DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING,

MANIT BHOPAL

Minor Project Report (EC-326)

Group No. 10

Project Title:-

Sign language to text using Convolutional Neural Network (CNN)

S.No.	Scholar No.	Name	Signature
1	181114009	Yash Modi	Ymodi
2	181114117	Prakhar Tripathi	Prakhar
3	181114120	Prerna Singh	
4	181114151	Abhishek Agrawal	Abhishek
5	181114039	Mohammad Anas	Anas

Name & Signature of faculty mentor: Dr. Rahul Kumar Chaurasiya

DECLARATION BY THE CANDIDATE

We hereby declare that the project work entitled **“Sign language to text using Convolutional Neural Network”** is our own work conducted under the guidance of **Dr. RAHUL KUMAR CHAURASIYA**, Assistant Professor, Department of Electronics and Communication Engineering, MANIT, Bhopal.

We further declare that to the best of our knowledge the project does not contain any part of any work that has been submitted for the award of any degree either in this institute or in any other university without proper citation.

S.No.	Scholar No.	Name	Signature
1	181114009	Yash Modi	Ymodi
2	181114117	Prakhar Tripathi	Prakhar
3	181114120	Prerna Singh	
4	181114151	Abhishek Agrawal	Abhishek
5	181114039	Mohammad Anas	Anas

This is to certify that the declaration made above by the candidates is true to the best of myknowledge.

Dr. RAHUL KUMAR CHAURASIYA
(PROJECT GUIDE)
Sr. Assistant Professor

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SESSION 2020-2021

CERTIFICATE

This is to certify that YASH MODI, PRAKHAR TRIPATHI, PRERNA SINGH, ABHISHEK AGRAWAL and MOHAMMAD ANAS of third year in B.Tech Electronics and Communication Engineering of the institute have completed their minor project and its report entitled **SIGN LANGUAGE TO TEXT USING CONVOLUTIONAL NEURAL NETWORK.**

It has been submitted in fulfillment of their minor project in Department of Electronics and Communication Engineering under the guidance of project in charge and mentor

Dr. RAHUL KUMAR CHAURASIYA.

Dr. RAHUL KUMAR CHAURASIYA

Assistant ProfessorMANIT, Bhopal

ACKNOWLEDGEMENT

Our deepest thanks to **Dr. RAHUL KUMAR CHAURASIYA**, the Guide of the project for guiding and correcting us at all stages of development of this project with attention and care.

We express our thanks to **Dr. KAVITA KHARE**, Head of the Department Electronics and Communication Engineering, for extending his support and providing us necessary facilities.

Besides our guide we would like to thank our faculty members for providing us the much- needed support and encouragement.

Lastly, we would also like to express our deep appreciation towards our classmates without whom this project would have been a distant reality.

ABSTRACT

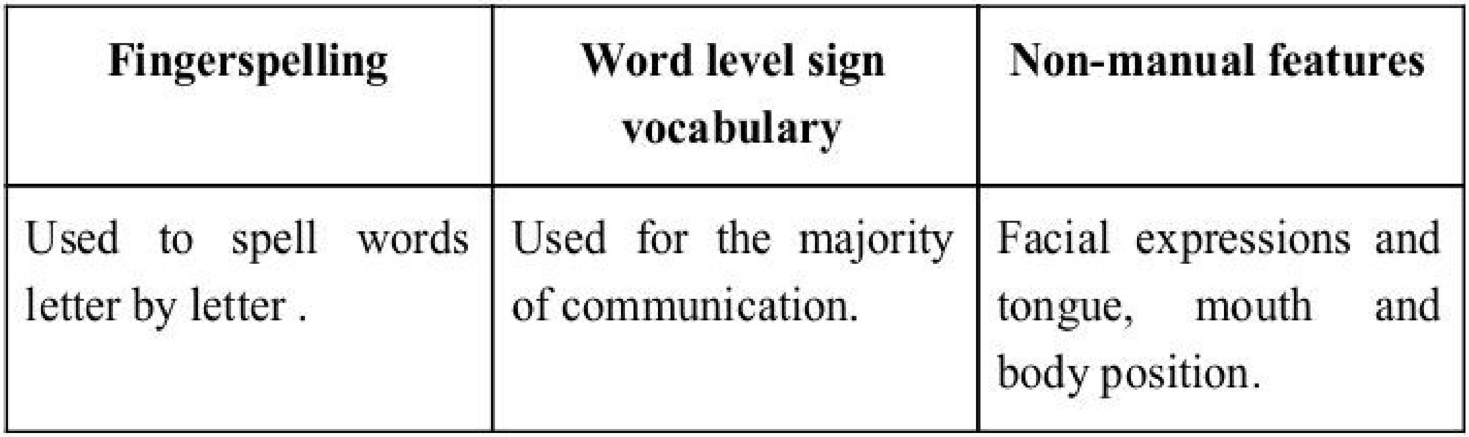
Sign language is one of the oldest and most natural form of language for communication, but since most people do not know sign language and interpreters are very difficult to come by we have come up with a real time method using neural networks for fingerspelling based american sign language. In our 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.

