ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**INTERPRETATION OF INDIAN SIGN LANGUAGE WITH SPECIAL FOCUS ON STATEMENTS AND WORDS USING DEEP LEARNING AND NEURAL NETWORKS**" is the bonafide work of "**Dhivyashri Ramesh, Kavya Sridhar, Jyotir Aditya Giri**" who carried out the project work under my supervision.

DATE:	**SIGNATURE**
PLACE:	**Dr. K THANGARAMYA**
	COURSE IN-CHARGE
	TEACHING FELLOW
	DEPARTMENT OF CSE
	ANNA UNIVERSITY
	CHENNAI

TABLE OF CONTENTS

ABSTRACT

Hand gestures, facial expressions and body movements form the crux of communication in sign language. Both word-level signs and fingerspelling together form a sign language. It is the only means of communication for hearing impaired people. People capable of hearing well on the other hand seldom attempt to learn sign language. As a result, the hearing-impaired are unable to interact with people who are not fluent in sign language. A problem arises where an interpreter is required for essential communication. To address this, a system that identifies sign language automatically is required. The creation of such a system provides a platform for hearing-impaired people to interact with the rest of the world without the aid of an interpreter. A method is presented for automatically identifying fingerspelling in Indian sign language. This produces a means for communication between the parties involved. For identifying different signs, the suggested system employs digital image processing techniques and neural networks. This system aims to generate complete sentences by interpreting the Indian Sign Language instead of interpreting just a single letter from the signs. This diminishes the barrier of communication between people who use Indian Sign Language and people who don’t know Indian Sign Language or people who communicate using any other sign language.

CHAPTER 1
INTRODUCTION

1.1. Sign Language

Sign language is a visual means of communication carried out with hand signals, gestures, facial expressions, and body language. Though it's the main form of communication for the hearing-impaired community, sign language can be useful for other groups of people as well. Sign languages are full-fledged natural languages with their own grammar and lexicon. Every country has its own version of sign language which sometimes even differs based on region. The American and British Sign Language both involve fingerspelling with a single hand whereas the Indian Sign Language involves fingerspelling with two hands. The aforementioned term fingerspelling refers to the representation of alphabets and numbers using hands alone.

1.2. Problem Description

Bridging the communication gap between people who use Indian Sign Language and people who does not use it with more effective and efficient methods. The major inconvenience people with hearing impairment have is communication related as they cannot use spoken languages. Hence the only way for them to communicate is through sign language and when sign language isn't understood by other parties involved, there is a need for an interpreter. For the interpreter itself, focus needs to be interpreting Indian Sign Language and also sentences and words which form communication apart from just alphabets.

This problem falls under the category of Image Processing and specifically, Image classification; which is aimed to be achieved by using convolutional neural networks. The aim is to develop a user friendly human computer interface where the computer understands the Indian Sign Language.

This system aims to generate complete sentences by interpreting the Indian Sign Language instead of interpreting just a single letter from the signs. This diminishes the barrier of communication between people who use Indian Sign Language and people who don’t know Indian Sign Language or people who communicate using any other sign language.

1.3. SWOT Analysis

Strength

This model provides prediction for sentences and words from Indian Sign Language at a high enough accuracy from webcam captured input.

Weakness

Due to similarity of some of the given input letters, there is scope for error in prediction. Model will also struggle with distinguishing certain alphabets.

Opportunity

This will enable interpretation of Indian Sign Language without the presence of an interpreter enabling communication between the hearing impaired and people unfamiliar with the Indian Sign Language.

Threat

Inaccurate prediction can lead to miscommunication which is an unwanted circumstance and must be avoided.

1.4. Contributions

This project presents a user-friendly human computer interface where the computer understands the Indian Sign Language. This diminishes the barrier of communication between people who use Indian Sign Language and people who don't know Indian Sign Language or people who communicate using any other sign language.

Scope of this project encompasses recognition of sign language (Indian Sign Language), display of the recognized phrase/sentence/word along with recognizing letters and the display of word recommendation while recognizing a letter.

CHAPTER 2
LITERATURE SURVEY

Adithya V and Vinod P R, [2018], [10] proposed model that uses canny edge detection for preprocessing the image and SURF method for feature extraction to train the model. The paper employs Support Vector Machine, Convolutional Neural Network and Recurrent Neural Network for identifying the Indian Sign Language and concludes that the Support Vector Machine has a higher accuracy when combined with K-means clustering and Bag of Visual words classifiers. In addition to the gesture to text conversion, the paper goes a step ahead and proposes Google Speech Recognition Application Programming Interface for speech to gesture conversion which proves to minimize the communication gap. Usage of 20% data for validation serves as an advantage since the self-created dataset consists of 1200 different images for each image class. The proposed idea of self-creation of dataset is also a notable addition. Alongside this, the validation class used could be noted to provide a check for the overfitting and underfitting of a model. Though the accuracies of the models proposed are high, using multiple models significantly increases the computational costs.

Kartik Shenoy et. al, [2018], [5] presents a system which can recognize hand poses and gestures from the Indian Sign Language in real-time using grid-based features. The gestures in Indian Sign Language were considered for research on k-Nearest Neighbors, Hidden Markov Model using grid-based features. The system presented in this paper is accurately able to track hand movements of the sign demonstrator using techniques such as Object Stabilization, Face elimination, Skin color Extraction and then Hand extraction. It can classify all 33 hand poses in Indian Sign Language with an accuracy of 99.7%. The system was also able to classify 12 gestures with an average accuracy of 97.23%. A key takeaway from this project is the increased accuracy with the introduction of appropriate models for the input data which is provided in a real time fashion as well.

Thakur et. al, [2020], [9] proposed a model using Convolutional Neural Network Visual Geometric Group -16 model for the recognition of American Sign Language along with provisions for text to sign language conversion to enable a two-way communication between people with hearing impairments and people with no knowledge of sign language. As said in the paper, the proposed method worked well most of the time, but got many misclassifications. And thus, video capturing of the user's hand and splitting the frames and applying the algorithm on each individual frames and passing it to a majority voting classifier is done to improve the accuracy which has proven to work. An accuracy of 92.7% has been achieved in this paper.

Ankit Ojha et. al, [2020], [1] proposed creating a desktop application that uses a computer's webcam to capture visual footage of a person signing gestures for the American sign language (ASL), which uses the acquired footage to translate it into the corresponding text and speech in real time. To enable the detection of gestures, they have used a Convolutional Neural Network. A fingerspelling sign language translator is obtained which has an accuracy of 95%. The project can be extended to other sign languages by building the corresponding dataset and training the Convolutional Neural Network which can be adapted even for a project which is not centered around the American Sign Language. However, there are certain aspects of the project that need to be reconsidered. As mentioned in the paper itself, the threshold needs to be monitored to avoid distorted grayscales in the frames. Hence gloves are recommended to be used for signing. Another drawback is that bad gestures will not yield correct predictions. The existing project only works for American Sign Language; however, it can be extended to other languages with the right and sizeable dataset and training. This project again implements a fingerspelling translator and requires a higher degree of processing for interpreting words and sentences.

