Sign Language To Text Conversion

Submitted for the Partial Fulfillment of the Requirements for the Degree of

Bachelor of Technology

in Computer Science and Engineering

J.C. Bose University of Science and Technology, Faridabad

Submitted By:

Bittu Singh

Roll No. 19020004030

Under the guidance of

Ms. Bhawana Srivastava
Asst. Prof. CSE Dept. SDIET

Dr. Pawan Bhadana
Principal, SDIET

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Satyug Darshan Institute of Engineering and Technology, Faridabad

Declaration of Authorship

This is to certify that the project titled "**Sign Language to Text Conversion**" is the bonafide work carried out by **BITTU SINGH (CSE-19/030)**, student of B Tech (CSE) of Satyug Darshan Institute of Engineering & Technology, Faridabad affiliated to J.C. Bose University of Science and Engineering, Faridabad, Haryana (India) during the academic year 2022 – 23, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science Engineering) under the superintendence of Ms. Bhawana Srivastava, Asst. Prof. of the Department of Computer science and Engineering (CSE), Satyug Darshan Institute of Engineering and Technology (SDIET), Faridabad. It's additionally declared that this project has not been submitted anywhere else for any degree or sheepskin. Info derived from the printed and unpublished work of others has been acknowledged within the text and a listing of references is given.

Author	**Signature of the Guide**
BITTU SINGH	**Place:**
Roll no. 19020004030	**Date:**

Acknowledgement

First and foremost, I would like to convey my heartfelt gratitude to My Parents for their tremendous support. I would like to express my deep and sincere gratitude to our guide, Ms. Bhawana Srivastava, Asst. Prof. Department of Computer Science & Engineering, Satyug Darshan Institute of Engineering and Technology , for allowing us to do the project and providing invaluable guidance throughout this project. Her dynamism, vision, sincerity and motivation have deeply inspired us. She has taught us the methodology to carry out the project and to present the project works as clearly as possible. It was a great privilege and honour to work and study under her guidance. I am extremely grateful for what she has offered us. I would also like to thank my fellow classmates for their friendship, empathy, and helping attitude. This was quite a great experience and I learned a lot from it. It helped me to explore my skills and increased my interest in this project.

Abstract

Sign language is one of the oldest and most natural forms of language for communication, but since most people do not know sign language and interpreters are very difficult to come by, I have come up with a real time method using neural networks for fingerspelling based on American sign language. In our method, the hand is first passed through a filter and after the filter is applied the hand is passed through a classifier which predicts the class of the hand gestures. Our method provides 95.7 % accuracy for the 26 letters of the alphabet.

In addition to achieving high accuracy for fingerspelling recognition, our method also offers several advantages over traditional sign language interpretation. By leveraging neural networks and real-time processing, our system eliminates the need for an interpreter, providing immediate and efficient communication for individuals who are deaf or hard of hearing. With the potential to expand beyond fingerspelling, our research opens up possibilities for developing comprehensive sign language recognition systems that enhance inclusivity and facilitate seamless communication between deaf and hearing individuals.

Keywords: Sign Language, Feature Extraction and Representation, ANN, Fingerspelling, American sign language, neural networks, CNN, real-time processing, Tensorflow, Keras inclusivity.

Index

	Title Page	i
	Declaration of the Student	ii
	Acknowledgement	iii
	Abstract	iv
1.	**INTRODUCTION**	**1**
2.	**MOTIVATION**	**3**
3.	**LITERATURE REVIEW**	**4**
	3.1 Data Acquisition	4
	3.2 Data Preprocessing	4
	3.3 Feature Extraction	5
	3.4 Gesture Classification	5
4.	**KEYWORD AND DEFINITION**	**7**
	4.1 Feature Extraction and Representation	7
	4.2 Artificial Neural Network (ANN)	7
	4.3 Convolutional Neural Network (CNN)	8
	4.4 TensorFlow	10
	4.5 Keras	10
	4.6 openCV	10
5.	**METHODOLOGY**	**11**
	5.1 Data Set Generation	11
	5.2 Gesture Classification	13
	5.3 Finger Spelling Sentence Formation Implementation	16

	5.4 Auto-correct Feature	17
	5.5 Training and Testing	17
6.	**CHALLENGES FACED**	**18**
7.	**RESULTS**	**19**
8.	**CONCLUSION**	**21**
9.	**FUTURE SCOPE**	**22**
10.	**REFERENCES**	**23**
	APPENDIX	**24**

1. INTRODUCTION

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 use of their hands to express 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.