INDEX

INTRODUCTION

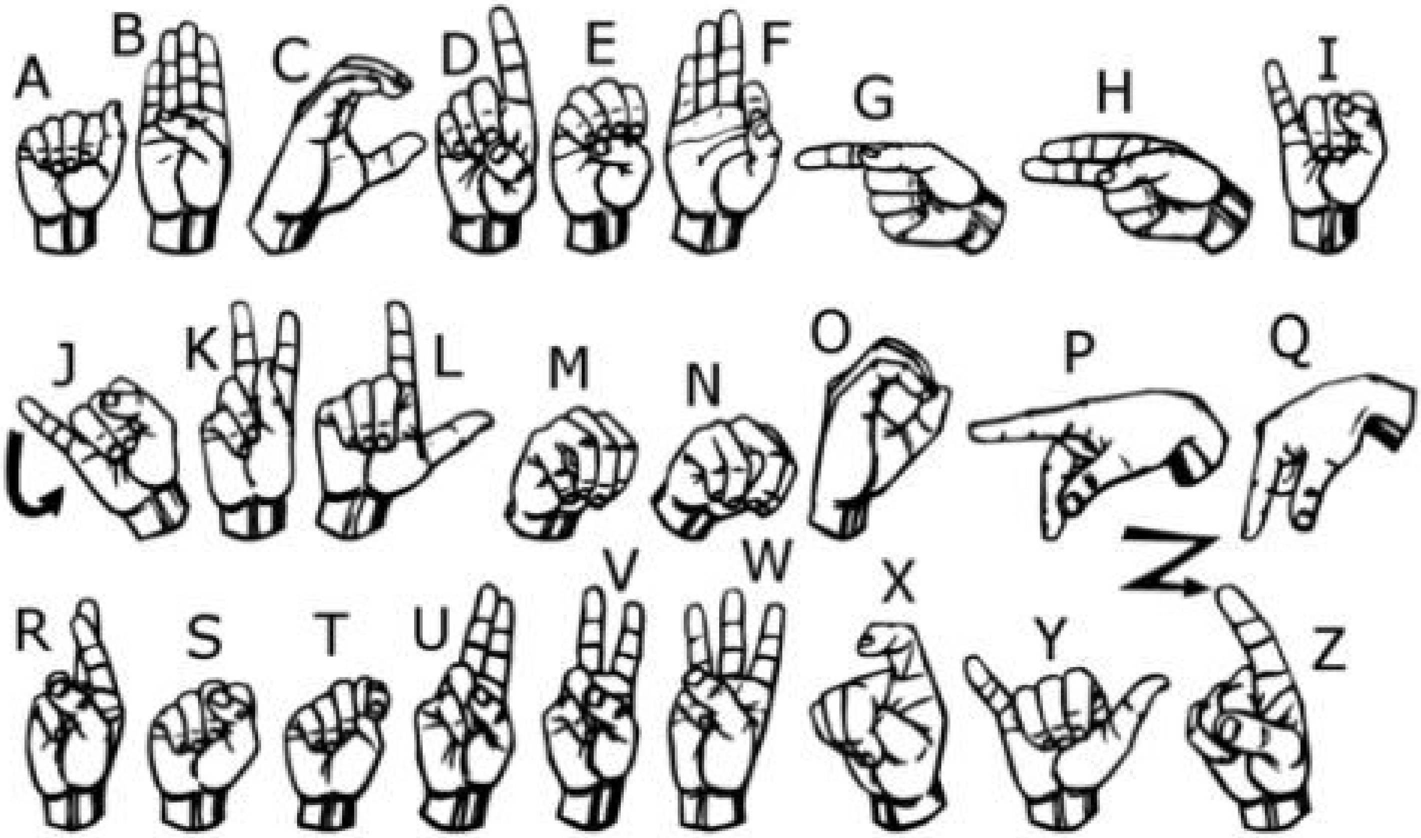
American sign language is a predominant sign language Since the only disability D&M people have is communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals. Deaf and dumb (D&M) people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