Dhivyasri S and Krishnaa Hari K B, [2021], [2] have proposed a method that translates the fingerspelling in Indian Sign Language to textual form using an Artificial Neural Network that has one input layer, one output layer and two hidden layers. The model uses only 10 images to train upon and 5 images for its testing for each of the image classes. The hand region is extracted from the background using skin color based segmentation in the YCbCr color space. The result of segmentation produces a binary image and the features are extracted using Fourier descriptors. The usage of a feed forward neural network and backpropagation algorithm with fully connected layers has helped achieve a significantly high accuracy of 91.5% even though the dataset is notably small. Therefore, the proposed model provides a solution to the problem of the lack of a large dataset for the Indian Sign Language. However, restricting the training set to have a black background and using skin color-based segmentation questions the usability of the proposed model in real world scenarios where the input is not always received under such conditions.

Gustaf Halvardsson et. al, [2021], [4] proposed to use Convolutional Neural Networks and transfer learning to make the computers be able to interpret signs of the Swedish Sign Language hand alphabet. It consists of the implementation of a pre-trained InceptionV3 network, and the usage of the mini-batch gradient descent optimization algorithm. Giving an accuracy of 85 percent, the paper presents using convolutional neural network as a promising approach for the purpose of interpretation of fingerspelling in swedish sign language. Furthermore, they have described the implementation details of the model to interpret signs as a user-friendly web application. Here, the problem of having a small dataset of the Swedish Sign Language data is solved using transfer-learning with a pre-trained model. However, this itself is the drawback too. Thus, it does not necessarily suggest that the same thing can be done for the rest of the sign language or for the hand alphabets of other sign languages.

Muhammad Al-Qurishi et. al, [2021], [7], reviewed the majority of currently available approaches for Sign Language Recognition tasks that are based on deep neural architectures that have been created over the last few years and grouped them into clusters based on their main characteristics. Interpretation of American Sign Language by integrating Convolutional Neural Networks with multiple Principal Component Analysis layers is proposed in this paper. The positions of hand components (important points in the hand) that are relevant for sign language communication are considered as the crux of feature extraction. Using a statistical approach for allowing the neural model to learn the probabilities of the learned features and its correlation with specific classes adds to the advantage.

Sakshi Sharma and Sukhwinder Singh, [2021], [8] proposed to perform the feature extraction and classification of ISL gestures. This study has primarily three contributions, a large Indian Sign Language dataset, increased generalizability of dataset by using augmentation and a Convolutional Neural Network model for feature extraction and classification. Being inspired by LeNet-5, which was proposed by LeCun, et al, the proposed convolutional neural network model has been designed to reduce the computation cost while maintaining accuracy and time consumption. The model proposed also does not use a lot of convolutional layers which reduces the computation cost. However, the drawback is that more work has to be done to enhance the accuracy of ISL recognition for the real-time scenario.

Ahmed Kasapbas et. al, [2022], [6] proposed to develop an American sign language recognition dataset and use it in the deep learning model which depends on neural networks to interpret gestures of sign language and hand poses to natural language. It contains a new addition to the datasets used for sign language recognition systems. Using about 3 datasets, the Convolutional Neural Network model itself has three convolutional layers displaying optimal results. A major plus point of this paper is the dataset that they've introduced factors to account for light and distance. It takes into account various conditions such as lighting and distance, which makes it superior to other datasets which use non-variable conditions. This dataset is a new addition to other datasets in the field of Sign Language Recognition, which can assist researchers and students in developing Sign Language Recognition systems. Furthermore, this work presents a Convolutional Neural Network that performs at a high accuracy under various conditions. Moreover, they have compared the accuracy of their dataset with those from other studies. Despite the different conditions and volume of the new dataset, it achieved 99.38% accuracy with a text-to-speech engine. However, this paper too deals with the Alphabet dataset alone and therefore doesn't deal with gestures as such for specific words. The notable features however are introduction of an inclusive dataset and methods to achieve accuracy.

CHAPTER 3
PROPOSED SYSTEM

3.1. System Architecture

Firstly, the dataset is created by clicking images of the various fingerspelling gestures along with the captured images of blanks to account for spaces in sentence formation. The obtained images are stored in created folders for compartmentalizing the data. Thereafter, to increase the generalizability of the dataset and to increase its size, augmentation is performed. Then the proposed system is then fed the augmented dataset which is then preprocessed by means of processes such as color space conversion, blurring and thresholding. Subsequently, the dataset undergoes a 75-25 split into training and testing data. The augmented data is then fed to the Convolutional Neural Network. Finally, the trained model is used by the application to make predictions.

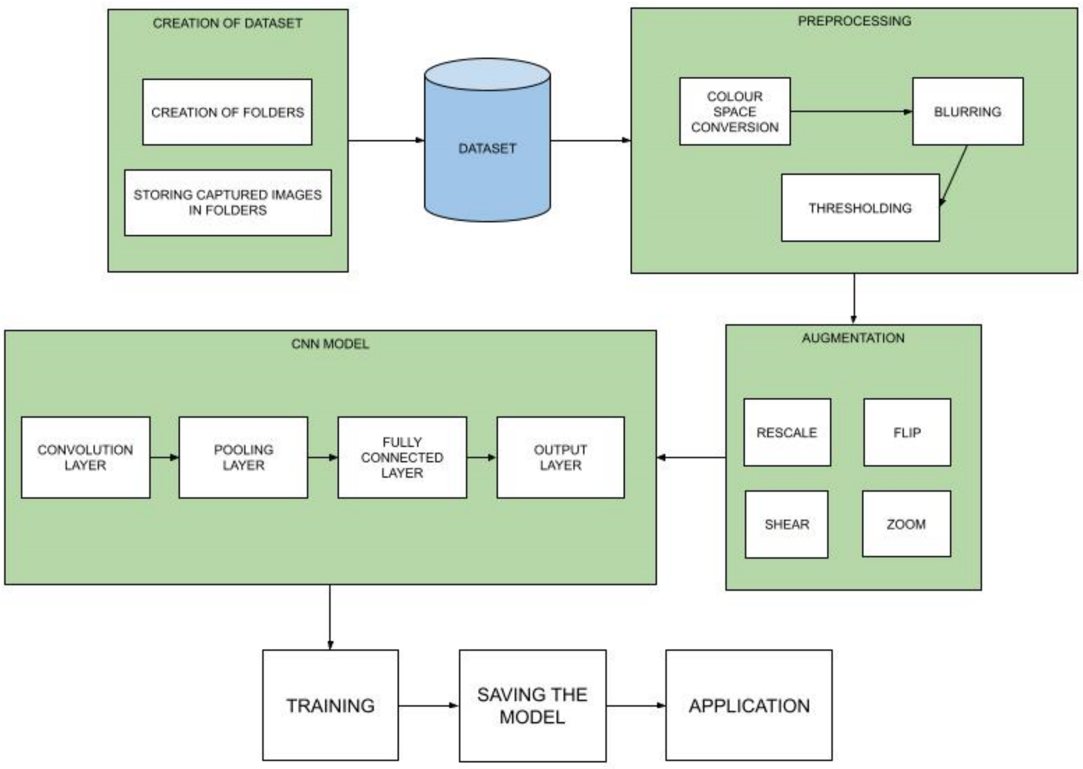

Fig 3.1: System Architecture

3.2. Proposed Methodology

3.2.1. Image Acquisition and Creating dataset

The first and foremost step involves creation of folders for each alphabet in which the captured images for respective alphabets is stored. 26 such folders are created and the created dataset is used for the system.