Sign language is a visual language and consists of 3 major components [6]:

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 body position.

Figure 1. Components of Sign Language

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. The system is built for the automated conversion of ASL into textual content and speech.

In my project I 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 I aim to train are as given in the image below.

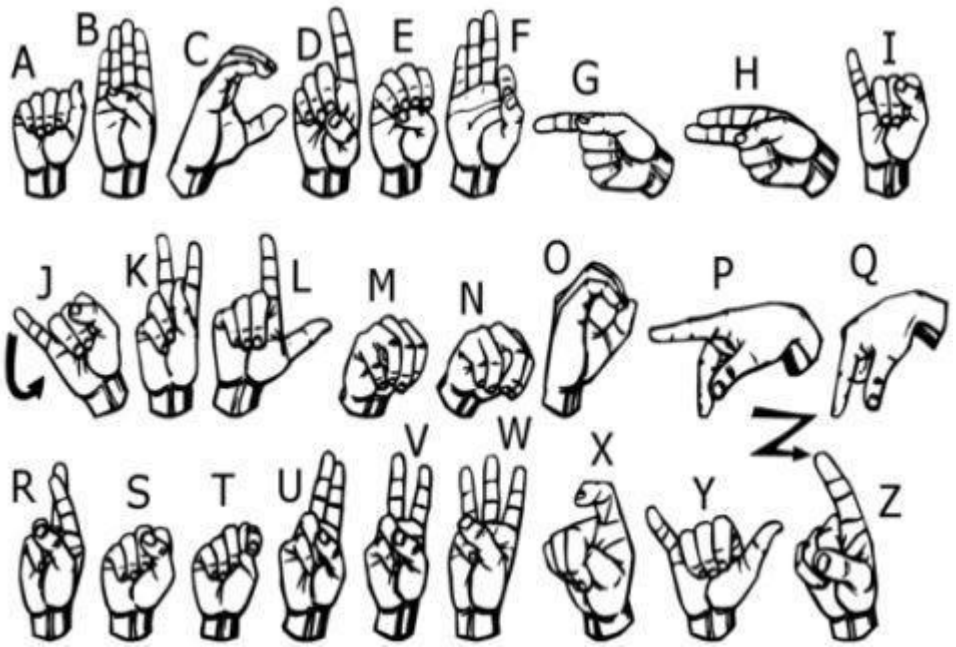

Figure 2. Finger spelling gestures

2. MOTIVATION

2. MOTIVATION

For interaction between normal people and Deaf and Mute 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-Deaf and non-Mute people. So, research has been made for a vision-based interface system where Deaf and Mute 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.

3. LITERATURE SURVEY

3. LITERATURE SURVEY

In recent years there has been tremendous research done on hand gesture recognition. With the help of literature survey, I realized that the basic steps in hand gesture recognition are:-

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

3.1 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.

3.2 DATA PRE-PROCESSING

- In [1] the approach for hand detection combines threshold-based colour detection with background subtraction. I can use AdaBoost face detector to differentiate between faces and hands as they both involve similar skin-color.
- I 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].

3.3 FEATURE EXTRACTION FOR VISION-BASED APPROACH

- For extracting the necessary image which is to be trained I can use instrumented gloves as mentioned in [4]. 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.
- I 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 I got for the segmentation I tried to do were not so great. Moreover, I 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 I decided that in order to produce better accuracies for our large number of symbols, rather than segmenting the hand out of a random background I keep background of hand a stable single colour so that I don’t need to segment it on the basis of skin colour. This would help us to get better results.

3.4 GESTURE CLASSIFICATION

- In [1] 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 [2] 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. KEYWORDS AND DEFINITIONS

4. KEYWORDS 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 ARTIFICIAL NEURAL NETWORKS (ANN):

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.

