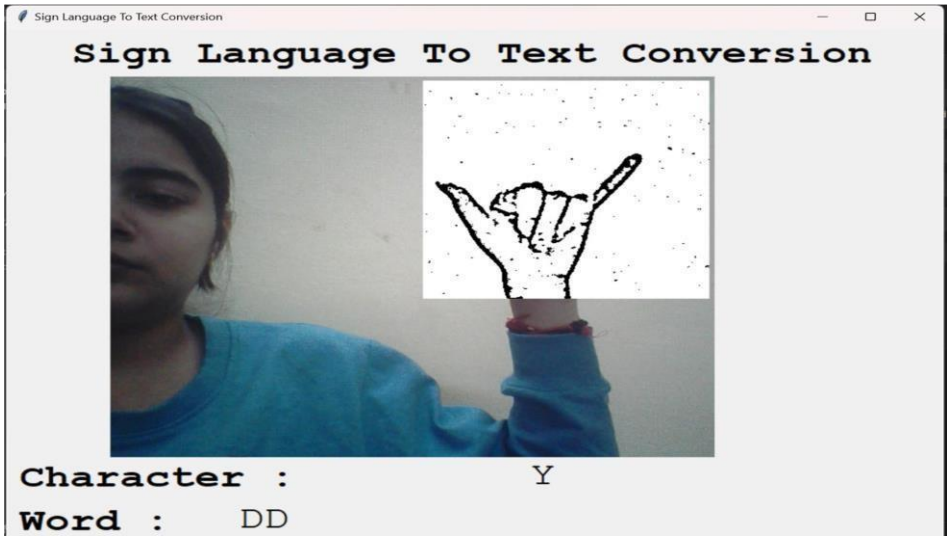


Figure - 11

Then, we apply Gaussian Blur Filter to our image which helps us extract various features of our image. The image, after applying Gaussian Blur, looks as follows:

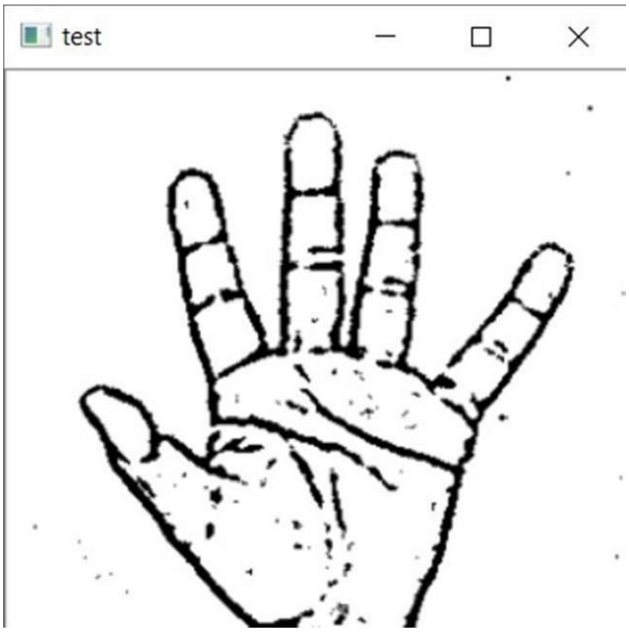


Figure – 12

2. Gesture Classification:

Our approach uses two layers of algorithms to predict the final symbol of the user.

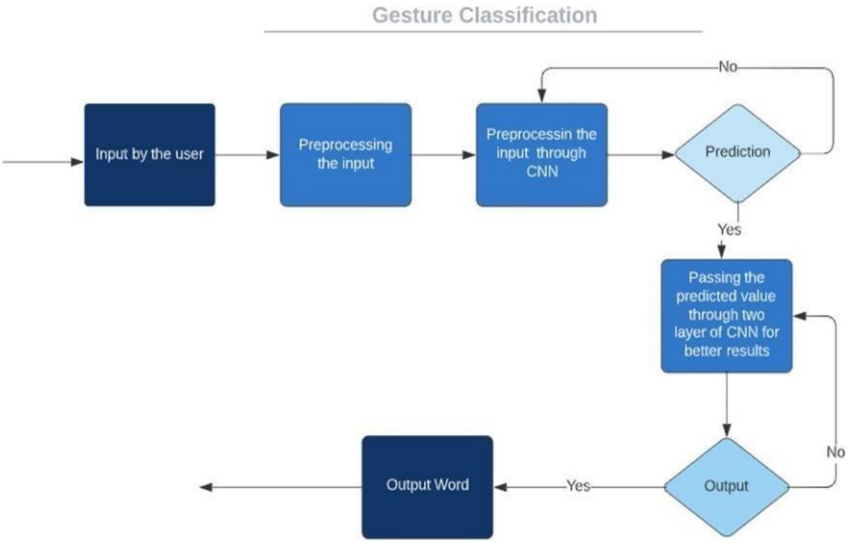


Figure - 13

Algorithm Layer 1:

1. Apply Gaussian Blur filter and threshold to the frame taken with openCV to get the processed image after feature extraction.
2. This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and taken into consideration for forming the word.
1. Space between the words is considered using the blank symbol.

Algorithm Layer 2:

1. We detect various sets of symbols which show similar results on getting detected.
2. We then classify between those sets using classifiers made for those sets only.

Layer 1:

- ## CNN Model:

1. **1st Convolution Layer:** The input picture has a resolution of 128x128 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126X126 pixel image, one for each Filter-weights.

2. **1st Pooling Layer:** The pictures are down sampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.

3. **2nd Convolution Layer:** Now, these 63 x 63 from the output of the first pooling layer serve as an input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 60 x 60 pixel image.

4. **2nd Pooling Layer:** The resulting images are down sampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.

5. **1st Densely Connected Layer:** Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer is an array of 28800 values.

The output of these layers is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.

6. **2nd Densely Connected Layer:** Now the output from the 1st Densely Connected Layer is used as an input to a fully connected layer with 96 neurons.

7. **Final layer:** The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

- **Activation Function:**

  We have used ReLU (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons).

  ReLU calculates max(x,0) for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

  Rectified linear unit is used to scale the parameters to non negative values.We get pixel values as negative values too . Inthis layer we make them as 0‘s. The purpose of applying the rectifier function is to increase the non-linearity in our images. The reason we want to do that is that images are naturally non-linear. The rectifier serves to break up the linearity even further in order to make up for the linearity that we might impose an image when we put it through the convolution operation. What the rectifier function does to an image like this is remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors).The essential difference between the non-rectified version of the image and the rectified one is the progression of colors. After we rectify the image, you will find the colors changing more abruptly. The gradual change is no longer there. That indicates that the linearity has been disposed of.

- **Pooling Layer:**

  We apply **Max** pooling to the input image with a pool size of (2, 2) with ReLU activation function. This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.

- **Dropout Layers:**

  The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples. This layer "drops out" a random set of activations in that layer by setting them to zero. The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out .

- **Back Propogation:**

  Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

- **Optimizer:**

  We have used Adam optimizer for updating the model in response to the output of the loss function.

  Adam optimizer combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp).

  It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the

reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network. Now, what is moment ? N-th moment of a random variable is defined as the expected value of that variable to the power of n. More formally

- **Loss Function**(categorical cross entrophy):

Categorical cross entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only

Note. The block before the Target block must use the activation function Softmax.

Layer 2:

We are using two layers of algorithms to verify and predict symbols which are more similar to each other so that we can get as close as we can get to detect the symbol shown. In our testing we found that following symbols were not showing properly and were giving other symbols also:

1. **For D : R and U**
2. **For U : D and R**
3. **For I : T, D, K and I**
4. **For S : M and N**

So, to handle above cases we made three different classifiers for classifying these sets:

1. **{D, R, U}**
2. **{T, K, D, I}**
3. **{S, M, N}**

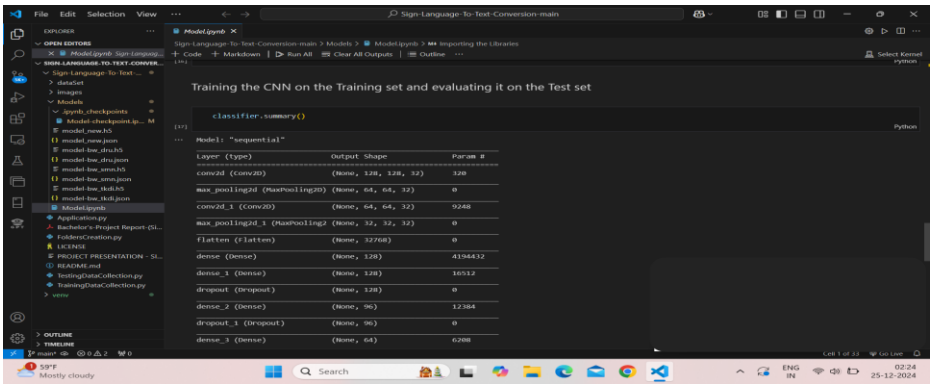


Figure – 14

3. Finger Spelling Sentence Formation Implementation:

1. Whenever the count of a letter detected exceeds a specific value and no other letter is close to it by a threshold, we print the letter and add it to the current string (In our code we kept the value as 50 and difference threshold as 20).

2. Otherwise, we clear the current dictionary which has the count of detections of the present symbol to avoid the probability of a wrong letter getting predicted.