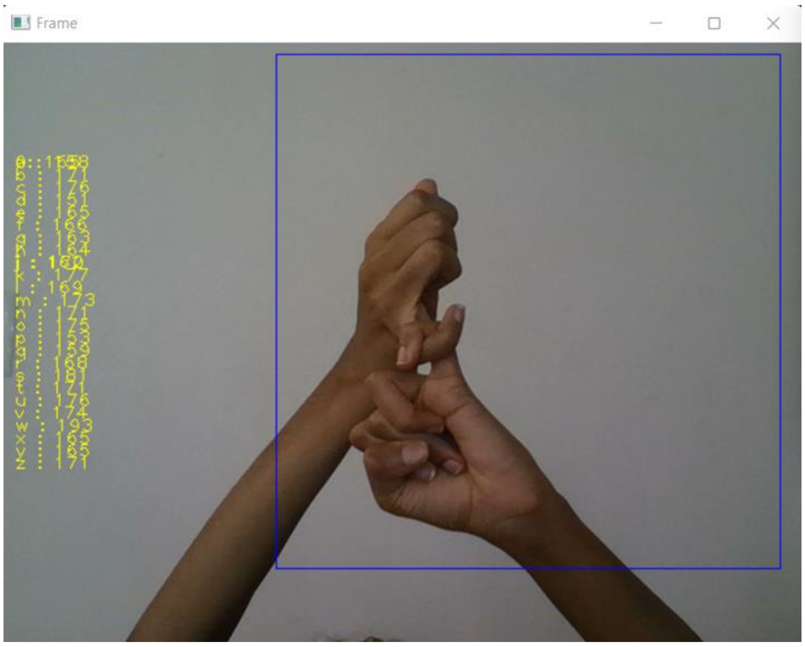

Fig 3.2: Interface for Image Acquisition

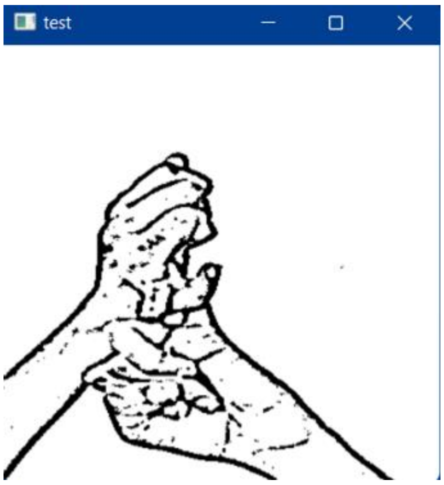

Fig 3.3: Preview of preprocessed image in the Interface

The machine's webcam is used to capture the image and in each frame a region of interest is defined using OpenCV and is denoted by a blue bounded square (Fig 3.2). The interface also allows for previewing the preprocessed image of the input (Fig 3.3).

There are around 170 images for each of the 26 classes and a total of 4552 images in the dataset (Fig 3.4).

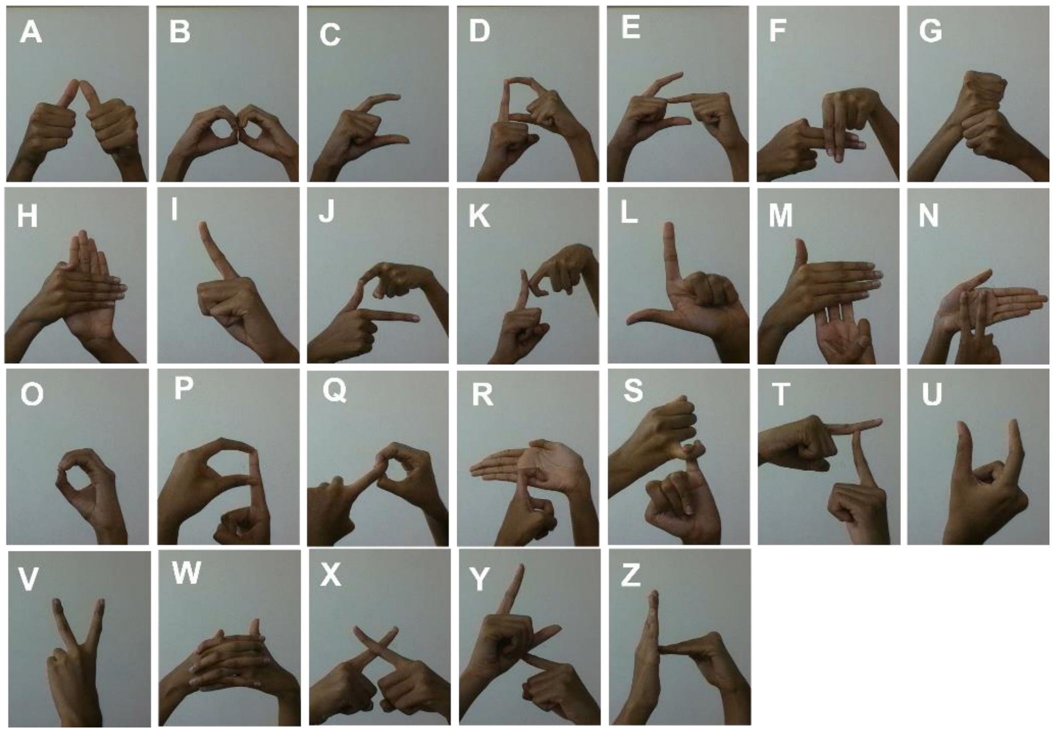

Fig 3.4: Dataset Captured

3.2.2. Preprocessing of Dataset

After capturing the image from the ROI, a Gaussian blur filter is applied to the grayscale converted image which helps to extract various features and reduces the noise in the input image. In thresholding, the gray-scale image is reduced to a binary image.

Color space conversion

The conversion of a color image into a grayscale image is converting the RGB values (24 bit) into grayscale value (8 bit).

Gaussian Blurring

It is a low-pass filter that removes high-frequency components of the image.

Thresholding

Thresholding is a type of image segmentation, where we change the pixels of an image to make the image easier to analyze. It converts the grayscale into a binary image which contains either black or white pixels.

The preprocessed images (Fig 3.5) are used to train the CNN model.