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



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.

In our project we basically 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.



OBJECTIVE OF THE PROJECT

For interaction between normal people and D&M people a language barrier is created as sign language structure which 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 the gestures can be easily understood by the other 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 interfaces (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.

DESCRIPTION

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

- Data acquisition
- Data preprocessing
- Feature extraction
- Gesture classification

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 computer camera is the input device for observing the information of hands or fingers. The Vision Basedmethods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. 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 is to cope with the large variability of human hand's appearance due to a huge number ofhand movements, to different skin-colour possibilities as well as to thevariations in view points, scales, and speed of the camera capturingthe scene.

Data preprocessing and Feature extraction for visionbased approach:

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

● We can also extract necessary image which is to be trained by applying a filter called Gaussian blur. The filter can be easily applied using open computer vision also known as OpenCV and is describedin [3].

● For extracting necessary image which is to be trained we can use instrumented gloves as mentioned in [4]. This helps reduce computation time for preprocessing and can give us more concise and accurate data compared to applying filters on data received from videoextraction.

We tried doing the hand segmentation of an image using color segmentation techniques but as mentioned in the research paper skin color and tone is highly dependent on the lighting conditions due to which output we got for the segmentation we tried to do were no 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 'K', 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 color so that we don't need to segment it on the basis of skin color . This would help us to get better results.

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-colour blobs corresponding to the hand into a body– face space centered 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 colourimetry (Y,U,V) of the skin colour pixelsin 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 theclassification 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 paper on "Human Hand Gesture Recognition Using a Convolution Neural Network" by Hsien-I Lin , Ming-Hsiang Hsu, andWei-Kai Chen graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan, they construct a skin model to extract the hand 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 center the image about it. 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 their model they produce an accuracy of around 95% for those 7 gestures.

Key Words and Definitions

Feature Extraction and Representation :

The representation of an image as a 3D matrix having dimension as ofheight 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.

Artificial Neural Networks :

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

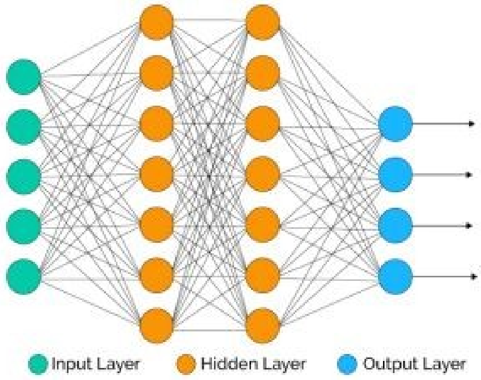


Figure 5.1: Artificial neural networks

There are capable of learning and they have to be trained. There are different learning strategies :
Unsupervised Learning
Supervised Learning
Reinforcement Learning

Convolution Neural Network :

Unlike regular Neural Networks, in the layers of CNN, the neuronsare 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.