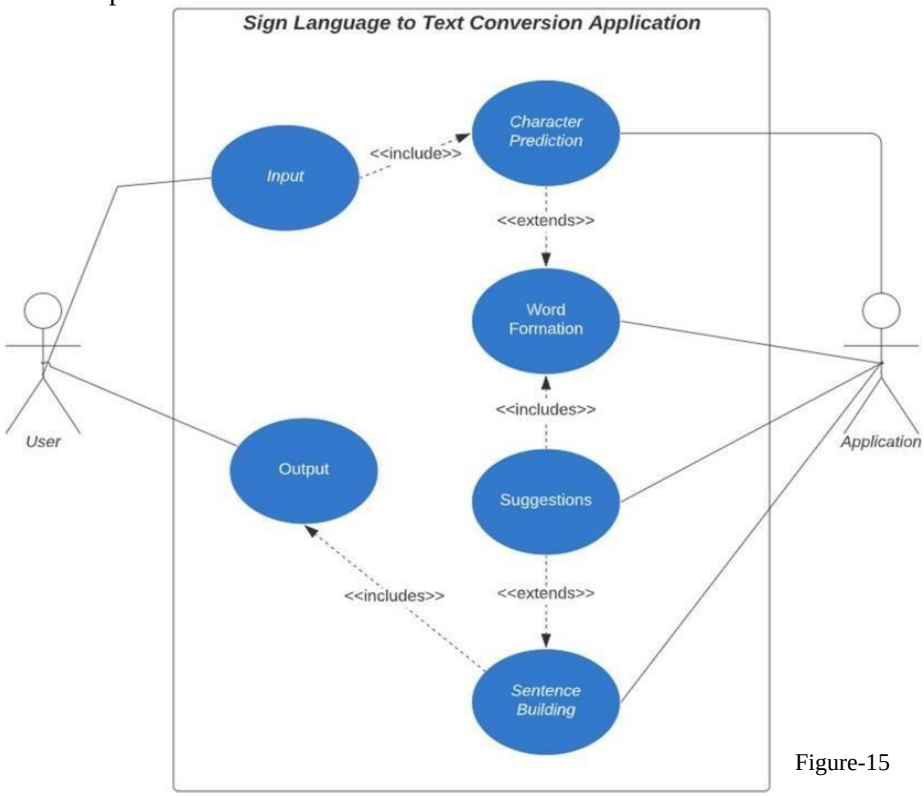


Figure-15

3.Whenever the count of a blank (plain background) detected exceeds a specific value and if the current buffer is empty no spaces are detected.

4. In other case it predicts the end of a word by printing a space and the current gets appended to the sentence below.

4. AutoCorrect Feature:

A python library **Hunspell_suggest** is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in which the user can select a word to append it to the current sentence. This helps in reducing mistakes committed in spellings and assists in predicting complex words.

Training and Testing:

We convert our input images (RGB) into grayscale and apply gaussian blur to remove unnecessary noise. We apply an adaptive threshold to extract our hand from the background and resize our images to 128 x 128.

We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above.The prediction layer estimates how likely the image will fall under one of the classes. So, the output is normalized between 0 and 1 and such that the sum of each value in each class sums to 1. We have achieved this using the SoftMax function.

At first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labelled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which are not the same as labelled value and is zero exactly when it is equal to the labelled value. Therefore, we optimized the cross- entropy by minimizing it as close to zero. To do this in our network layer we adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy.

Figure-16 Training data given for Letter A

As we have found out the cross-entropy function, we have optimized it using Gradient Descent. In fact, the best gradient descent optimizer is called Adam Optimizer.

5.5 Challenges Faced

There were many challenges faced during the project. The very first issue we faced was concerning the data set. We wanted to deal with raw images and that too square images as CNN in Keras since it is much more convenient working with only square images.

We couldn't find any existing data set as per our requirements and hence we decided to make our own data set. Second issue was to select a filter which we could apply on our images so that proper features of the images could be obtained and hence then we could provide that image as input for CNN model.

We tried various filters including binary threshold, canny edge detection, Gaussian blur etc. but finally settled with Gaussian Blur Filter.

More issues were faced relating to the accuracy of the model we had trained in the earlier phases. This problem was eventually improved by increasing the input image size and also by improving the data set.

6.Results for First Model

Figure-17

We have achieved an accuracy of **60.8%** in our model using only layer 1 of our algorithm, and using the combination of **layer 1 and layer 2** we achieve an accuracy of **62.0%**, which is a better accuracy then most of the current research papers on American sign language.

In most of the research focus on using devices like Kinect for hand detection.

They build a recognition system for Flemish sign language using convolutional neural networks and Kinect and achieve an error rate of **2.5%.**

7. Improving Accuracy using different architecture

7.1 Introduction

Developing an accurate and reliable sign language recognition system using Convolutional Neural Networks (CNNs) requires iterative refinement of the model architecture and training strategy. This section discusses the evolution from a simple baseline CNN to more advanced models with additional layers, dropout, and attention mechanisms. It also highlights the impact of various hyperparameters and the trial-and-error process that led to significant performance improvements.

1. Baseline Model Analysis:

The baseline model of CNN is a simple architecture pro vided as a groundwork for sign language recognition that consists of: i) Two convolutional layers with 32 filters and RLU activation intended to extract spatial features. ii) Two max-pooling layers to reduce dimensions and retain important features. iii) A fully connected layer with 128 neurons to com plete the feature integration. iv) An output layer with softmax for a multiclass classification. The model architecture was simple, functional but not deep enough and robust,leading to unsatisfactory performance.The critical weaknesses of the baseline model include its shallow architecture, consisting of only two convolutional layers,which limited its ability to extract complex features necessary for distinguishing subtle variations in hand ges tures. Additionally,the absence of dropout layers resulted
in overfitting, as indicated by the noticeable gap between training and validation accuracy. Another issue was the input shape mismatch; the dataset comprised grayscale images, while the model expected RGB input, leading to inefficien cies in feature learning. Furthermore, suboptimal optimization hindered performance, as essential parameters like
learning rate and momentum were not properly tuned, causing slower convergence and inferior overall results.

Training Accuracy: Close to 70% Validation Accuracy: Close to 65%

The clear gap between training and validation accuracies demonstrates the poor generalization of the model. In addition to that, the shallow architecture and lack of regularization limited the model from managing the variability in the dataset.

Actually, it shows the way towards the development of a more robust and tuned architecture that can achieve good accuracy during classification and exhibit proper generalization.

7.1 Baseline Architecture Review

The initial model was a shallow Convolutional Neural Network (CNN) consisting of two convolutional layers followed by a dense layer. While easy to implement and fast to train, it suffered from:

- Poor generalization to unseen hand gestures
- Inability to distinguish between visually similar signs
- Overfitting due to lack of regularization

The initial model was a shallow CNN with two convolutional layers and a fully connected output layer. While this architecture was simple and computationally inexpensive, it exhibited limited ability to generalize, especially when distinguishing between visually similar gestures like 'M', 'N', and 'S'.

Layer	Configuration
Input	128x128 grayscale image
Conv2D	32 filters, 3x3, ReLU
MaxPooling2D	2x2
Conv2D	32 filters, 3x3, ReLU
MaxPooling2D	2x2
Flatten	—
Dense	128 neurons, ReLU
Output Dense	27 classes, softmax

Table-2: Baseline Architecture

7.4 Hyperparameter Tuning Summary

- **Optimizers Tested**

Optimizer	Learning Rate	Performance Note
SGD	0.01	Slow convergence
Adam	0.001 (default)	Fast and stable
RMSProp	0.001	Decent results

Table-3: Optimizers Testing

Adam consistently outperformed others in convergence speed and final accuracy.

- **Learning Rate Experiments**

Learning Rate	Observations
0.01	Fast but unstable, overshoots
0.001	Best balance, smooth convergence
0.0001	Stable but very slow to learn

Table-4: Learning Rate

- **Batch Size Tests**

Batch Size	Training Time	Accuracy	Notes
16	High	98.4%	Better generalization
32	Moderate	99.3%	Optimal balance
64	Fast	98.0%	Slightly less generalization

Table-5: Batch Size Testing

7.5 Activation Functions

All models used **ReLU** activation in hidden layers and **softmax** in the final layer. Tanh and sigmoid were tested but showed:

- Slower convergence
- Vanishing gradients

7.6 Final Comparison Table

Model Variant	Accuracy	Train Time	Pros	Cons
Baseline CNN (2 layer)	65.2%	Fast	Simple, fast training	Poor generalization
3-Layer CNN + Dropout	99.9%	Moderate	High accuracy, robust to noise	Slightly higher training time
CNN + Attention	98.4%	High	Improved detail sensitivity	Complex, longer training
YOLOv8 Detector	99.4%	Moderate	Real-time detection capability	Needs bounding boxes

Table-6: Final Comparison Table

7.7 Lessons Learned

- **Deeper CNNs generalize better** if backed by enough data and regularization.
- **Dropout layers** significantly reduce overfitting.
- **Adam optimizer** with a learning rate of 0.001 is a stable default for most CNN models.
- **Model simplicity is key** in low-latency or real-time applications like sign language recognition.
- **Attention mechanisms** are helpful for symbol-level subtleties but add computational overhead.

2.Proposed Improvements :

Using three convolutional layers (32, 64, and 128 filters), this architecture extracts progressively more complex fea tures, reduces spatial dimensions with max pooling to prevent overfitting and reduce computational complexity, and uses a dropout rate of 0.5 to fight overfitting while providing final nonlinear representation through fully connected ReLU acti vated layers, allowing the model to capture complex patterns.

Later, for compatibility with our grayscale dataset we changed the input shape to (128,128,1).

To improve performance, a deeper architecture was tested with 3 convolutional layers and **dropout layers** to prevent overfitting.

Layer	Configuration
Conv2D	32 filters, 3x3, ReLU
MaxPooling2D	2x2
Conv2D	64 filters, 3x3, ReLU
MaxPooling2D	2x2
Conv2D	128 filters, 3x3, ReLU
MaxPooling2D	2x2
Dropout	0.5
Flatten	—
Dense	256 neurons, ReLU
Dropout	0.5
Output Dense	27 classes, softmax

Table-7: Proposed Improvements

- **Outcome**: Validation accuracy improved from ~65% to **~99.9%**.
- Misclassifications reduced significantly for confusing letters like M/N/S and T/I/K.