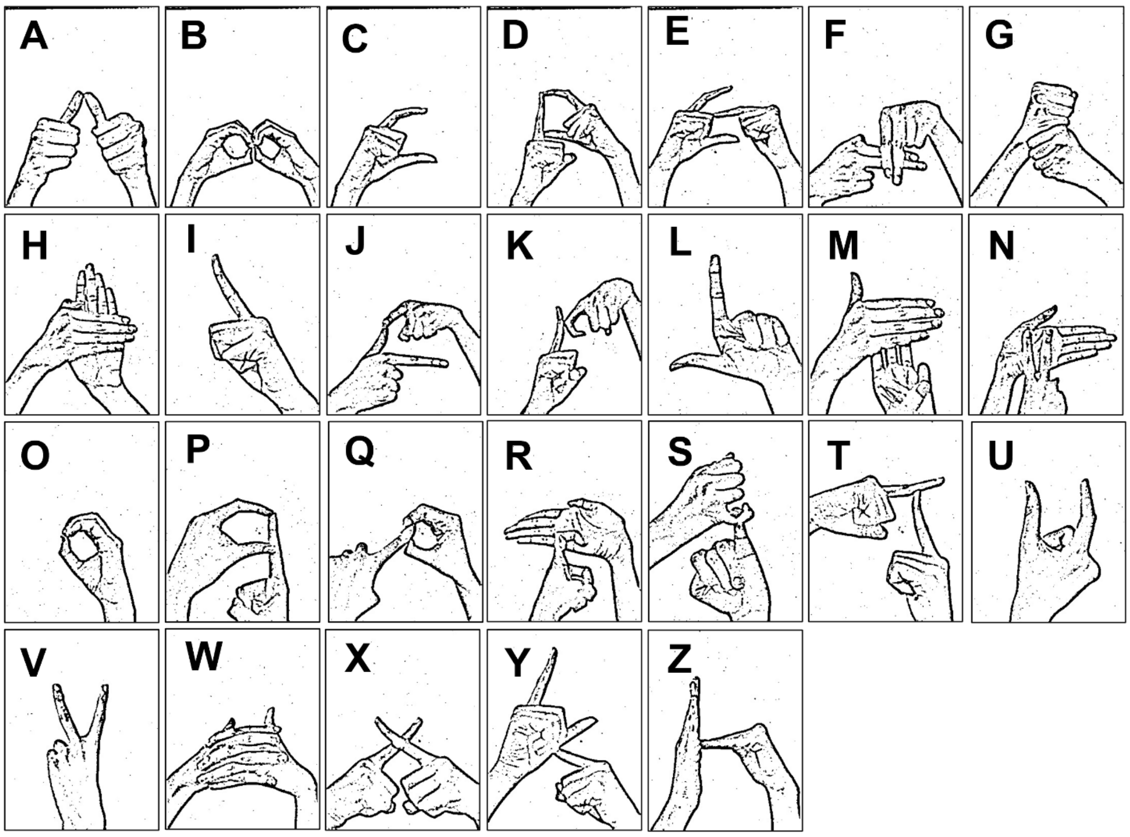

Fig 3.5: Pre-Processed Dataset

3.2.3. CNN Architecture

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.

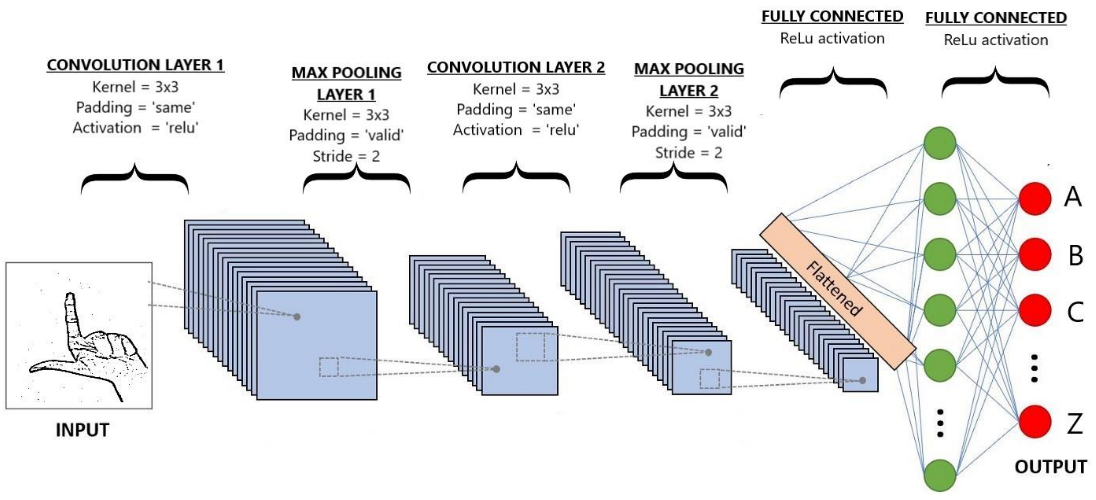

Fig 3.6: CNN Architecture

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.

The model (Fig 3.6) is trained using the layers described below.

Sequential

Sequential groups a linear stack of layers into a Model. Sequential provides training and inference features on this model

Convolutional Layers

A convolution layer transforms the input image in order to extract features from it. In this transformation, the image is convolved with a kernel (or filter). A kernel is a small matrix, with its height and width smaller than the image to be convolved. It is also known as a convolution matrix or convolution mask.

Fully Connected Layers

They are layers in neural network where all the inputs from one layer are connected to every activation unit of the next layer (Fig 3.7). In most machine learning models, the last few layers are full connected layers which compiles the data extracted by previous layers to form the final output.

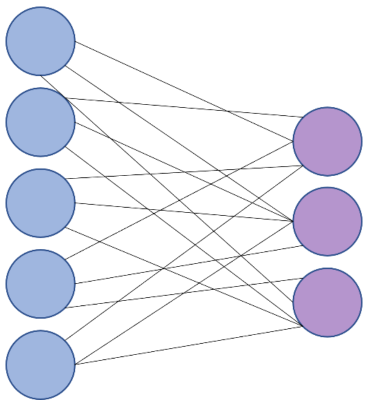

Fig 3.7: Fully Connected Layer

Max Pooling Layer

Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (Fig 3.8).

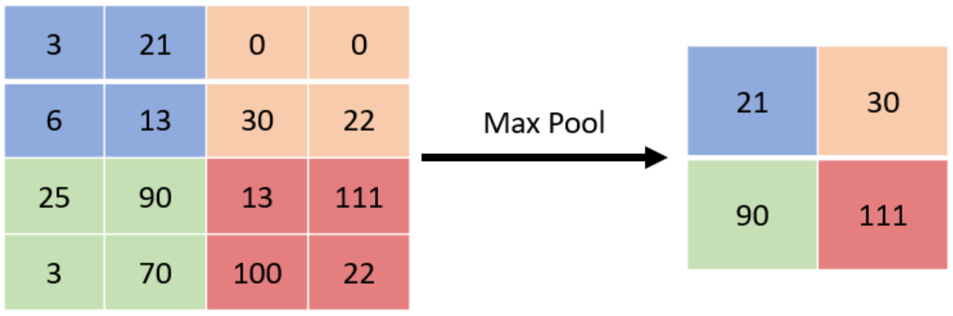

Fig 3.8: Max Pooling

Dropout Layer

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting

Flattening

Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

Dense Layer

Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers. Working of single neuron. A layer contains multiple number of such neurons.

The CNN Architecture (Table 3.1) of the model is shown below.