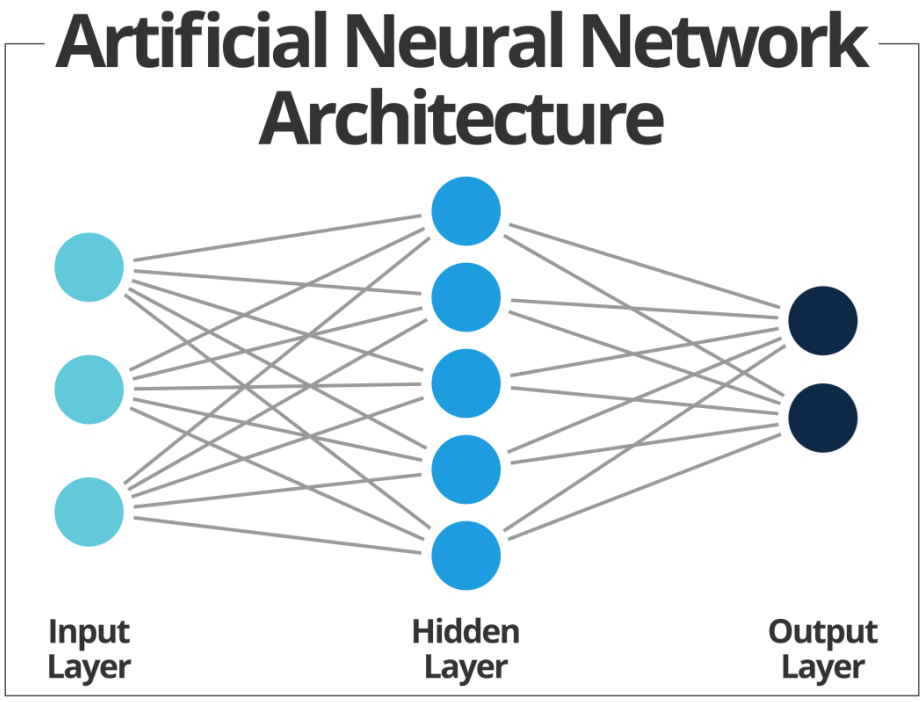

Figure 3. Artificial Neural Networks Architecture

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

1. Unsupervised Learning

2. Supervised Learning
3. Reinforcement Learning

4.3 CONVOLUTIONAL NEURAL NETWORKS (CNN):

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 I will reduce the full image into a single vector of class scores.

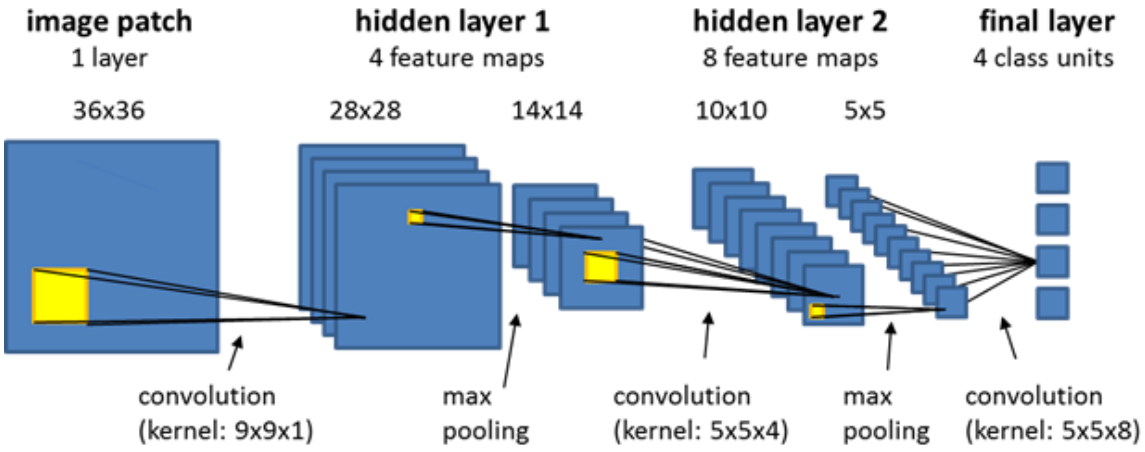

Figure 4. Convolutional Neural Networks Architecture

1. **Convolution Layer:**

   In convolution layer I 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 I slid the window by stride size [typically 1], and computed the dot product of filter entries and input values at a given position.

   As I continue this process I 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.

2. **Pooling Layer:**

I use a pooling layer to decrease the size of the activation matrix and ultimately reduce the learnable parameters. There are two types of pooling:

A. Max Pooling:

In max pooling I 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, I take advantage of all Values in a window.