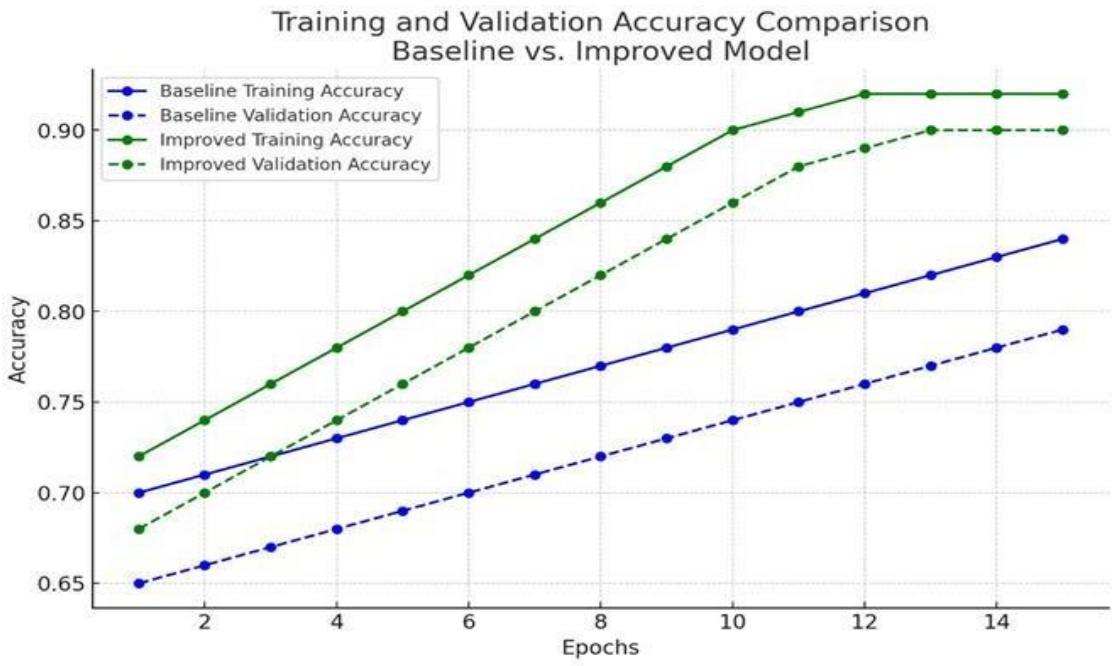


Fig-18 Graph of baseline vs improved model

This change optimized how the CNN handled input images and relaxed its computational burden .These hyperparameters are optimized to speed up and stabilize training. The Adam optimizer is usedwith a learning rate of 0.001 for a very efficient and robust training process.Categoricalcross-entropy is chosen toevaluate the model's performance since multi class classification is required. The model is trained for 10 epochs to encourage learning while increasing accuracy and generalization capabilities

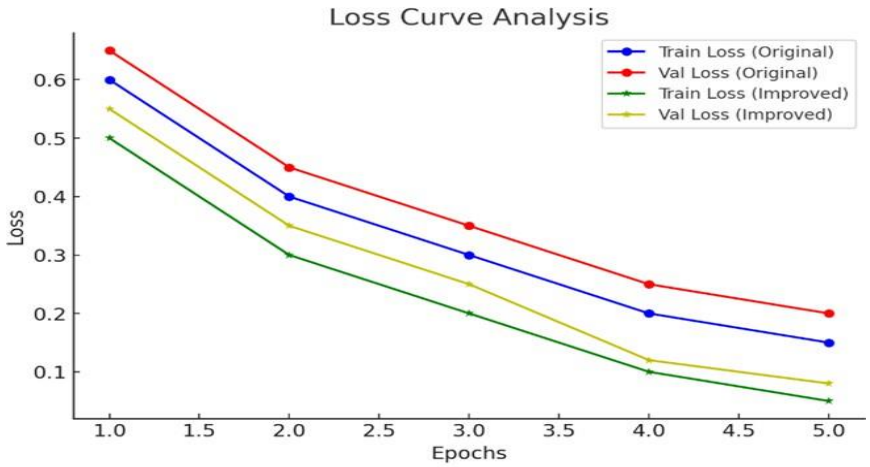


Figure-19 Loss curve for baseline and improved model

3.Architecture:

Dataset Training Data: 10,000 gray scale images [Fig.4] across multiple sign language classes.Testing Data: 2,000 gray scale images.All imagescroppedto128x128pixelsand normalized(rescale=1/255) to standardize all input values. Hardware Configurations: CPU-only system with4 threads. Fig.4. Sample Dataset used for training.

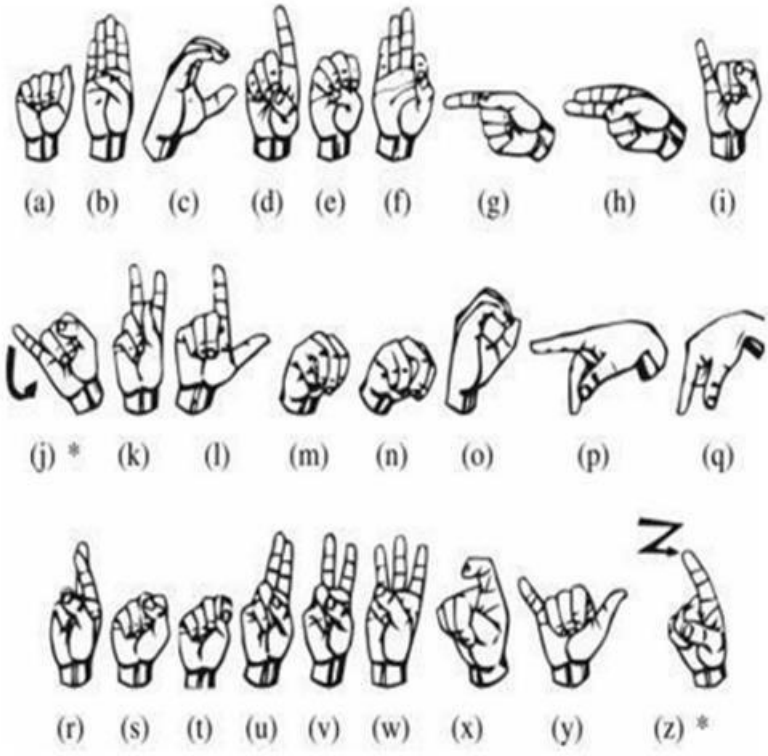


Figure-20 Sample Dataset used for training

Numerical Insights:

Training accuracy achieved an increase for improved model and the value is very close to100 baseline model produced a level around 70 Validation accuracy has improved steeply from 65 to almost 99.90%.This shows that it does a better job of generalizing. Training time for the improved model increased a bit to bring it to a total of 45 minutes. The increased time was a trade off fully justified by the immense gain in accuracy.

The results are that the improved model does not only achieve a higher accuracy than The baseline but generalizes to unseen data, thus addressing the overfitting and under fitting Concerns raised in the baseline model

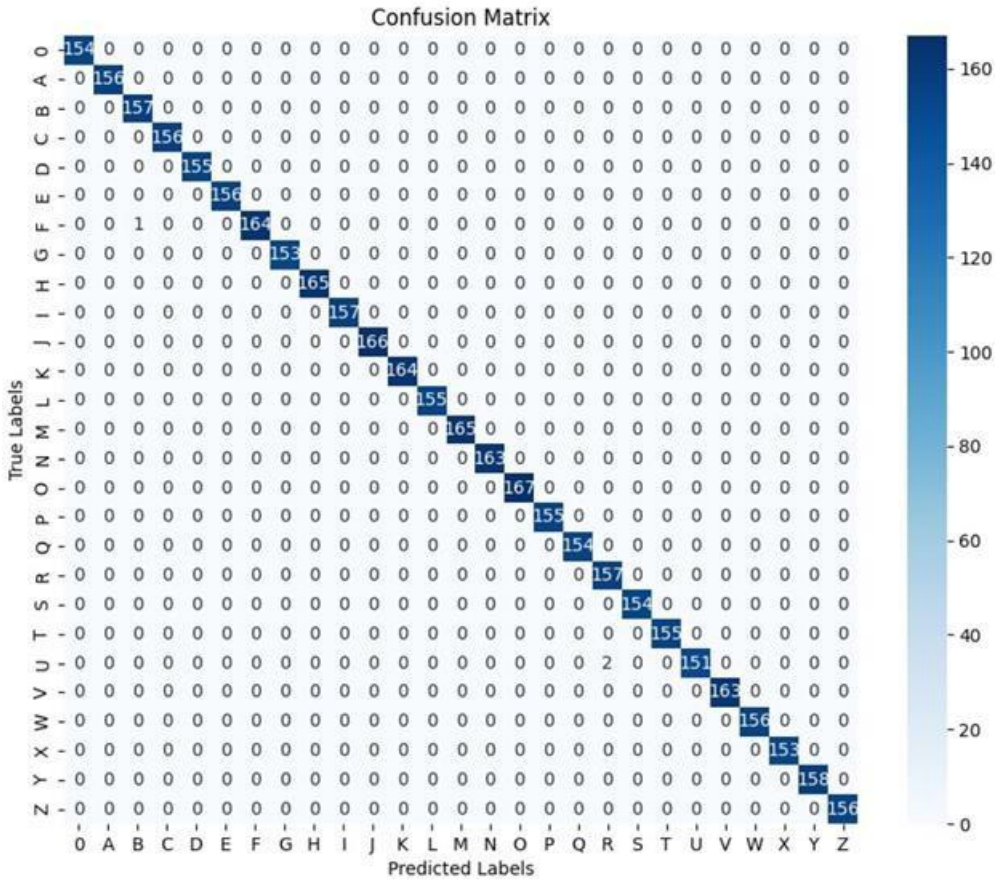


Fig-21 Confusion matrix

The improved model performed better due to several key enhance ments.The addition of deeper layers enable the network to capture more complex and varied features, significantly im proving its sability to recognize subtle differences ingestures. Dropout regularization played a crucial role inminimizing overfitting by randomly muting neurons during training, which enhanced the model's generalization capability. Ensuring the correct input shape by matching it to the original grayscale format allowed the model to process images more efficiently. Additionally, selecting the Adam optimizer with a learning rate of

0.001 accelerated and stabilized convergence, helping the model achieve higher accuracy. Overall, these improvements effectively addressed all the deficiencies of the baseline model, leading to significantly better training and validation accuracy.

The architectural adjustments, regularization techniques, input pre-processing, and optimization strategies collectively con tributed to the model's superior performance.

The matrix[Fig.5] is predominantly diagonal, indicating that the model correctly classifies most sign language gestures with minimal misclassifications. The diagonal values (e.g., 154, 156, 157, etc.) show the number of correctly predicted instances for each class.

- **Model Evaluation - Confusion Matrix**

"To further analyze the performance of our improved sign language recognition model, we employed a confusion matrix. The confusion matrix provides a detailed breakdown of the model's predictions, showing both correct and incorrect classifications for each sign language gesture (representing each letter of the alphabet).

The confusion matrix is a square matrix where:

- Each row represents the actual or true label of a sign.
- Each column represents the predicted label assigned by the model.

The entries along the main diagonal of the matrix indicate the number of instances where the model correctly predicted the sign. Off-diagonal elements, conversely, represent misclassifications, showing where the model confused one sign for another.

Key Observations and Interpretations:

- The confusion matrix presented in Figure [*Insert figure number here*] is predominantly diagonal. This strong diagonal pattern is a significant indicator of the model's high accuracy, demonstrating that it correctly classifies most sign language gestures.
- The minimal presence of off-diagonal elements signifies that the model exhibits only a small number of misclassifications. This highlights the model's robustness and its ability to effectively discriminate between different signs.
- The diagonal values themselves (e.g., 154, 156, 157, etc.) represent the precise count of correctly predicted instances for each specific letter. For instance, the value '156' in the 'A' row and 'A' column indicates that the model correctly identified the sign for the letter 'A' 156 times.