Table 3.1: CNN Model Summary

Layer (type)	**Output Shape**	**Parameters**
conv2d (Conv2D)	128, 128, 32	320
max_pooling2d (MaxPooling2D)	64, 64, 32	0
conv2d_1 (Conv2D)	64, 64, 32	9248
max_pooling2d_1 (MaxPooling2D)	32, 32, 32	0
flatten (Flatten)	32768	0
dense (Dense)	128	4194432
dropout (Dropout)	128	0
dense_1 (Dense)	96	12384
dropout_1 (Dropout)	96	0
dense_2 (Dense)	64	6208
dense_3 (Dense)	27	1755

3.2.4. Augmentation of Dataset

The ImageDataGenerator class ensures that the model receives new variations of the images at each epoch. But it only returns the transformed images and does not add it to the original corpus of images. An advantage of ImageDataGenerator class is that it requires lower memory usage. This is so because without using this class, we load all the images at once. But on using it, we are loading the images in batches which saves a lot of memory. The following augmentation techniques are used.

Random Zoom

The zoom augmentation either randomly zooms in on the image or zooms out of the image. ImageDataGenerator class takes in a float value for zooming in the zoom_range argument.

Random Flips

ImageDataGenerator class has parameters horizontal_flip and vertical_flip for flipping along the vertical or the horizontal axis.

Rescale

The parameter rescale is to multiply every pixel in the preprocessing image. If the rescaling factor is none or 0, no rescaling is applied, otherwise we multiply the data by the value provided.

Shear

A transformation that slants the shape of an object is called the shear transformation. shear_range=0.2 means shear the image by 20%.

Applying these augmentations on the acquired images increases the instances of the dataset from 4552 images to 27,090 images.

3.2.5. Training the CNN Model

The images from the dataset and their augmented images along with the labels are used to train the model after preprocessing where the noise in the data is minimized. The input image as a matrix passes through the layers of the CNN model and prediction layer estimates how likely the image will fall under one of the classes. The number of neurons in the final layer is same as the number of output classes and thus has 27 neurons. The output from the prediction layer is normalized between 0 and 1 and such that the sum of each value in each class sums to 1. This has been achieved using SoftMax function.

There are 4,224,282 trainable parameters in the CNN model and all the parameters are updated in each epoch. The model trains on batch learning and 10 such batches are used for the training of the model. It is observed that the prediction accuracy increases upon each epoch (Table 3.2).

Table 3.2: Epoch Metrics

Epoch No.	Loss	Accuracy	Val_Loss	Val_Accuracy
1	1.1878	0.6269	0.2581	0.9468
2	0.3298	0.8933	0.0931	0.9795
3	0.2368	0.9254	0.0743	0.9797
4	0.1827	0.9427	0.0327	0.9886
5	0.1500	0.9515	0.0790	0.9809
6	0.1258	0.9606	0.0597	0.9834
7	0.1140	0.9630	0.0446	0.9873
8	0.0969	0.9691	0.0265	0.9926
9	0.0941	0.9708	0.0522	0.9880
10	0.0792	0.9756	0.0594	0.9851
11	0.0806	0.9752	0.0484	0.9867
12	0.0713	0.9794	0.0720	0.9849

3.2.6. ISL Graphical User Interface

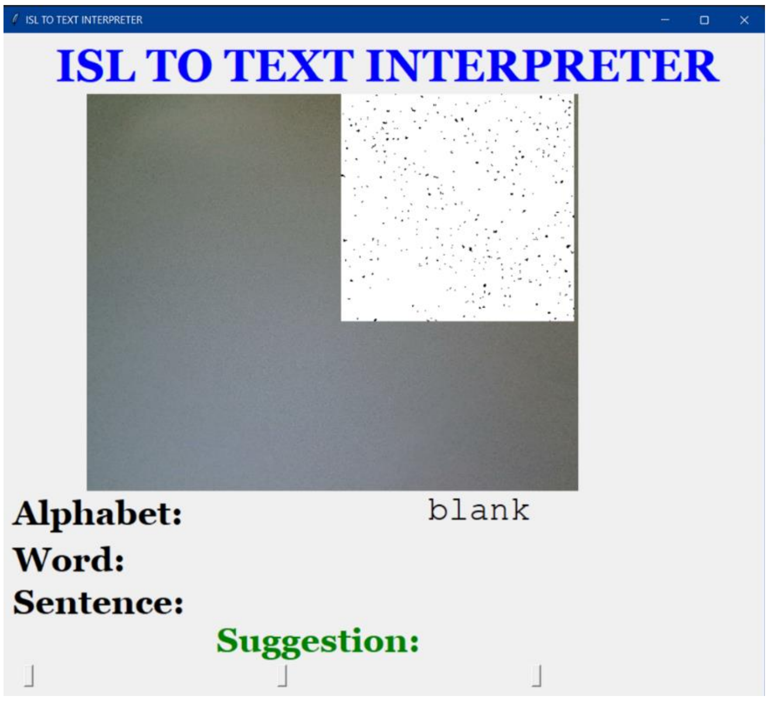

Fig 3.9: Graphical User Interface for ISL Interpretation

The Graphical User Interface (Fig 3.9) consists of a frame for the convenience of the user to see the input being captured via the web cam. As the user fingerspells the application takes time to register a particular alphabet and combines many such meaningfully entered alphabets to form words while producing suggestions for the same. The registered words are presented as sentences.

CHAPTER 4
RESULT AND ANALYSIS

4.1. Prediction

The process of prediction occurs sequentially in the application. Firstly, the application waits for the user to fingerspell and locks a particular letter once it has recognized the letter while producing suggestions for the word to be formed. As shown in Fig 4.1, letter ‘I’ which by itself is a word has been recognized after being demarcated by a space and it has been promoted to the sentence display. Then the new alphabet ‘L’ is recognized, marking its presence in alphabet segment.