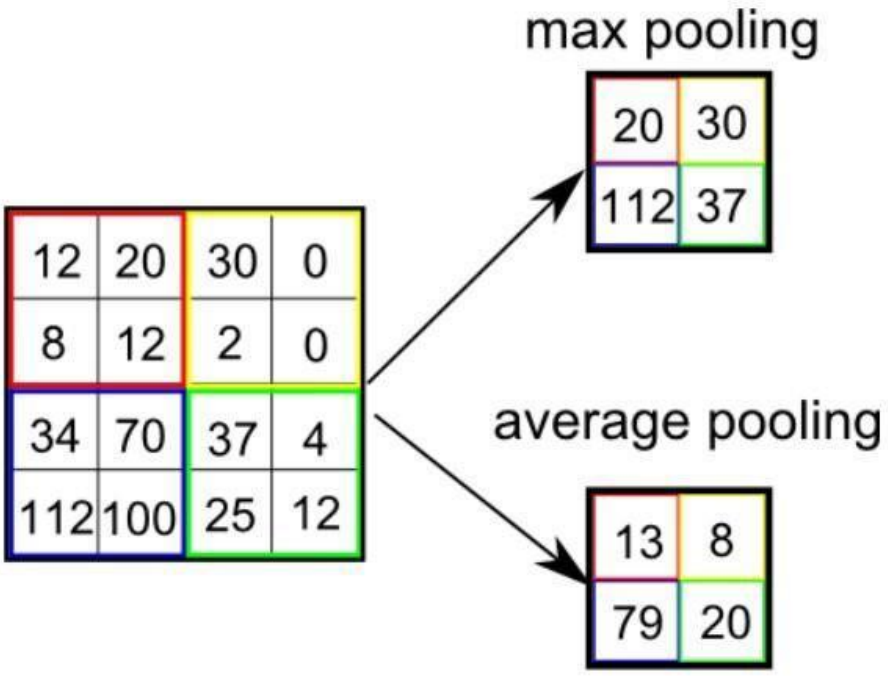

Figure 5. Activation Matrices of Pooling Layer

3. **Fully Connected Layer:**

In the convolution layer, neurons are connected only to a local region, while in a fully connected region, I will connect all the inputs to neurons.

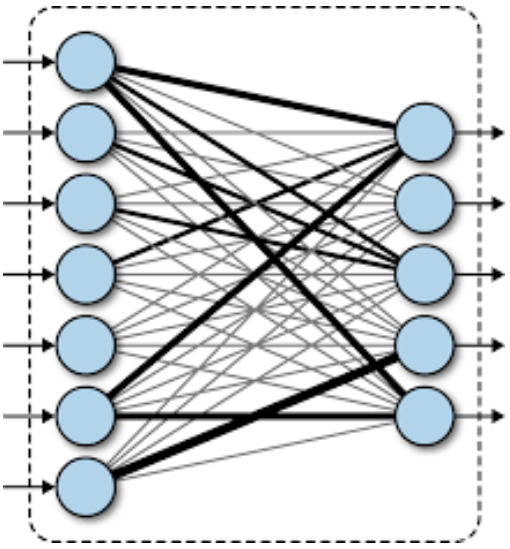

Figure 6. Fully connected layer of CNN

4. **Final Output Layer:**

   After getting values from a fully connected layer, I 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.

4.4 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.

4.5 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 I 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.

4.6 OPEN-CV:

OpenCV (Open-Source Computer Vision) is an open-source library of programming functions used for real-time computer-vision.

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.

5. METHODOLOGY

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.

5.1 DATASET GENERATION:

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

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

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

Figure 7. Image capturing using webcam

Secondly, I apply Grayscale Conversion to convert the images to grayscale. Grayscale images are represented using a single channel, where each pixel's intensity value corresponds to its brightness. This can be achieved by applying a formula that combines the original red, green, and blue (RGB) channels into a single grayscale channel. Common formulas include averaging the RGB values or using weighted combinations that mimic human perception.

I used the Open computer vision (OpenCV) library to convert an image from BGR color space to Grayscale.