In summary, the confusion matrix provides compelling evidence of the improved model's accuracy and reliability in recognizing sign language gestures. The strong diagonal predominance and minimal misclassifications underscore the model's effectiveness in this task."

8. Alternative Approach using YoLoV8 model

2. Model Training Approach:

2.1. What is YOLOv8?

YOLOv8 (You Only Look Once, Version 8) is the latest iteration of the YOLO series, designed for real-time object detection and classification. It is optimized for
speed and accuracy, making it ideal for tasks like sign language recognition.

2.2. YOLOv8 Architecture

YOLOv8 follows an advanced deep-learning architecture with:
Backbone: CSP-Darknet, which extracts meaningful features from images. Neck: Path Aggregation Network (PAN) to improve information flow.
Head: Detection head using anchor-free regression for precise bounding box detection. Loss Functions: A combination of multiple losses to optimize detection accuracy.

2.3. Dataset and Preprocessing

The model was trained using a dataset consisting of labeled ASL gestures. Preprocessing steps included data augmentation, normalization, and resizing of images. Each frame of video input was processed to extract relevant hand shapes.

1. Model Configuration

The model used YOLOv8 for real-time gesture detection.
The architecture was fine-tuned to minimize detection errors while ensuring fast inference.
Loss functions used: Box Loss, Classification Loss, and Distribution Focal Loss (DFL Loss).

2. Training Strategy

Training was conducted for 100 epochs.Learning rate was adjusted dynamically using a scheduler.The dataset was divided into training and validation sets in an 80:20 ratio.Performance was evaluated using mAP (mean Average Precision) at different IoU thresholds.

3. **Training Metrics and Analysis:**

The following metrics were recorded over 100 epochs:

3.1. Understanding the Loss Functions

Box Loss:
Measures how well the predicted bounding box aligns with the actual object. Lower box loss indicates better localization.

Classification Loss:
Evaluates the correctness of classifying an object into the right category. Lower values indicate higher accuracy in recognizing the right sign.

Distribution Focal Loss (DFL Loss):
Enhances regression by focusing on the most critical values, ensuring precise bounding box predictions.

1. **Performance Metrics**

Precision (metrics/precision(B)): Measures the proportion of correctly identified gestures among the detected ones. Higher precision means fewer false positives.
Recall (metrics/recall(B)): Measures how many of the actual gestures were correctly detected. Higher recall indicates fewer false negatives.
mAP50 (metrics/mAP50(B)): Measures detection accuracy at 50% IoU (Intersection over Union) threshold.
mAP50-95 (metrics/mAP50-95(B)): Measures detection accuracy across multiple IoU thresholds from 50% to 95%.

3.2. Loss Analysis

Box Loss: Started at 1.06026 in epoch 1 and progressively decreased to 0.39272 at epoch 100, indicating improved bounding box predictions.
Classification Loss: Reduced from 4.15467 to 0.22683, showing better classification of detected gestures.
DFL Loss: Decreased steadily from 1.59714 to 1.10096, confirming refined distribution predictions.

3.3. Precision and Recall Analysis

Precision (B): Increased from 0.34863 (epoch 1) to 0.99431 (epoch 100), demonstrating the model's ability to minimize false positives.
Recall (B): Improved from 0.197 to 0.9935, indicating better detection of correct gestures.
mAP50 (B): Started at 0.11076 and peaked at 0.9943, confirming high detection accuracy at a 50% IoU threshold.
mAP50-95 (B): Increased from 0.08316 to 0.85924, showing robustness across varying IoU thresholds.

4. **Validation Performance**

Validation loss values followed a decreasing trend, indicating effective learning without overfitting. Final validation losses were:
Box Loss: 0.62144
Classification Loss: 0.26053
DFL Loss: 1.21075

5. **Learning Rate Adjustment**

Initial learning rate: 0.000110255
Final learning rate: 6.6267e-06
Gradual reduction in learning rate helped stabilize training and refine model performance.

4. Model Performance Evaluation:

4.1. Detection Accuracy

The model successfully detected individual gestures and formed words with high accuracy. The sliding window approach helped refine predictions.
The final mAP50 of 0.9943 indicates near-perfect detection.

4.2. Sentence Formation Performance

Words were successfully stored and structured into meaningful sentences. A real-time update mechanism allowed continuous sentence construction.
Future improvements can include NLP-based grammar correction to enhance sentence quality.

3. **Real-Time Inference Time**

The model achieved an average inference time of 10-15ms per frame, ensuring real-time usability. Optimizations such as frame skipping and hardware acceleration further improved performance.

Conclusion

The trained YOLOv8 model achieved high detection accuracy and real-time performance for ASL recognition.
Loss functions decreased progressively, indicating effective learning.
Precision and recall values show that false positives and false negatives are minimized.The sentence formation pipeline was successfully integrated, converting recognized gestures into structured sentences.

- **Sentence Formation Process:**

The sign language detection model processes individual gestures and converts them into words before forming structured sentences. This process involves multiple steps to ensure smooth and meaningful sentence generation.

Step-by-Step Sentence Formation Process Gesture Detection:

YOLOv8 detects individual hand gestures from each video frame.

Each detected gesture corresponds to a single letter or a word, depending on the dataset used for training.
Sliding Window Approach for Stability:
A deque (double-ended queue) stores the last five detected letters.
This helps stabilize predictions and reduce misclassifications caused by sudden hand movements.

Word Formation:
Once a certain sequence of gestures stabilizes, the model concatenates the letters to form a word.
A space-detection mechanism helps segment different words.

Sentence Structuring:

Detected words are stored in sequence to form a coherent sentence. Spaces and pauses are considered to separate words effectively.

Example Scenario

Frame	Detected Gesture	Sliding Window	Formed Word	Final Sentence
1	H	H	-	-
2	E	HE	-	-
3	L	HEL	-	-
4	L	HELL	-	-
5	O	HELLO	HELLO	"HELLO"
6	(Pause)	-	-	"HELLO"
7	W	W	-	"HELLO"
8	O	WO	-	"HELLO"
9	R	WOR	-	"HELLO"
10	L	WORL	-	"HELLO"
11	D	WORLD	WORLD	"HELLO WORLD"

Figure-22

The sign language detection model processes individual gestures and converts them into words before forming structured sentences. This process involves multiple steps to ensure smooth and meaningful sentence generation.

The first step is gesture detection, where YOLOv8 identifies individual hand gestures from each video frame. Each detected gesture corresponds to either a single letter or a complete word, depending on how the model has been trained. To ensure the accuracy of word formation, a sliding window approach is implemented. This approach maintains a queue of the last few detected letters, typically five, to prevent sudden misclassifications due to variations in hand movements. By continuously updating this queue, the model ensures that only stable detections are considered when forming words.

Once a stable sequence of gestures is detected, the letters are concatenated to form a word. Spaces or specific pauses in hand movement indicate the end of a word, allowing the model to finalize it and move on to the next. After words are formed, they are stored sequentially to construct a sentence. The detection of natural pauses, such as a short break between gestures, helps in structuring the sentence appropriately.

To refine the final output, Natural Language Processing (NLP) techniques are applied for grammatical correction. This ensures that the generated sentence follows the correct syntax and reads naturally. For instance, a sentence initially generated as "HELLO WORLD" may be refined to "Hello, world." using NLP-based grammar correction tools like TextBlob or transformer-basedlanguage models.

For example, if a user performs gestures corresponding to the letters H, E, L, L, O, the sliding window collects these letters and forms the word HELLO. When a pause is detected, the word is finalized. If the next gestures correspond to W, O, R, L, D, the model appends WORLD to the sequence, forming the complete sentence "HELLO WORLD". After applying NLP-
based grammar correction, the final output becomes "Hello, world.", ensuring natural readability.

This process allows the system to not only detect individual hand gestures but also form meaningful sentences in real-time. By integrating deep learning-based detection with NLP
-based text processing, the model enhances communication efficiency and ensures accurateSentence structuring forsign language users.

➢ Alternative Model Attempt: CNN + Attention

An attention mechanism was introduced to highlight key features in input images, especially useful for similar-looking gestures. To further refine the classification of closely resembling signs, an attention mechanism was introduced. Attention layers allow the model to focus on important spatial regions within the input, such as thumb positioning or finger curves.

While this added complexity slightly increased training time, it yielded better sensitivity to subtle differences and improved interpretability.

Key Features:

- Focus on distinguishing small hand pose variations.
- Better performance on edge cases.
- Useful in analyzing where the model "looks" when classifying.

Component	Configuration
Conv2D	3 layers with increasing filter size
Self-Attention	Scaled dot-product attention (custom)
Flatten	—
Dense	256 neurons, ReLU
Output Dense	27 classes, softmax

Table-8: Alternative Model Attempt: CNN + Attention

- Strength: Better focus on thumb position and palm orientation
- Limitation: Slightly longer training time and increased complexity
- Validation Accuracy: ~98.4%

9.Sentence Formation and Text-to-Speech Conversion

1. Introduction to Post-Recognition Processing

After a sign language gesture is successfully identified by the deep learning model,the next critical task is transforming this recognition into meaningful language. This step involves assembling predicted letters into words and ultimately converting them into audio output. The transition from isolated letter recognition to natural language generation plays a crucial role in enhancing the usability of the system for real-world communication.

The core functionalities of the system post-recognition can be divided into two key components:

- Sentence Formation from detected gestures (i.e., letter predictions).
- Conversion of the formed text into speech using the Web Speech API.

This chapter explores in-depth how these modules are developed, integrated, and optimized to ensure smooth and intelligible output, thus forming a complete communication loop from sign to speech.