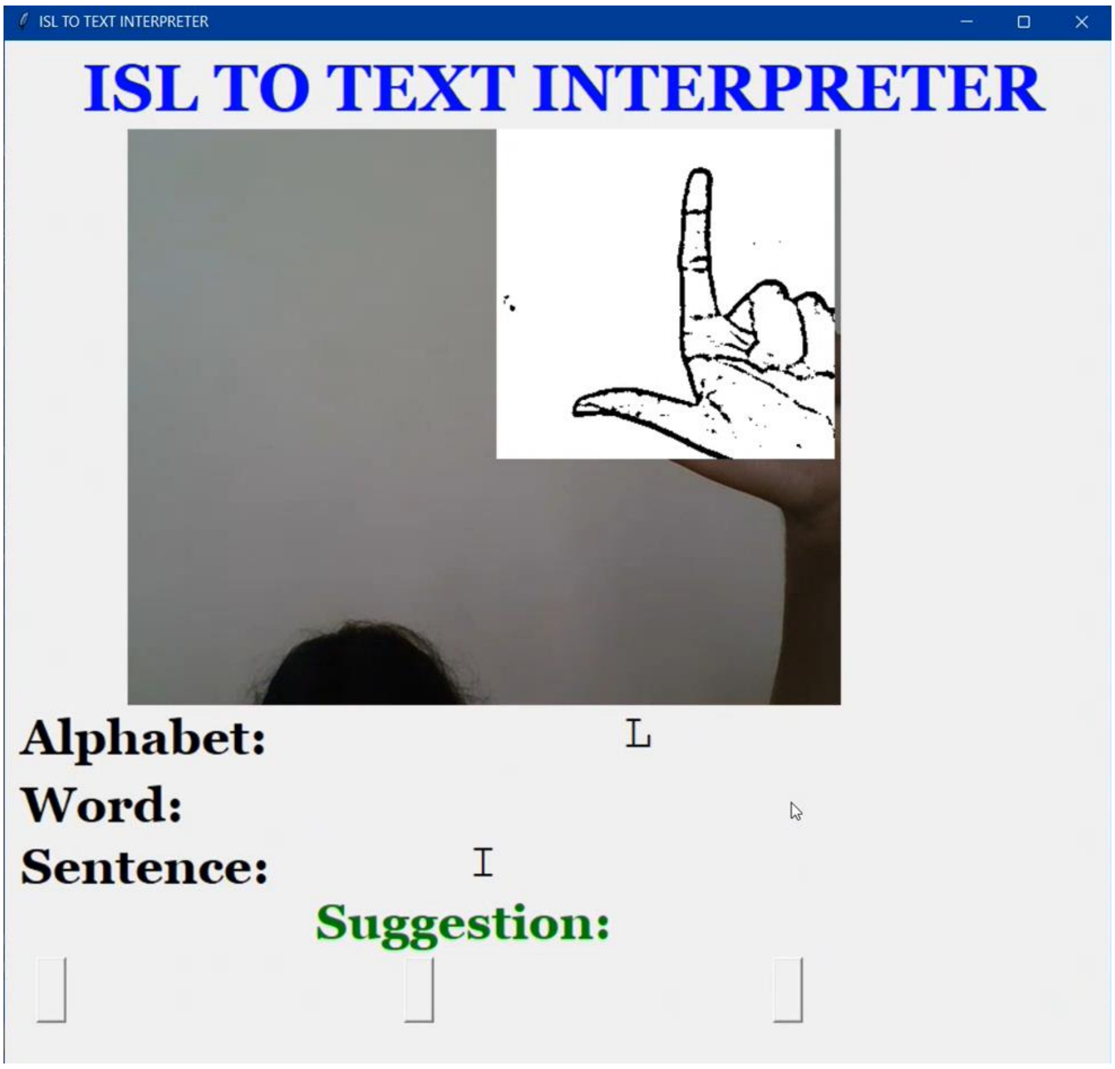

Fig 4.1: Formation of new word after promotion of word to sentence

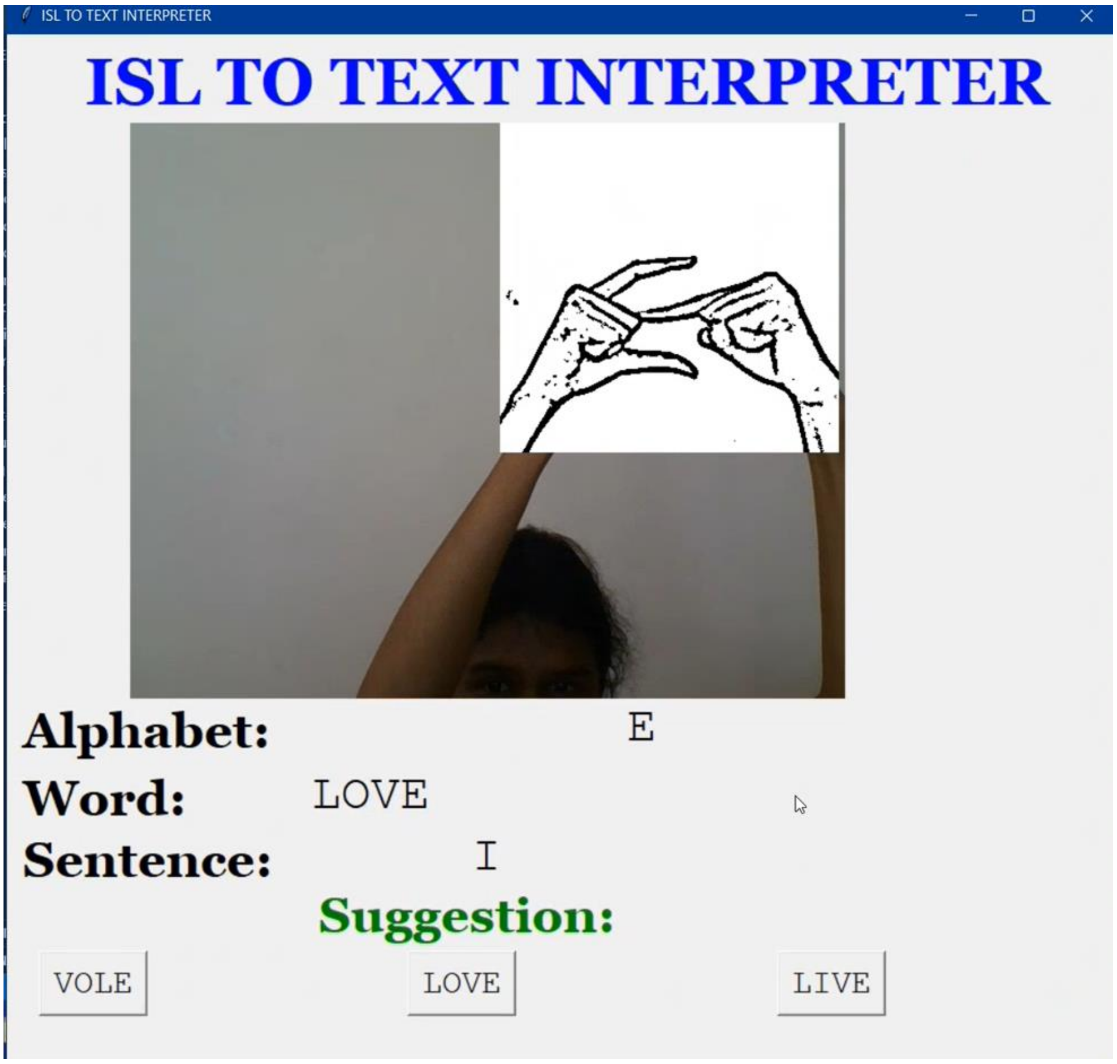

Fig 4.2: Suggestions based on the current word

Thereafter, when subsequent letters for the word are entered multiple suggestions are presented to aid the user (Fig 4.2). As each letter gets recognized and a meaningful word is formed demarcated by a space, the combined letters are moved down to be presented together as a word.

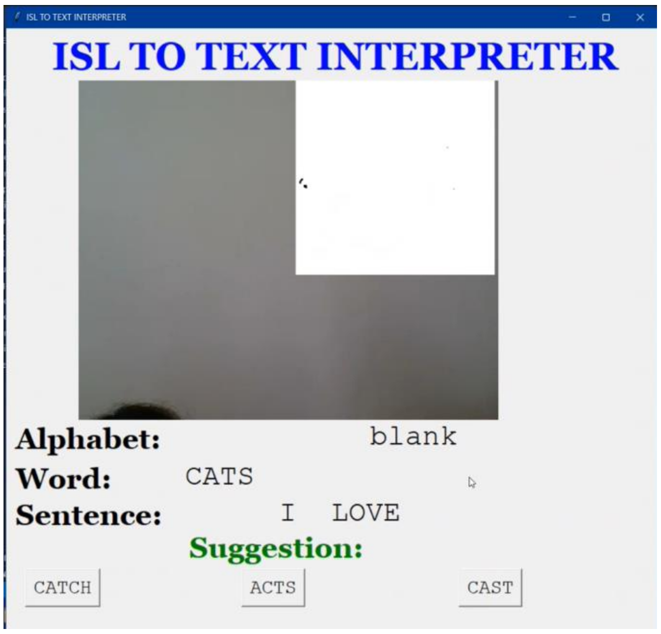

Fig 4.3: Formation of sentence

Similarly, the words which have been recognized are now promoted to the sentence segment where they will be forming meaningful sentences. It can be noted how the words “I” and “love” are a part of the sentence (Fig 4.3).