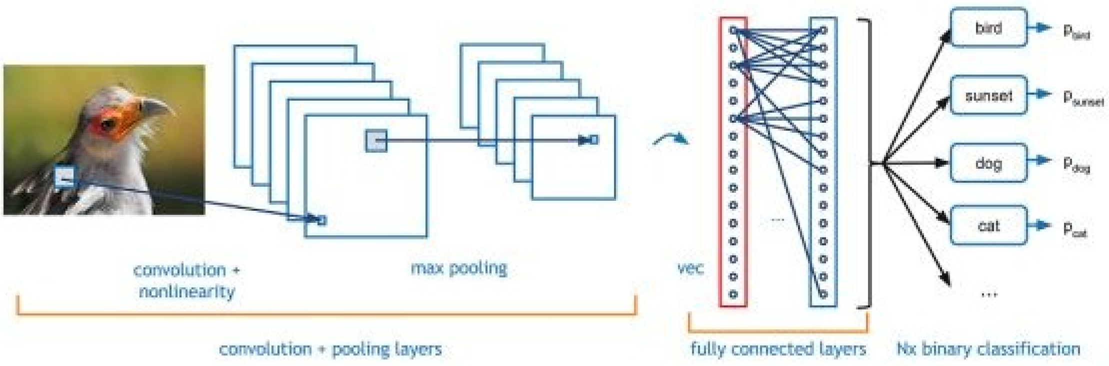


Figure 5.2: Convolution neural networks

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 consist of learnable filters of window size. During every iteration we slid 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 well 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 asan edge of some orientation or a blotch of some color

Pooling Layer : We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters.
There are two type ofpooling :
Max Pooling : In max pooling we take a window size [for examplewindow of size 2*2], and only take the maximum of 4 values. Well lidthis window and continue this process, so well finally get a activation matrix half of its original Size.
Average Pooling : In average pooling we take average of all values in a window.

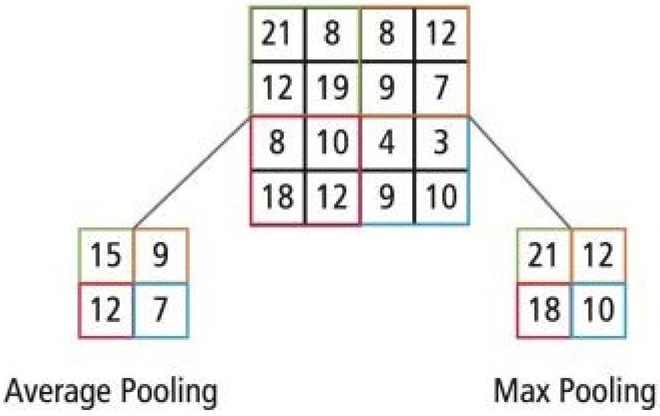


Figure 5.3: Types of pooling

1. **Fully Connected Layer :** In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

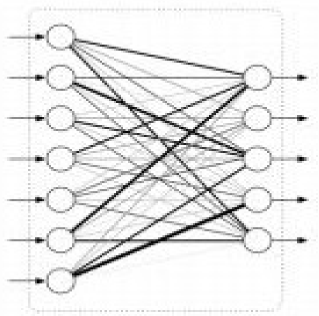


Figure 5.4: Fully Connected Layer

2.Final Output Layer : After getting values from fully connected layer, well connect them to 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.

TensorFlow :

Tensorflow is an open source software library for numerical computation. First we define the nodes of the computation graph, then inside a session, the actual computation takes place. TensorFlow is widely used in Machine Learning.

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.

OpenCV :

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.

Methodology

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

Data Set Generation

For the project we have taken our dataset from kaggle.

GESTURE CLASSIFICATION

The approach which we used for this project is :

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

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 are 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 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 downsampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, ourpicture is downsampled 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 resultin a 60 x 60 pixel image.