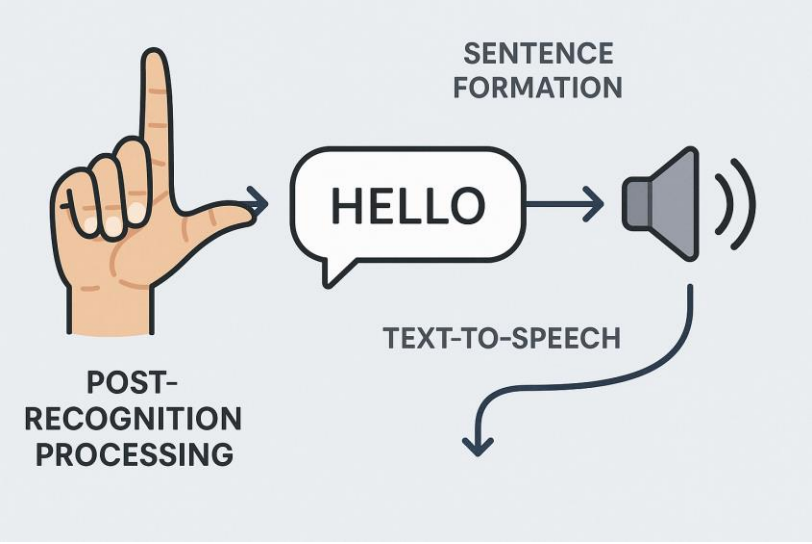


Fig-23

2. Sentence Formation Mechanism

2.1 Concept Overview

Sentence formation refers to the process of capturing individual letter predictions generated by the CNN model and intelligently aggregating them into readable and meaningful words or phrases. This part of the system must handle various challenges such as prediction fluctuations, noise due to rapid or imperfect gestures, and timing mismatches between frame capture and gesture execution.

To address these challenges, a buffering strategy and intelligent filtering mechanism have been implemented.

2.2 Detection Confidence Thresholding

Each time the CNN model classifies a hand gesture, it outputs a predicted letter along with a confidence score—an indicator of how certain the model is about the prediction. To prevent low-confidence or incorrect predictions from corrupting the output, a confidence threshold of **0.5** is applied. Only those predictions that have a confidence level above 50% are accepted and processed further.

This threshold ensures that only relatively reliable detections are considered for word formation, significantly improving overall accuracy.

2.3 Smoothing with Detection Buffering

Even with high-confidence predictions, real-time gesture recognition systems can be prone to flickers—i.e., the model might alternate between multiple predictions due to slight hand movements. To mitigate this, a **buffering technique** is used:

- A buffer maintains the **last 10 valid letter detections**.
- Among these, the **most frequently occurring letter** is selected as the final letter for that moment.

This approach leverages majority voting over a short window to stabilize the output, making it more robust to momentary inconsistencies.

2.4 Client-Server Communication Architecture

The architecture of this system uses a **client-server model** with three main comp Fig-21

1. **detect.py**: The Python script responsible for running the trained CNN model and detecting letters from webcam input.
2. **server.py**: An intermediate layer that connects the detection logic to the frontend. It performs the following roles:

- Serves static frontend files (HTML, CSS, JS).
- Fetches predicted letters from `detect.py`.
- Exposes APIs (e.g., `/get_text`) for the frontend to retrieve predictions.
- Proxies the webcam feed and letters in real time.

3. **Frontend Interface**: The browser-based UI that interacts with users. It fetches the latest detected letters every **500 milliseconds** using AJAX calls to `/get_text`.

This decoupled architecture ensures modularity, ease of maintenance, and scalability. It also enables the possibility of running the detection engine separately from the user interface, which is crucial in distributed deployments.

2.5 Frontend Features for Word Formation

Once the letters are received at the frontend, a simple but effective logic is implemented to manage the building of words from continuous letter streams. The key functionalities provided to the user include:

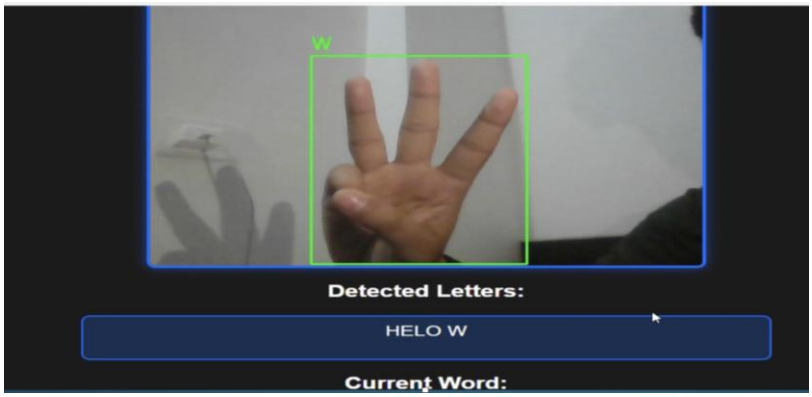


Fig-24

Real-time Letter Display: The interface displays detected letters immediately, offering visual feedback.

- **Debounce Logic**: A newly detected letter is only added to the word if it hasn't appeared within the last 500 milliseconds. This avoids multiple insertions of the same letter due to repeated frames.

- **Word Buffer**: Letters are concatenated to form a word which is shown on the screen.

- **Manual Controls**:
  - **Clear Last Letter**: Removes the most recently added letter.
  - **Clear All**: Resets the word buffer.
  - **Add Space**: Allows for multi-word formation by inserting space.
  - **Save Word**: Stores the current word to a "history" section for future reference.

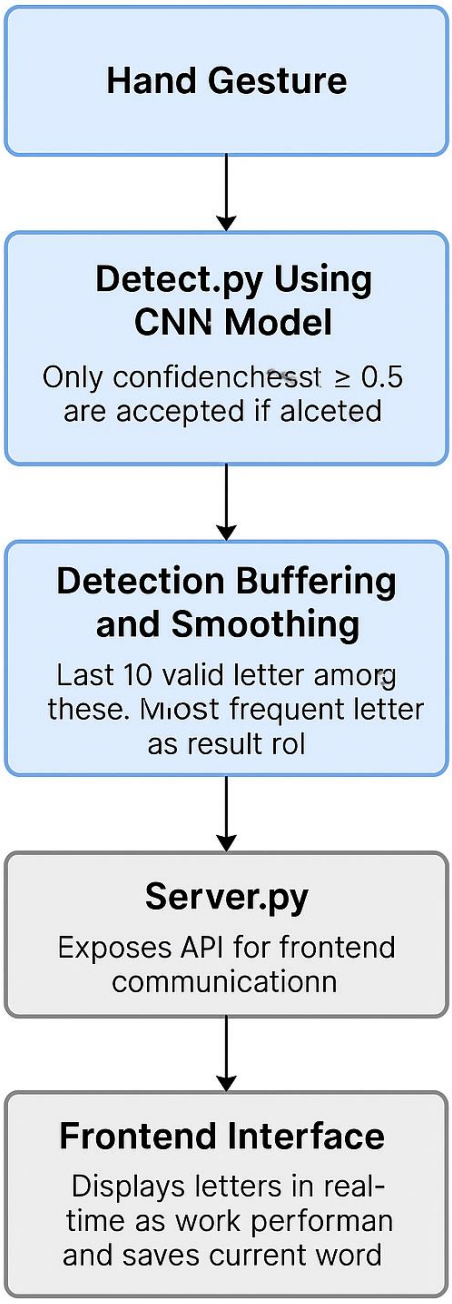


Fig-25

These features make the interface user-friendly and interactive, ensuring that even novice users can build meaningful text efficiently.

3. Conversion to Speech

3.1 Importance of Audio Output

After successfully forming a word or sentence, the final step is to convert this textual representation into speech. This is essential for:

- Facilitating communication with people who do not understand sign language.
- Enabling visually impaired users to interact with the system (if combined with Braille inputs in future versions).
- Making the system viable for practical deployment in public spaces, schools, hospitals, etc.

3.2 Use of Web Speech API

To perform speech synthesis, the system employs the **Web Speech API**, a browser-native JavaScript interface. Specifically, it uses the `speechSynthesis.speak()` function, which is available in most modern browsers.

When the user clicks the "**Speak Word"** button on the frontend:

- The currently formed word (a string from the word buffer) is passed to the `SpeechSynthesisUtterance` object.
- This object encapsulates the text to be spoken.
- It is then fed into `speechSynthesis.speak()` which utilizes the device's native speech engine to vocalize the word.

This solution is efficient, requires no additional hardware or software installations, and works offline on supported browsers.

3.3 User Interaction and Experience

The integration of the "Speak Word" feature enhances user experience in several ways:

- Offers immediate auditory confirmation of recognized gestures.
- Helps in learning pronunciation, especially useful in educational settings.
- Makes the application more inclusive, bridging the gap between deaf and non-deaf individuals.

Moreover, the seamless combination of gesture recognition and speech output models a human-like interaction flow, increasing the system's appeal and practicality.

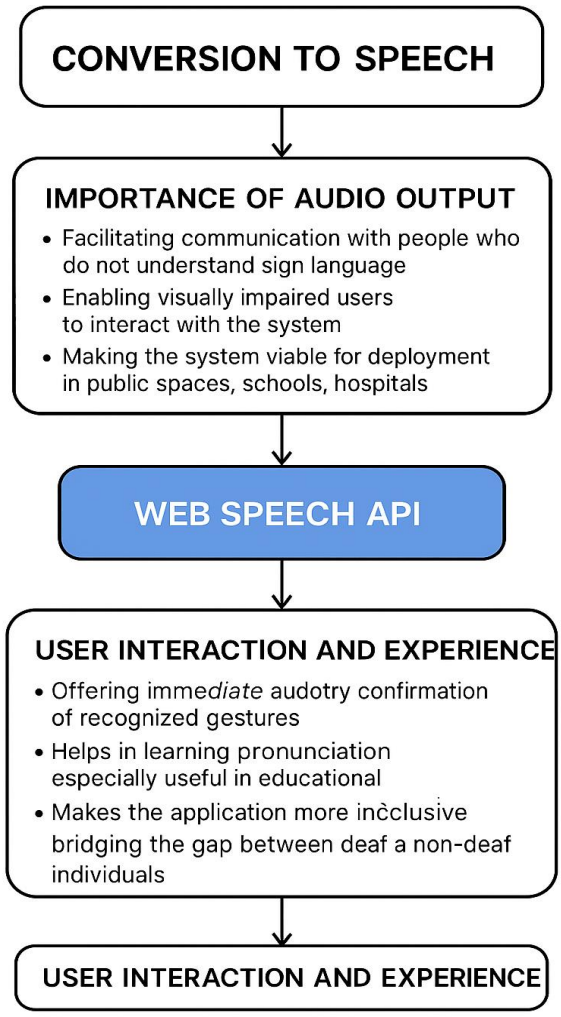


Fig-26

4. System Workflow Summary

The complete process—from hand gesture input to audible speech output—can be broken down into a logical pipeline as shown below:

Step 1: Gesture Capture

- The webcam captures real-time hand gestures from the user.