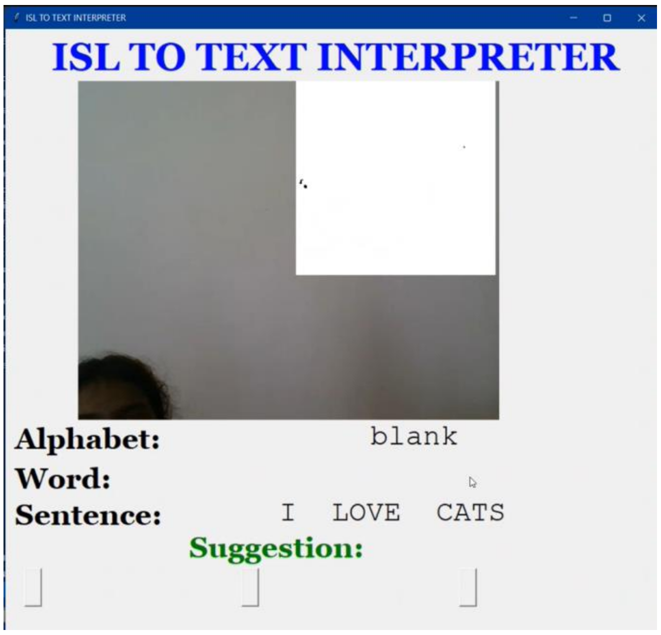

Fig 4.4: Completion of a sentence

The final presence of a blank (Fig 4.4) now signifies the end of the final word which is moved into the sentence view and the alphabet view shows a blank denoting that there is current no valid fingerspelling gesture. Thus, the final sentence is displayed in the sentence view (Fig 4.4).

4.2. Evaluation Metrics

4.2.1. Accuracy

An accuracy metric is used to measure the algorithm’s performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage. It is the measure of how accurate your model's prediction is compared to the true data. It is the ratio of correctly predicted images to the total number of predictions of the proposed classifier. It is also known as classification accuracy and can be determined using Equation 4.1.

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+FP+TN+FN)} \qquad \text{Eq 4.1}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

4.2.2. Loss

A loss function is used to optimize a learning algorithm. The loss is calculated on training and validation and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or validation sets. Loss value implies how poorly or well a model behaves after each iteration of optimization. The loss function used is Categorical Cross-entropy.

Categorical Cross-entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one. The categorical cross-entropy loss function calculates the loss of an example by computing the Equation 4.2.

$$\text{Loss} = -\sum_{i=1}^{output\ size} \text{yi}.\log(\hat{y}i) \qquad \text{Eq 4.2}$$

where $\hat{y}i$ is the ith scalar value in the model output, yi is the corresponding target value, and output size is the number of scalar values in the model output.

4.3. Acquired metrics

The proposed system was able to achieve a validation accuracy of 98.49% with a validation loss of 0.0720.

The training and validation accuracies have been plotted to analyze the fitting and the learning rates of the model. The accuracy curve (Fig 4.5) presents that the model has very little overfitting and is of a good fit.

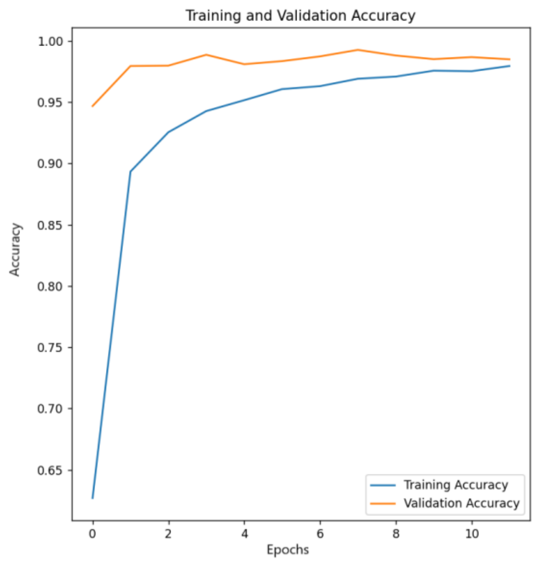

Fig 4.5: Accuracy Curve

As the distance between the training and validation curve minimizes and stabilizes while the training loss curve (Fig 4.6) shows a good learning rate, we can confirm the stability of this model through this.

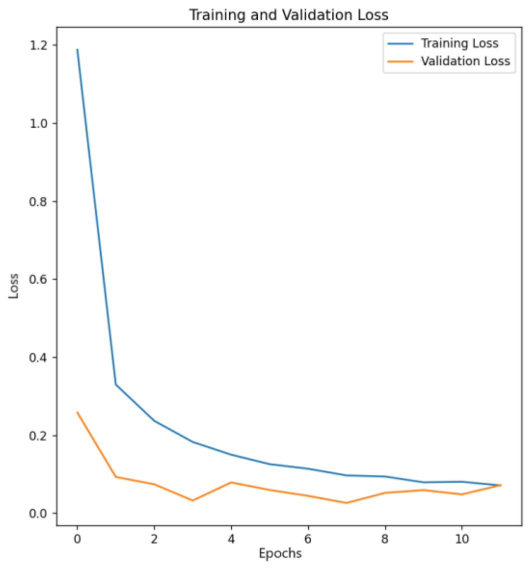

Fig 4.6: Loss Curve

CHAPTER 5
CONCLUSION

From the results it can be said that the proposed model works with an expected accuracy considering the size and diversity of the dataset with minimal error in some images. The project involved the creation of a dataset, which is a rarely explored language in the domain as most projects focus on American or British Sign Language. In the field of sign language recognition, both the tasks of producing an interpretation for sign language with suggestions for words and sentences and providing interpretation for Indian Sign Language have been under explored. The paper modestly achieves both the goals satisfactorily. The proposed system was able to achieve a validation accuracy of 98.49% with a validation loss of 0.0720. The model however can definitely be improved upon to encompass further details. Though the model provides prediction for alphabets, work can be done to reduce the time required to predict letters by ensuring that similar letters are well differentiated by the model. Currently, the working application requires input without a great amount of noise, future work could improve upon processing data from a much more realistic standpoint where there can be interference in the background.