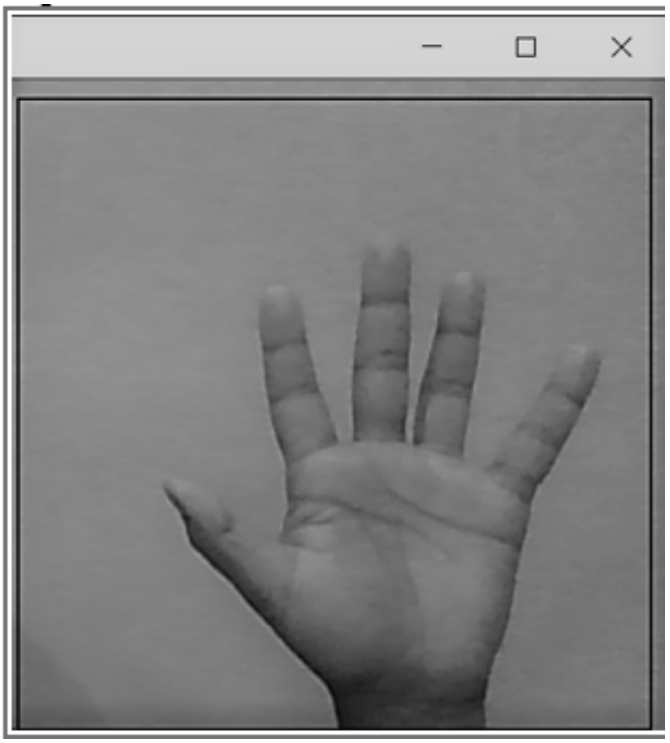

Figure 8. Applying Grayscale to Image

Then, I apply Gaussian Blur Filter to the image which helps me extract various features of our image. I used the Open computer vision (OpenCV) library to apply Gaussian Blur Filter on Grayscale images.

The images, after applying Gaussian Blur, looks as follows:

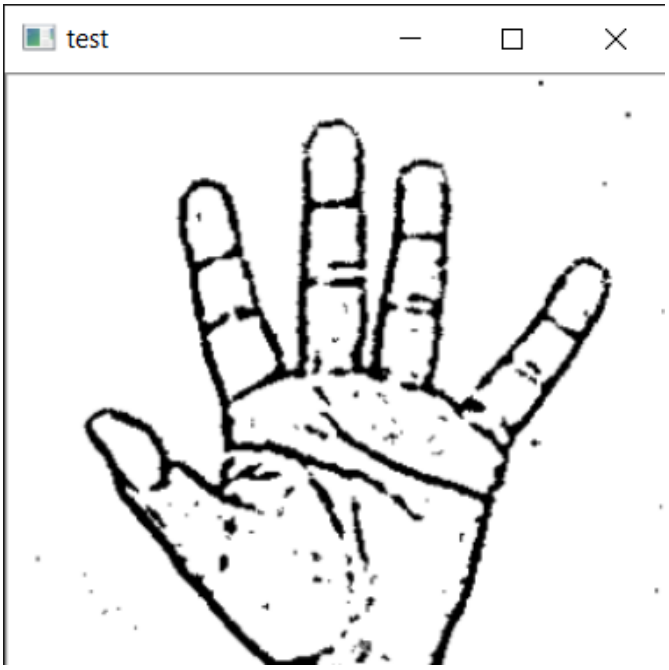

Figure 9. Applying Gaussian Blur Filter

5.2 GESTURE CLASSIFICATION

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

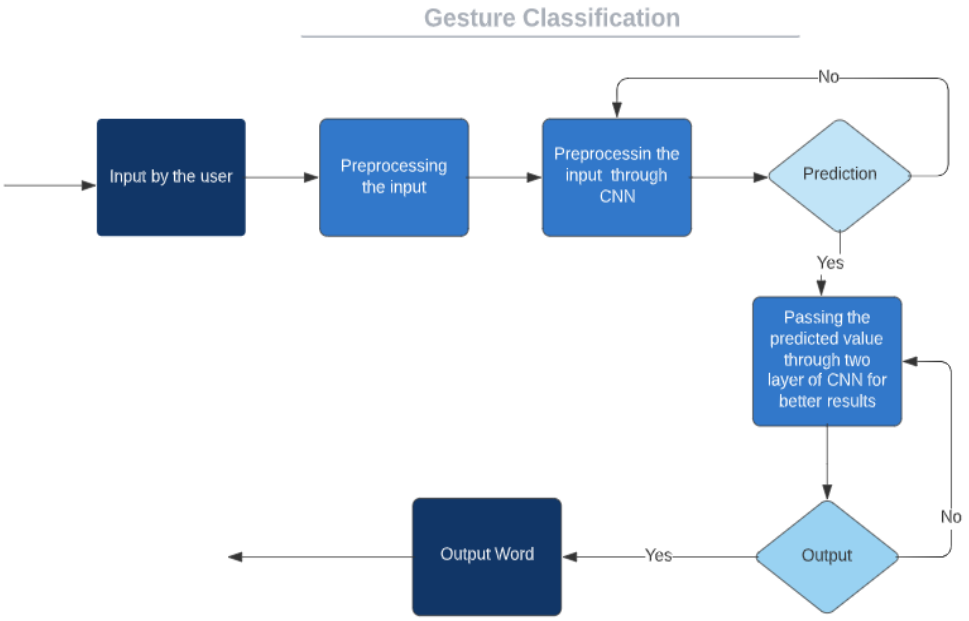

Figure 10. Gesture Classification

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.