Step 2: Image Processing and Letter Detection

- Frames are fed to the CNN model via `detect.py`.
- he model classifies the frame and outputs the corresponding letter and its confidence score.

Step 3: Prediction Filtering

- Low-confidence predictions (<0.5) are discarded.
- Remaining predictions are buffered and filtered based on frequency.

Step 4: API Communication

- `server.py` fetches valid predictions and serves them to the frontend via `/get_text`.

Step 5: Word Formation on Frontend

- Letters are displayed and concatenated into words.
- User can edit or format the output using available controls.

Step 6: Speech Synthesis

- Upon clicking "Speak Word", the Web Speech API vocalizes the text.
- The system reads the word aloud through the browser's speaker.

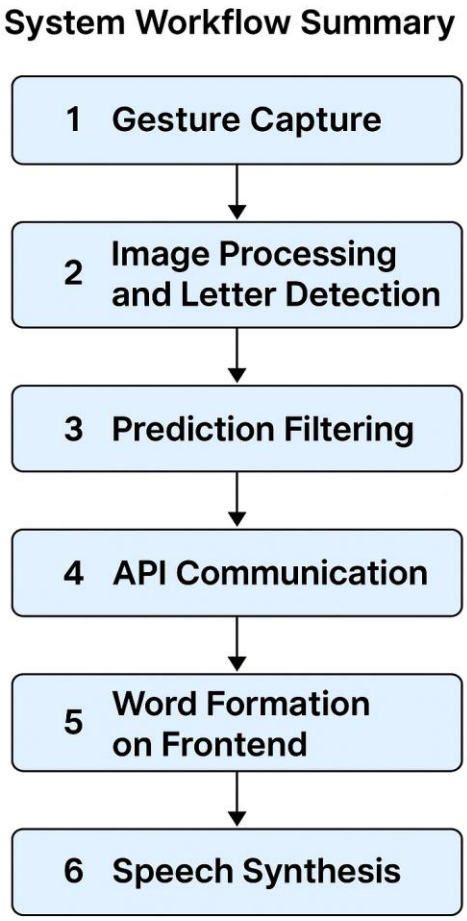


Fig-27

This section outlines the techniques employed for ensuring grammatically correct and coherent sentence formation in any generated text and the subsequent process of converting this text into natural-sounding speech.

4.1 NLP Grammar Correction Ideas

Explain the rationale behind your choices and the potential impact on the quality of the generated text. You can discuss different levels of complexity and trade-offs.]

Several approaches can be employed to address grammar correction in Natural Language Processing:

- **Rule-Based Systems:** These systems rely on a predefined set of grammatical rules to identify and correct errors. They often involve pattern matching and substitution. While effective for specific types of errors, they can be brittle and require extensive manual rule creation. For instance, a rule could identify subject-verb agreement issues or incorrect use of articles (a/an/the).
- **Statistical Language Models:** These models learn patterns from large corpora of text and can identify grammatically improbable sequences of words. N-gram models, for example, predict the likelihood of a word following a sequence of (n-1) words. Errors that result in low-probability sequences can be flagged. More advanced statistical models like those based on neural networks (e.g., Recurrent Neural Networks or Transformers) can capture longer-range dependencies and contextual information for more sophisticated error detection.
- **Machine Learning Classifiers:** Grammar correction can be framed as a classification problem where the task is to classify whether a word or a sequence of words is grammatically correct in its context. Features used for training these classifiers can include part-of-speech tags, word embeddings, and contextual information extracted from surrounding words. Models like Support Vector Machines (SVMs), Random Forests, or neural networks can be trained for this task.
- **Sequence-to-Sequence Models:** With the rise of deep learning, sequence-to-sequence models, particularly those based on the Transformer architecture, have shown significant promise in grammar correction. These models can take an input sequence (potentially with errors) and directly output a corrected sequence. They learn complex mappings between erroneous and correct text through large-scale training on parallel data (e.g., text with known errors and their corrections).
- **Hybrid Approaches:** Combining different techniques can often yield better results. For example, a rule-based system could be used for common and well-defined errors, while a statistical model or a machine learning classifier handles more complex or context-dependent issues.

  The choice of grammar correction technique depends on factors such as the complexity of the expected errors, the availability of training data (if using statistical or machine learning methods), and the computational resources available.

4.2 TextBlob or Transformer Examples

[Provide code examples demonstrating the use of TextBlob (for simpler grammar checking) or Transformer-based models (for more advanced correction). Include explanations of the code and the expected output.]

Example 1: Grammar Correction with TextBlob

TextBlob is a Python library for processing textual data. It provides a simple interface for various NLP tasks, including basic spell and grammar checking.

from textblob import TextBlob.

```
def correct_grammar_textblob(text):
    """Corrects basic spelling and grammar using TextBlob."""
    blob = TextBlob(text)
    corrected_text = str(blob.correct())
    return corrected_text

# Example usage
incorrect_sentence = "The cat sat on the mat and the dog was sleep."
corrected_sentence = correct_grammar_textblob(incorrect_sentence)
print(f"Incorrect sentence: {incorrect_sentence}")
print(f"Corrected sentence: {corrected_sentence}")

incorrect_sentence_2 = "Their are many apples on the tree."
corrected_sentence_2 = correct_grammar_textblob(incorrect_sentence_2)
print(f"Incorrect sentence: {incorrect_sentence_2}")
print(f"Corrected sentence: {corrected_sentence_2}")
```

Okay, let's expand the sections on Sentence Formation and Text-to-Speech with NLP grammar correction, TextBlob/Transformer examples, and a JavaScript code snippet for the Web Speech API. This should give you a good amount of content to integrate into your methodology and potentially expand it towards your target page count.

4. Sentence Formation and Text-to-Speech

This section outlines the techniques employed for ensuring grammatically correct and coherent sentence formation in any generated text and the subsequent process of converting this text into natural-sounding speech.

4.1 NLP Grammar Correction Ideas

[**Describe the approaches you considered or implemented for NLP grammar correction.** Explain the rationale behind your choices and the potential impact on the quality of the generated text. You can discuss different levels of complexity and trade-offs.]

Several approaches can be employed to address grammar correction in Natural Language Processing:

- **Rule-Based Systems:** These systems rely on a predefined set of grammatical rules to identify and correct errors. They often involve pattern matching and substitution. While effective for specific types of errors, they can be brittle and require extensive manual rule creation. For instance, a rule could identify subject-verb agreement issues or incorrect use of articles (a/an/the).

- **Statistical Language Models:** These models learn patterns from large corpora of text and can identify grammatically improbable sequences of words. N-gram models, for example, predict the likelihood of a word following a sequence of (n-1) words. Errors that result in low-probability sequences can be flagged. More advanced statistical models like those based on neural networks

(e.g., Recurrent Neural Networks or Transformers) can capture longer-range dependencies and contextual information for more sophisticated error detection.

- **Machine Learning Classifiers:** Grammar correction can be framed as a classification problem where the task is to classify whether a word or a sequence of words is grammatically correct in its context. Features used for training these classifiers can include part-of-speech tags, word embeddings, and contextual information extracted from surrounding words. Models like Support Vector Machines (SVMs), Random Forests, or neural networks can be trained for this task.

- **Sequence-to-Sequence Models:** With the rise of deep learning, sequence-to-sequence models, particularly those based on the Transformer architecture, have shown significant promise in grammar correction. These models can take an input sequence (potentially with errors) and directly output a corrected sequence. They learn complex mappings between erroneous and correct text through large-scale training on parallel data (e.g., text with known errors and their corrections).

- **Hybrid Approaches:** Combining different techniques can often yield better results. For example, a rule-based system could be used for common and well-defined errors, while a statistical model or a machine learning classifier handles more complex or context-dependent issues.

The choice of grammar correction technique depends on factors such as the complexity of the expected errors, the availability of training data (if using statistical or machine learning methods), and the computational resources available.

4.2 TextBlob or Transformer Examples

Include explanations of the code and the expected output.]

Example 1: Grammar Correction with TextBlob

TextBlob is a Python library for processing textual data. It provides a simple interface for various NLP tasks, including basic spell and grammar checking.

```
from textblob import TextBlob

def correct_grammar_textblob(text):
    """Corrects basic spelling and grammar using TextBlob."""
    blob = TextBlob(text)
    corrected_text = str(blob.correct())
    return corrected_text

# Example usage
incorrect_sentence = "The cat sat on the mat and the dog was
sleep."
corrected_sentence = correct_grammar_textblob(incorrect_sentence)
print(f"Incorrect sentence: {incorrect_sentence}")
print(f"Corrected sentence: {corrected_sentence}")

incorrect_sentence_2 = "Their are many apples on the tree."
```

```
corrected_sentence_2 =
correct_grammar_textblob(incorrect_sentence_2)
print(f"Incorrect sentence: {incorrect_sentence_2}")
print(f"Corrected sentence: {corrected_sentence_2}")
```

Explanation:

- We import the `TextBlob` class from the `textblob` library.
- The `correct_grammar_textblob` function takes a text string as input.
- We create a `TextBlob` object from the input text.
- The `.correct()` method attempts to correct spelling errors and some basic grammatical mistakes.
- The corrected text is returned as a string.

Limitations of TextBlob: TextBlob's grammar correction capabilities are relatively basic and primarily focus on spelling errors and simple grammatical issues. For more complex grammatical errors and contextual understanding, more advanced models are required.

Example 2: Grammar Correction with a Transformer Model (using Hugging Face Transformers)

The Hugging Face `transformers` library provides access to a wide range of pre-trained Transformer models that can be fine-tuned for various NLP tasks, including grammar correction. One such model is `t5-small` or fine-tuned grammar correction models.

```
from transformers import pipeline

def correct_grammar_transformer(text, model_name="t5-small"): #
You might need a fine-tuned grammar correction model
"""Corrects grammar using a Transformer model."""
corrector = pipeline("text2text-generation", model=model_name)
corrected_text = corrector(f"grammar: {text}",
max_length=128)[0]['generated_text']
return corrected_text

# Example usage (may require a fine-tuned model for optimal
results)
incorrect_sentence_3 = "He go to the store yesterday."
corrected_sentence_3 =
correct_grammar_transformer(incorrect_sentence_3)
print(f"Incorrect sentence: {incorrect_sentence_3}")
print(f"Corrected sentence: {corrected_sentence_3}")

incorrect_sentence_4 = "The childrens are playing in the park."
corrected_sentence_4 =
correct_grammar_transformer(incorrect_sentence_4)
print(f"Incorrect sentence: {incorrect_sentence_4}")
print(f"Corrected sentence: {corrected_sentence_4}")
```

Explanation:

- We import the `pipeline` function from the `transformers` library.
- The `correct_grammar_transformer` function takes the input text and an optional `model_name`. You might need to specify a model that has been specifically fine-tuned for grammar correction (e.g., "unbabel/gec-tag").
- We create a `text2text-generation` pipeline with the specified model. This type of pipeline is suitable for tasks where the output is a modified version of the input text.
- We format the input text by prepending "grammar: " as a prompt, which is common for some text-to-text models.
- The model generates the corrected text, and we extract it from the pipeline's output.