APPENDICES

Pre-processing

```
def func(path):
        frame = cv2.imread(path)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray,(5,5),2)
        th3=cv2.adaptiveThreshold(blur,255,
                                cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY_INV,11,2)
        ret,res = cv2.threshold(th3,minValue,255,
                        cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
        return res
```

Dataset Augmentation

```
import cv2
import numpy as np
import os
import string
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
directory = 'dataset/'
sdir = 'aug_dataset/'
if not os.path.exists("aug_dataset"):
   os.makedirs("aug_dataset")
if not os.path.exists("aug_dataset/0"):
   os.makedirs("aug_dataset/0")

#A to Z folders in the training and testing folders

for i in string.ascii_uppercase:
   if not os.path.exists("aug_dataset/" + i):
      os.makedirs("aug_dataset/" + i)
```

```
#Count of existing images
count = {
    '0': len(os.listdir(directory+"/0")),
    'a': len(os.listdir(directory+"/A")),
    'b': len(os.listdir(directory+"/B")),
    'c': len(os.listdir(directory+"/C")),
    'd': len(os.listdir(directory+"/D")),
    'e': len(os.listdir(directory+"/E")),
    'f': len(os.listdir(directory+"/F")),
    'g': len(os.listdir(directory+"/G")),
    'h': len(os.listdir(directory+"/H")),
    'i': len(os.listdir(directory+"/I")),
    'j': len(os.listdir(directory+"/J")),
    'k': len(os.listdir(directory+"/K")),
    'l': len(os.listdir(directory+"/L")),
    'm': len(os.listdir(directory+"/M")),
    'n': len(os.listdir(directory+"/N")),
    'o': len(os.listdir(directory+"/O")),
    'p': len(os.listdir(directory+"/P")),
    'q': len(os.listdir(directory+"/Q")),
    'r': len(os.listdir(directory+"/R")),
    's': len(os.listdir(directory+"/S")),
    't': len(os.listdir(directory+"/T")),
    'u': len(os.listdir(directory+"/U")),
    'v': len(os.listdir(directory+"/V")),
    'w': len(os.listdir(directory+"/W")),
    'x': len(os.listdir(directory+"/X")),
    'y': len(os.listdir(directory+"/Y")),
    'z': len(os.listdir(directory+"/Z")) }
# Initialising the ImageDataGenerator class.
datagen = ImageDataGenerator(rescale = 1./255,
                             shear_range = 0.2,
                             zoom_range = 0.2,
                             horizontal_flip = True)
```

```
#Augmenting folders
for j in range(1,count['0']):
    img =
tf.keras.preprocessing.image.load_img(directory+"/"+"0"+"/"+str(j)+".jpg")
    x = tf.keras.preprocessing.image.img_to_array(img)
    # Reshaping the input image
    x = x.reshape((1, ) + x.shape)
    # Generating and saving 5 augmented samples//using the above defined
parameters.
    k = 0
    for batch in datagen.flow(x, batch_size = 1,
                        save_to_dir = sdir+"/0",
                        save_prefix =len(os.listdir(sdir))+1, save_format ='jpeg'):
        k += 1
        if k > 5:
            break

#choose from A to Z
for i in range(65,91):
    #traversing through all the images in a alphabet
    for j in range(1,count[chr(i+32)]):
        img = tf.keras.preprocessing.image.load_img
(directory+"/"+str(chr(i))+"/"+str(j)+".jpg")
        # Converting the input sample image to an array
        x = tf.keras.preprocessing.image.img_to_array(img)
        # Reshaping the input image
        x = x.reshape((1, ) + x.shape)
        # Generating and saving 5 augmented samples
        k = 0
        for batch in datagen.flow(x, batch_size = 1,
                            save_to_dir = sdir+"/"+chr(i),
                            save_prefix=len(os.listdir(sdir+str(chr(i))))+1,
save_format ='jpeg'):
            k += 1
        if k > 5:
            break
```

CNN Model

```
#Initializing the CNN
classifier = tf.keras.models.Sequential()

#1.Convolution
classifier.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding="same",
activation="relu", input_shape=[128, 128, 1]))
#2.Pooling

classifier.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2,  padding='valid'))
#3.Adding a second convolutional layer

classifier.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,padding="same",
activation="relu"))
classifier.add(tf.keras.layers.MaxPool2D(pool_size=2,  strides=2,  padding='valid'))

#4.Flattening

classifier.add(tf.keras.layers.Flatten())

#5.Full Connection

classifier.add(tf.keras.layers.Dense(units=128,  activation='relu'))

classifier.add(tf.keras.layers.Dropout(0.40))

classifier.add(tf.keras.layers.Dense(units=96, activation='relu'))

classifier.add(tf.keras.layers.Dropout(0.40))

classifier.add(tf.keras.layers.Dense(units=64, activation='relu'))

classifier.add(tf.keras.layers.Dense(units=26, activation='softmax'))

#Training the CNN / Compiling the CNN

classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
```

```
#Training the CNN

classifier.summary()
nepochs = 10
history = classifier.fit(training_set,epochs = nepochs ,validation_data = test_set)
#Saving the Model
model_json = classifier.to_json()
with open("model_new.json", "w") as json_file:
    json_file.write(model_json)
print('Model Saved')
classifier.save_weights('model_new.h5')
print('Weights saved')
```

REFERENCES

[1] Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, Dr. Dayananda P, 2020, Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCAIT – 2020 (Volume 8 – Issue 15).

[2] D. S, K. H. K B, A. M, S. M, D. S and K. V, "An Efficient Approach for Interpretation of Indian Sign Language using Machine Learning," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), 2021, pp. 130-133, doi: 10.1109/ICSPC51351.2021.9451692.

[3] G. Ananth Rao and P.V.V. Kishore, "Selfie video based continuous Indian sign language recognition system", ScienceDirect, 2018.

[4] Halvardsson, G., Peterson, J., Soto-Valero, C. et al. Interpretation of Swedish Sign Language Using Convolutional Neural Networks and Transfer Learning. SN COMPUT. SCI. 2, 207 (2021).

[5] Kartik Shenoy, Tejas Dastane, Varun Rao and Devendra Vyavaharkar, "Real-time Indian Sign Language (ISL) Recognition", IEEE Xplore, October 2018.

[6] KASAPBAŞI, Ahmed & Elbushra, Ahmed & AL-HARDANEE, Omar & YILMAZ, Arif. (2022). DeepASLR: A CNN based Human Computer Interface for American Sign Language Recognition for Hearing-Impaired Individuals. Computer Methods and Programs in Biomedicine Update. 2. 100048. 10.1016/j.cmpbup.2021.100048.

[7] M. Al-Qurishi, T. Khalid and R. Souissi, "Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues," in IEEE Access, vol. 9, pp. 126917-126951, 2021, doi: 10.1109/ACCESS.2021.3110912.

[8] Sharma, Sakshi & Singh, Sukhwinder. (2022). Recognition of Indian Sign Language (ISL) Using Deep Learning Model. Wireless Personal Communications. 123. 10.1007/s11277-021-09152-1.

[9] Thakur, Amrita & Budhathoki, Pujan & Upreti, Sarmila & Shrestha, Shirish & Shakya, Subarna. (2020). Real Time Sign Language Recognition and Speech Generation. Journal of Innovative Image Processing. 2. 65-76. 10.36548/jiip.2020.2.001.

[10] V. Adithya, P. R. Vinod and U. Gopalakrishnan, "Artificial neural network based method for Indian sign language recognition," 2013 IEEE Conference on Information & Communication Technologies, 2013, pp. 1080-1085, doi: 10.1109/CICT.2013.6558259.