3. Space between the words is considered using the blank symbol.

Algorithm Layer 2:

1. I detect various sets of symbols which show similar results on getting detected.

2. I 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 I 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 is served 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. I am 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 I am classifying (alphabets + blank symbol).

- **Activation Function:**

I 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.

- **Pooling Layer:**

I 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 [5].

- **Optimizer:**

I 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).

Layer 2:

I am using two layers of algorithms to verify and predict symbols which are more similar to each other so that I can get as close as I can to detect the symbol shown. In our testing I 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 I made three different classifiers for classifying these sets:

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

5.3 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, I print the letter and add it to the current string (In our code I kept the value as 50 and difference threshold as 20).
2. Otherwise, I clear the current dictionary which has the count of detections of the present symbol to avoid the probability of a wrong letter getting predicted.
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 cases it predicts the end of word by printing a space and the current gets appended to the sentence below.

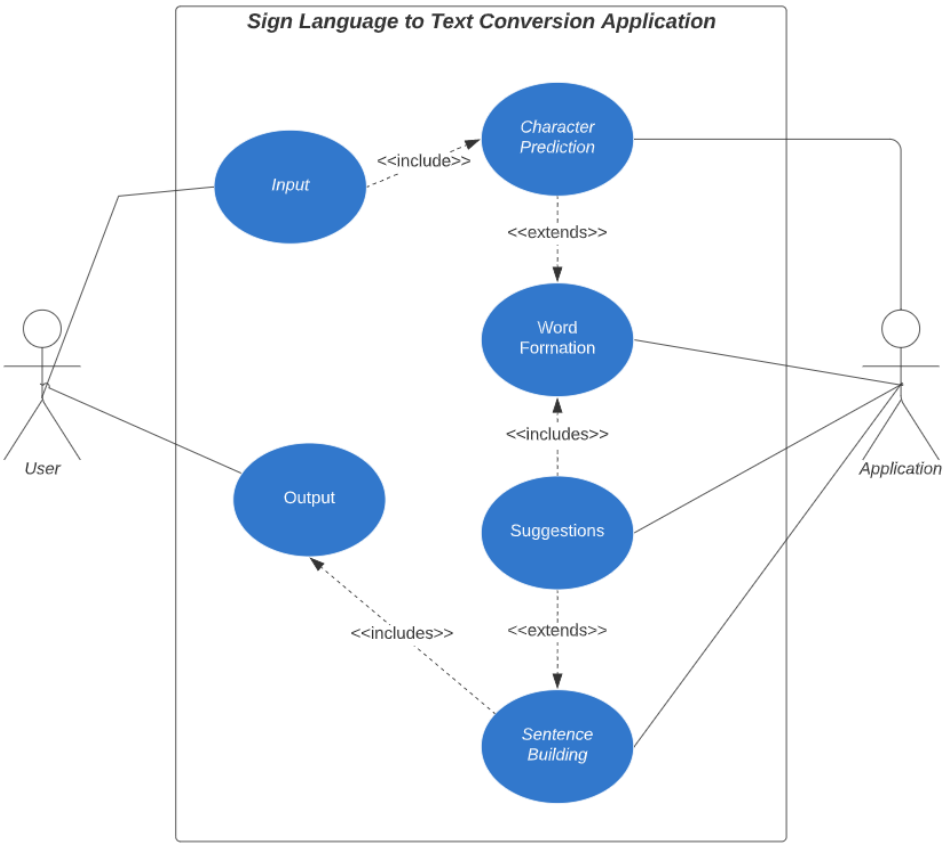

Figure 11. Sentence Formation from Fingers Spelling

5.4 AUTO-CORRECT FEATURE:

A python library **Phunspell_suggest** is used to suggest correct alternatives for each (incorrect) input word and I 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.