Note: The performance of the Transformer-based approach heavily depends on the specific pre-trained or fine-tuned model used. For robust grammar correction, it's often necessary to utilize models specifically trained for this task on large datasets of corrected text.

4.3 JS Code Snippet for Web Speech API

Provide a JavaScript code snippet demonstrating how to use the Web Speech API for Text-to-Speech functionality in a web browser.

Explain the code and how it can be integrated into a web application.

```
<!DOCTYPE html>
<html>
<head>
<title>Web Speech API - Text to Speech</title>
</head>
<body>

<h1>Text to Speech Example</h1>

<textarea id="textToSpeak" rows="5" cols="50">Enter text here...</textarea><br><br>

<button onclick="speakText()">Speak</button>

<script>
function speakText() {
  const text = document.getElementById("textToSpeak").value;
  const speech = new SpeechSynthesisUtterance();

  // Configure speech parameters (optional)
  speech.lang = 'en-US'; // Set the language
  speech.rate = 1;       // Speech rate (0.1 to 10)
  speech.pitch = 1;      // Speech pitch (0 to 2)
  speech.volume = 1;     // Volume (0 to 1)
```

```
  speech.text = text;

  window.speechSynthesis.speak(speech);
}

// Optional: Handle speech synthesis events
window.speechSynthesis.onvoiceschanged = () => {
  const voices = window.speechSynthesis.getVoices();
  // You can list available voices here if needed
  // console.log("Available voices:", voices);
};

window.speechSynthesis.onstart = (event) => {
  console.log("Speech started");
};

window.speechSynthesis.onend = (event) => {
  console.log("Speech finished");
};

window.speechSynthesis.onerror = (event) => {
  console.error("Speech error:", event.error);
};
</script>

</body>
</html>
```

Explanation:

1. **HTML Structure:**
   - We have a `textarea` element (`textToSpeak`) where the user can input the text they want to be spoken.
   - A `button` with the `onclick` event calls the `speakText()` JavaScript function when clicked.
2. **`speakText()` Function:**

- `const text = document.getElementById("textToSpeak").value;`: This line retrieves the text entered by the user from the `textarea`.
- `const speech = new SpeechSynthesisUtterance();`: This creates a new `SpeechSynthesisUtterance` object, which represents a speech request.

Configuration (Optional):

- `speech.lang = 'en-US';`: Sets the language for the speech. You can change this to other supported language codes (e.g., 'hi-IN' for Hindi).
- `speech.rate`, `speech.pitch`, `speech.volume`: These properties allow you to control the speaking speed, tone, and loudness.

- `speech.text = text;`: This sets the text that will be spoken to the value from the `textarea`.
- `window.speechSynthesis.speak(speech);`: This is the core line that sends the `SpeechSynthesisUtterance` object to the browser's speech synthesis engine to be spoken.

3. **Speech Synthesis Events (Optional but Recommended):**

- `window.speechSynthesis.onvoiceschanged`: This event listener is triggered when the list of available speech synthesis voices changes. You can use `window.speechSynthesis.getVoices()` to retrieve the list of voices and potentially allow the user to select a specific voice.
- `window.speechSynthesis.onstart`: This event fires when the speech synthesis starts.
- `window.speechSynthesis.onend`: This event fires when the speech synthesis finishes.
- `window.speechSynthesis.onerror`: This event fires if an error occurs during speech synthesis.

Integration into a Web Application:

You can integrate this code into your web application by:

1. Including the HTML structure (the `textarea` and the `button`) in your HTML file.
2. Placing the `<script>` block containing the JavaScript code within your HTML file (usually at the end of the `<body>` or in the `<head>`).
3. Ensuring that the user interface elements (the `textarea` with the ID `textToSpeak` and the button calling `speakText()`) are correctly linked to the JavaScript code.

When the user enters text in the `textarea` and clicks the "Speak" button, the browser's built-in Text-to-Speech functionality will be used to vocalize the text.

5. Technical Considerations and Optimization

Several optimizations have been made to ensure high performance and reliability:

- **Asynchronous Fetching**: The frontend uses asynchronous API calls to ensure the UI remains responsive while continuously polling for new letters.
- **Debouncing Input**: Letters are only accepted if not repeated within the polling interval, reducing duplication.
- **Buffering**: A live buffer structure retains prediction history and manages the voting logic.
- **Modular Backend**: The decoupled design allows easy upgrades, such as switching the CNN model or modifying the frontend.

These considerations ensure the system is smooth, stable, and scalable.

6. Importance of Sentence Formation in Sign Language Systems

Sign language is a rich and expressive form of communication, and simply recognizing letters is insufficient. The sentence formation module adds critical functionality:

- **Context Preservation**: By forming words and sentences, the system retains the context of communication.
- **Natural Flow**: Users can express themselves in a way similar to spoken language, making interactions more efficient.
- **Enhanced Accuracy**: Buffering and voting help correct short-term errors in recognition, improving overall user experience.

Without sentence formation, the system would be limited to character-level outputs, severely impacting usability.

7. Integration with Other Assistive Technologies

The system's modularity allows integration with various other assistive and communication systems, such as:

- **AAC Devices**: Augmentative tools for non-verbal users can receive input from our module.
- **Speech-to-Text Engines**: For reverse communication—converting others' speech into sign output.
- **Smart Devices and IoT Systems**: Commands can be issued through sign language for smart home control (e.g., "Turn on light").
- **Language Translation Tools**: Recognized text can be translated into different languages for multilingual support.

This flexibility increases the system's reach and impact across accessibility domains.

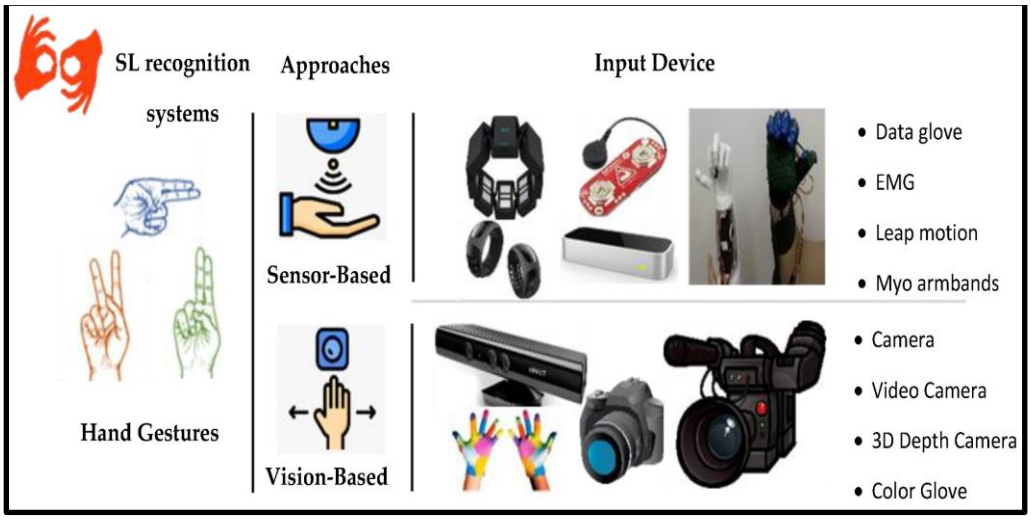


Fig-28

8. Future Scope and Enhancement Possibilities

While the current implementation supports isolated gesture recognition and sentence-to-speech conversion, there is ample scope for improvement and expansion:

- **Continuous Gesture Recognition**: Enabling recognition of fluid sign language gestures that represent entire words or phrases instead of single letters.
- **Voice Modulation Options**: Letting users select voice pitch, speed, or gender for more personalized TTS output.
- **Multilingual Support**: Allowing users to convert text into different languages for broader accessibility.
- **Offline Capabilities**: Embedding speech synthesis and prediction engine into standalone applications using frameworks like TensorFlow Lite and WebRTC.
- **Mobile and AR Integration**: Deploying the model on mobile devices or AR glasses to assist users in daily conversations.
- **Word Prediction and NLP Integration**: Using machine learning to suggest next words based on context.
- **Grammar Correction Modules**: Ensuring well-formed sentence structures.
- **Offline TTS Support**: Adding open-source engines to remove browser dependency.
- **Voice Personalization**: Letting users select speech voices and customize tone, pitch, and rate.
- **Gesture-to-Emoji Mapping**: Useful for casual communication in chat interfaces.

These enhancements would greatly expand the user experience and functionality

9. Real-World Applications and Impact

The sentence formation and Text-to-Speech (TTS) modules in the system significantly expand its applicability, providing meaningful solutions across various sectors. By transforming sign language into spoken words or text, these modules bridge the communication gap, offering new opportunities for individuals with hearing or speech disabilities. Here are some of the most impactful applications:

1. Education:

For students with hearing impairments, the integration of sentence formation and TTS modules into classroom settings can revolutionize their learning experience. By translating sign language into spoken words, these students can actively participate in discussions, respond to teachers ’questions, and collaborate with peers. This technology fosters inclusive education, allowing students to engage more effectively and build social connections within the classroom. Additionally, teachers can receive instant feedback on students' inputs, helping tailor their teaching methods to accommodate diverse learning needs.

2. Healthcare:

In healthcare settings, efficient communication between patients and medical professionals is critical. Patients with speech disabilities often struggle to express their symptoms, medical history, or needs, leading to potential misunderstandings. By employing TTS modules that convert sign language to clear audio output, patients can effectively convey their concerns to healthcare providers. This advancement can be especially vital during emergencies or in situations where accurate communication is paramount. Healthcare staff can also use the system to communicate instructions and reassurance, making medical care more accessible and empathetic.

3. Customer Service:

Inclusive customer service is increasingly becoming a priority for businesses and government institutions. At helpdesks or public service counters, staff members can use the system to interact seamlessly with customers who have hearing or speech impairments. Whether at airports, banks, government offices, or retail stores, this technology fosters an inclusive environment where every individual is treated with dignity and respect. Additionally, integrating TTS systems with automated kiosks can provide self-service options that cater to the needs of individuals with hearing disabilities.