5.5 TRAINING AND TESTING:

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

I 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. I 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 I 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 the labelled value and is zero exactly when it is equal to the labelled value. Therefore, I optimized the cross-entropy by minimizing it as close to zero. To do this in our network layer I adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy.

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

6. CHALLENGES FACED

6. Challenges Faced:

There were many challenges faced during the project. The very first issue I faced was concerning the data set. I 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.

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

I 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 I had trained in the earlier phases. This problem was eventually improved by increasing the input image size and also by improving the data set.

7. RESULTS

7. RESULTS

I have achieved an accuracy of **95.8%** in the model using only layer 1 of our algorithm, and using the combination of **layer 1 and layer 2** I achieve an accuracy of **98.0%**, which is a better accuracy then most of the current research papers on American sign language.

Most of the research papers focus on using devices like Kinect for hand detection.

In [7] they build a recognition system for Flemish sign language using convolutional neural networks and Kinect and achieve an error rate of **2.5%.**

In [8] a recognition model is built using a hidden Markov model classifier and a vocabulary of 30 words and they achieve an error rate of **10.90%**.

In [9] they achieve an average accuracy of **86%** for 41 static gestures in Japanese sign language.

Using depth sensors map [10] achieved an accuracy of **99.99%** for observed signers and **83.58%** and **85.49%** for new signers.

They also used CNN for their recognition system. One thing should be noted that our model doesn't use any background subtraction algorithm while some of the models present above do that.

So, once I try to implement background subtraction in our project the accuracies may vary. On the other hand, most of the above projects use Kinect devices but our main aim was to create a project which can be used with readily available resources. A sensor like Kinect not only isn't readily available but also is expensive for most of the audience to buy and our model uses a normal webcam of the laptop hence it is a great plus point.

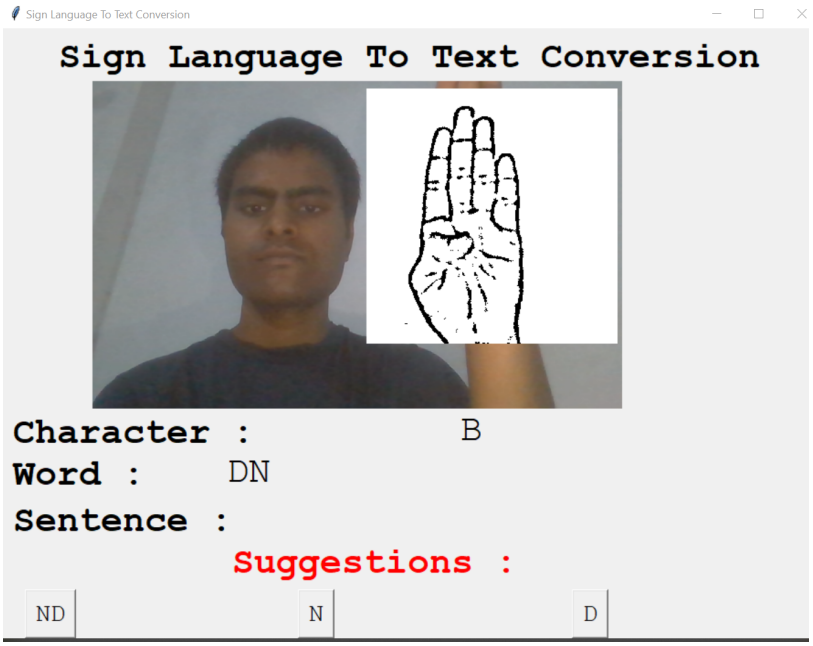

Figure 12. Output Image 1

Below are the confusion matrices for the results.

		Predicted																								
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	A	147	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0
	B	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
	C	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	145	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	1	0	0	2	10	0	0	0
C	G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
o	H	1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
r	I	0	0	0	33	0	0	0	0	108	0	2	0	0	0	0	0	0	0	0	7	1	0	0	0	0
r	J	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	K	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	L	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
t	M	0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
V	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
a	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
l	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
u	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
e	S	0	0	0	0	1	0	0	0	0	0	0	0	0	1	10	0	0	0	132	0	0	0	0	8	0
s	T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
	U	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	115	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algo 1

Figure 12. Confusion Matrix for Algorithm 1

						P	r	e	d	i	c	t	e	d		V	a	l	u	e	s					
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	A	147	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0
	B	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
	C	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	0	0	0	3	10	0	0	0
C	G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
o	H	1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
r	I	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
r	J	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	K	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	L	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
t	M	0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
V	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
a	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
l	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
u	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
e	S	0	0	0	0	1	0	0	0	0	0	0	0	0	0	10	0	0	0	133	0	0	0	0	8	0
s	T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
	U	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algo 1 + Algo 2

Figure 13. Confusion Matrix for Algorithm 1 + Algorithm 2

8. CONCLUSIONS

8. CONCLUSIONS

In this project, a functional real time vision based American Sign Language recognition for D&M people have been developed for asl alphabets.