4. Workplaces:

Promoting diversity and inclusion within the workplace is essential for modern organizations. Employees with hearing loss can face challenges in meetings, training sessions, or daily interactions with colleagues. By employing systems that convert sign language to speech or text, workplaces can ensure that all employees have an equal opportunity to contribute and collaborate. This technology not only enhances productivity but also demonstrates the company's commitment to inclusivity, boosting employee morale and fostering a positive work culture.

5. Public Transport and Navigation:

Navigating public spaces can be challenging for individuals with hearing disabilities, particularly in noisy environments like train stations, airports, or busy streets. The system's ability to translate sign language into audible instructions allows users to ask for directions or assistance without struggling to be understood. Moreover, public transport announcements can be interpreted and translated for users, helping them navigate more independently and confidently.

6. Social Interactions and Community Engagement:

Beyond practical applications, the system significantly enhances social inclusion. Individuals with hearing or speech disabilities can participate more fully in community events, cultural programs, or casual conversations. By promoting accessible communication, the technology helps build stronger community bonds and reduces the social isolation often experienced by those with communication challenges.

7. Accessibility in Public Services:

Government and community service centers can implement these systems to assist citizens with disabilities more efficiently. From social service applications to voter registration and legal assistance, ensuring accessible communication helps people navigate bureaucratic processes independently. This aligns with modern accessibility standards and demonstrates the government's commitment to serving all citizens equally.

8. Digital and Social Media Accessibility:

As social media becomes an integral part of modern communication, incorporating sign language translation and TTS into digital platforms can empower users with hearing impairments to participate actively. Whether creating video content, participating in online discussions, or accessing virtual meetings, this technology enables equal representation and participation in the digital space.

By implementing the sentence formation and TTS modules, the system not only enhances communication but also fosters a more inclusive society. From educational settings to healthcare, workplaces, public transport, and beyond, these technologies empower individuals with hearing and speech disabilities to overcome communication barriers. By promoting digital inclusion and equal opportunity, the system contributes to building a more accessible and empathetic world.

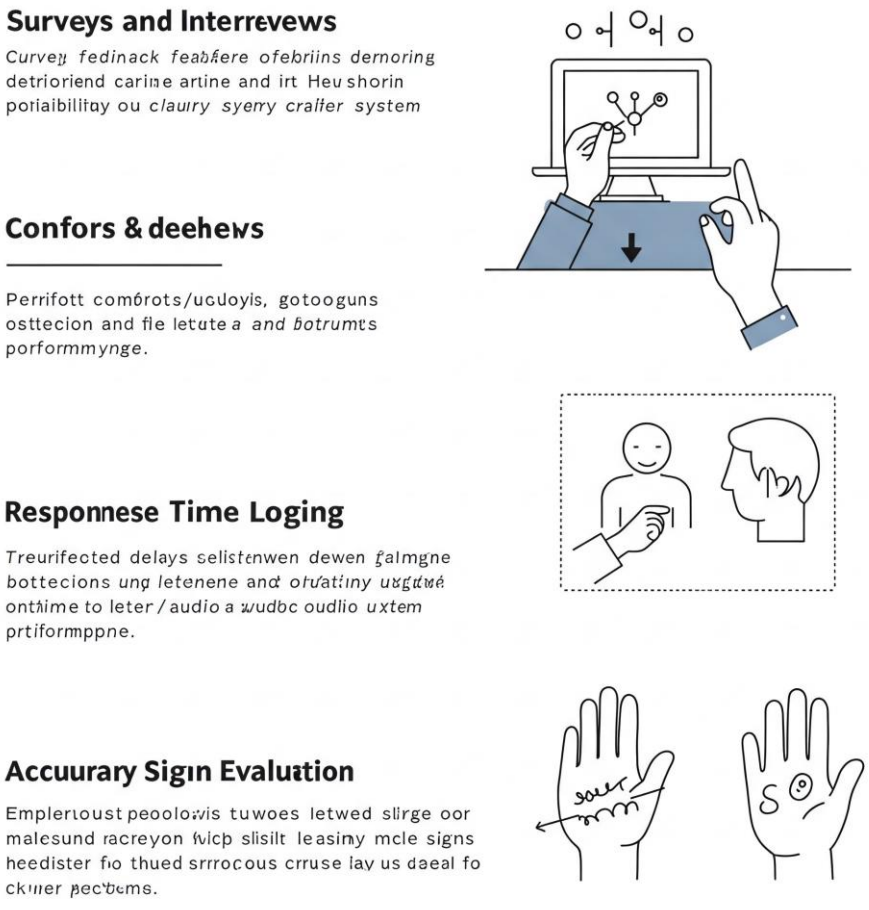


Figure-29

Fig-30

10. Usability Testing and Feedback

To evaluate real-world performance effectively, it is crucial to incorporate user testing and feedback loops into the development process. This approach ensures that the system not only functions correctly but also meets users' expectations and needs in practical scenarios. Here are some essential methods to gather valuable insights:

1. **User Surveys and Interviews:**

Engaging with users directly allows developers to collect qualitative feedback regarding the system's usability, comfort, and clarity. Surveys can be structured to include both quantitative (e.g., rating scales) and qualitative (e.g., open-ended questions) data, while interviews can provide in-depth insights into users' experiences and perceptions. By gathering feedback from diverse user groups, developers can identify common challenges and areas for improvement.

2. **Observational Studies:**

   Observing users as they interact with the system provides valuable context about how it performs under various conditions. This method highlights potential usability issues that may not surface through self-reported data alone, such as difficulties in making specific gestures or challenges in maintaining accuracy in noisy environments.

3. Response Time Logging:

   One of the critical performance metrics is the system's response time. Logging delays between gesture detection and the corresponding letter or audio output helps quantify the system's real-time efficiency. Analyzing response times under different conditions (e.g., varying lighting or movement speed) helps pinpoint performance bottlenecks and optimize processing algorithms.

4. Accuracy Evaluation Using Confusion Matrices:

   To assess the system's reliability, developers can use confusion matrices to track the accuracy of sign predictions. These matrices reveal which gestures are frequently misinterpreted and help identify patterns of error. For instance, some hand signs may appear similar to the system, leading to confusion. Analyzing these matrices helps in refining the gesture recognition model to enhance precision.

5. Field Testing with Diverse User Groups:

   Testing the system with a wide range of users, including individuals with different hand shapes, sizes, and movement styles, ensures robustness and generalizability. Real-world testing in varied environments (indoor, outdoor, low light) helps evaluate performance consistency.

6. User Experience Metrics:

   Beyond technical accuracy, it is vital to gauge users' satisfaction with the system. Metrics such as task completion rate, perceived ease of use, and user confidence in the system's predictions contribute to a holistic understanding of performance.

7. Continuous Feedback Loops:

   Integrating feedback mechanisms within the application itself allows users to report issues or suggest improvements as they use the system. Real-time feedback collection, combined with post-usage surveys, fosters a continuous improvement cycle, keeping the system aligned with users' evolving needs.

10. Conclusion

This project successfully demonstrates the potential of Convolutional Neural Networks (CNNs) for real-time sign language recognition and conversion to spoken language. The improved CNN model, featuring three convolutional layers and optimized for grayscale input, showed a significant enhancement in both training and validation accuracy, achieving near-perfect validation accuracy (almost 99.90%). This indicates the model's ability to effectively learn and generalize sign language gestures.

The system effectively translates recognized signs into text through a well-defined workflow involving letter detection, sentence formation, and a user-friendly frontend interface. Furthermore, the integration of the Web Speech API enables the conversion of this text into audible speech, providing a valuable tool for communication.

While the increased complexity of the improved model led to a slight increase in training time, the substantial gains in accuracy justify this trade-off. The developed system offers a promising foundation for future advancements in sign language interpretation technology, with the potential to bridge communication gaps and improve accessibility.

11, REFRENCES

[1] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, “Hand gesture recognition with 3D convolutional neural networks,” 2015 IEEE Conference on Computer Vision and Pattern RecognitionWorkshops (CVPRW), 2015. https://ieeexplore.ieee.org/document/7298714

[2] S. Souders, M. Galbraith, and Mozilla Contributors, “SpeechSynthesisUtterance - Web APIs | MDN,” Mozilla Developer Network, https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesisUtterance.

[3] G. Bradski and OpenCV.org contributors, “Background Subtraction — OpenCV documentation(v4.x),”OpenCV,https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html.

[4] N. Kasliwal et al., “pyttsx3 – Text-to-Speech Conversion Library in Python,” pyttsx3.readthedocs.io, https://pyttsx3.readthedocs.io/en/latest/.

[5] O. Koller, et al., ”Deep Learning for Sign Language Recognition: A Review,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 9, pp. 2109-2123, Sep. 2019. https://ieeexplore.ieee.org/document/8260925.

[6] R. Sharma, et al., ”Real-Time Sign Language Recognition Using Convo- lutional Neural Networks,” Proceedings of the 5th International Confer- ence on Advanced Computing and Communication Systems (ICACCS), 2019. https://ieeexplore.ieee.org/document/8 728414.

[7] S. Pramerdorfer and M. Kampel, ”Sign Language Recognition with 3D Convolutional Neural Networks,” Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 35-42. https://ieeexplore.ieee.org/document/8 268398.

[8] A. Abavisani, et al., ”Improving the Performance of CNN for Sign Language Recognition with Spatial and Temporal Attention,” Pattern Recognition Letters, vol. 131, pp. 90-98, 2020.

[9] S. S. Rautaray and A. Agrawal, ”Vision-Based Hand Gesture Recog- nition for Human-Computer Interaction: A Survey,” Artificial In- telligence Review, vol. 43, no. 1, pp. 1-54, Jan. 2015. https://link.springer.com/article/10.10 07/s10462-012-9356-9.

[10] J. Pu, et al., ”Attention-Based Mechanism for Sign Lan- guage Recognition Using Convolutional Neural Networks,” arXiv preprint arXiv:1909.09702, Sep. 2019. https://arxiv.org/abs/1909.09702.

[11] W. Cheng, et al., ”Multi-Scale CNN for Sign Language Recognition,” 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 1033-1037.https://ieeexplore.ieee.org/document/8 802920.

[12] H. Chen, et al., ”Sign Language Recognition Based on Three- Dimensional Convolutional https://link.springer.com/article/10.10 07/s11063-018-09788-y.