I achieved final accuracy of **98.0%** on my data set. I have improved my prediction after implementing two layers of algorithms wherein I have verified and predicted symbols which are more similar to each other.

This gives us the ability to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

The utilization of ANN and CNN algorithms proved to be highly effective in recognizing and classifying ASL gestures, showcasing the power of machine learning in sign language to text conversion. The project's focus on ASL alphabets specifically allowed for a targeted approach, enabling accurate recognition of individual letters.

However, it is important to acknowledge that the system's performance is limited to ASL alphabets and may not be readily applicable to more complex sign language structures or other sign language variations. Further research and development are required to expand the system's capabilities and address these challenges.

9. FUTURE SCOPE

9. FUTURE SCOPE

I am planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms.

I am also thinking of improving the Pre Processing to predict gestures in low light conditions with a higher accuracy.

This project can be enhanced by being built as a web/mobile application for the users to conveniently access the project. Also, the existing project only works for ASL, it can be extended to work for other native sign languages with the right amount of data set and training. This project implements a finger spelling translator; however, sign languages are also spoken in a contextual basis where each gesture could represent an object, or verb. So, identifying this kind of a contextual signing would require a higher degree of processing and natural language processing (NLP).

10. REFERENCES

10. REFERENCES

[1] T. Yang, Y. Xu, and "A., Hidden Markov Model for Gesture Recognition", CMU-RI-TR-94 10, Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA, May 1994.

[2] Pujan Ziaie, Thomas M uller, Mary Ellen Foster, and Alois Knoll "A Na¨ive Bayes Munich, Dept. of Informatics VI, Robotics and Embedded Systems, Boltzmannstr. 3, DE-85748 Garching, Germany.

[3]https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[4] Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.

[5]aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural Networks-Part-2/

[6] http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php

[7] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham

[8] Zaki, M.M., Shaheen, S.I.: Sign language recognition using a combination of new vision-based features. Pattern Recognition Letters 32(4), 572–577 (2011).

[9] N. Mukai, N. Harada and Y. Chang, "Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning," *2017 Nicograph International (NicoInt)*, Kyoto, Japan, 2017, pp. 19-24. doi:10.1109/NICOInt.2017.9

[10] Byeongkeun Kang, Subarna Tripathi, Truong Q. Nguyen" Real-time sign language fingerspelling recognition using convolutional neural networks from depth map" 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)

[11] Number System Recognition (https://github.com/chasinginfinity/number-sign-recognition)

[12] https://opencv.org/

[13] https://en.wikipedia.org/wiki/TensorFlow

[14] https://en.wikipedia.org/wiki/Convolutional_neural_nework

[15] http://hunspell.github.io/

11. APPENDIX

APPENDIX

1. OPEN-CV:

openCV (Open-Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use.

It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications.

Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand users and an estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mine inspection, stitching maps on the web or through advanced robotics.

2. CONVOLUTIONAL NEURAL NETWORKS:

CNNs use a variation of multilayer perceptron designed to require minimal pre-processing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

3. **TENSORFLOW:**

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google brain team for internal Google use. It was released under the Apache 2.0 open-source library on November 9, 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).

TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

4. **KERAS:**

Keras is a popular open-source deep learning framework written in Python. It provides a user-friendly and intuitive interface for building, training, and deploying artificial neural networks (ANNs). With a focus on simplicity and ease of use, Keras allows both beginners and experienced researchers to efficiently develop deep learning models.

One of the key advantages of Keras is its modular and flexible architecture. It provides a high-level API that allows users to define and configure neural networks using building blocks called layers. These layers can be stacked together to create complex network architectures with ease. Keras also

supports various types of layers, including dense (fully connected), convolutional, recurrent, and more, enabling the creation of diverse network structures for different tasks.