4. **2nd Pooling Layer :**

The resulting images are downsampled 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 anarray of 30x30x32 =28800 values. The input to this layer is an array of 28800 values. The output of these layer 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 are 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 thelayers (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 andspeeding up the training by reducing the computation time.

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 thecomputation 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.

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

We are using two layers of algorithms to verify and predict symbols which are more similar to each other so that we can get us 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 I : T, D and K
3. 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}

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 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 case it predicts the end of word by printing a space and the currentgets appended to the sentence below.

Training and Testing :

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

We feed the input images after preprocessing to our model for training and testing after applying all the operations mentioned above.

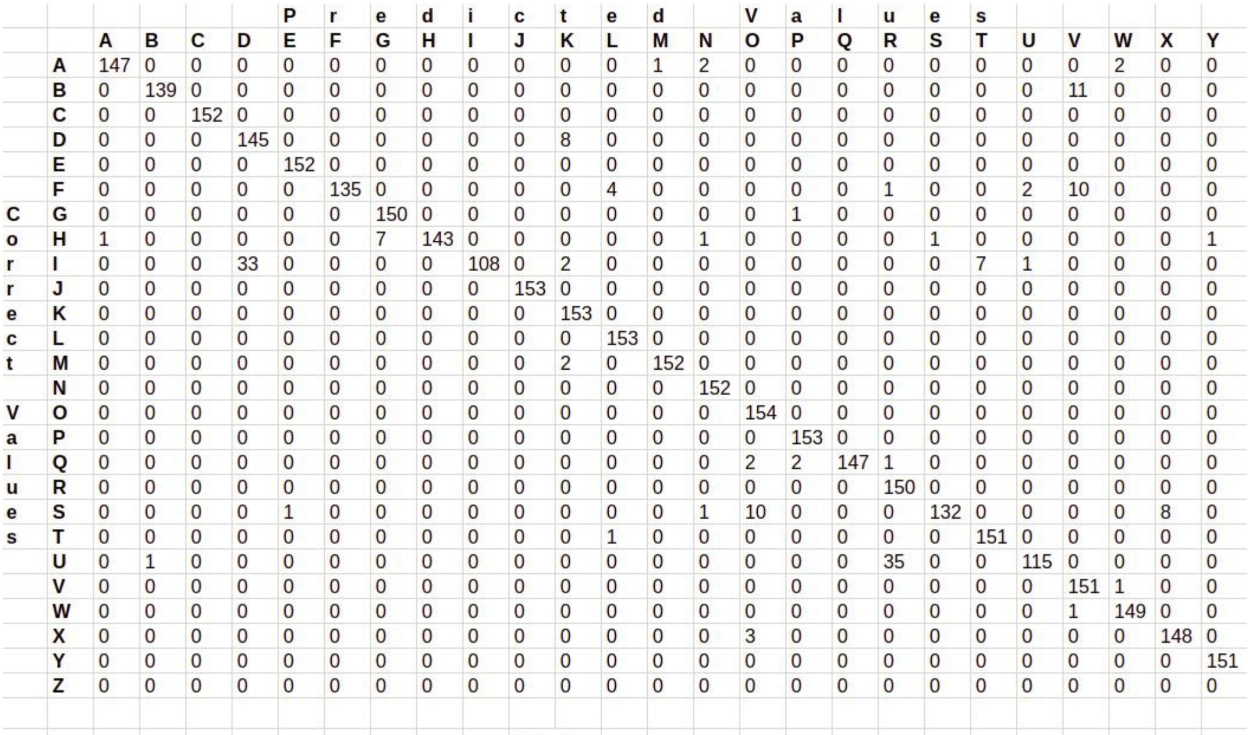
The prediction layer estimates how likely the image will fall under one of the classes. So the output is normalized between 0 and 1 and such that the sum of each values in each class sums to 1. We have achieved this using softmax function.

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 labeled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labeled value and is zero exactly when it is equal to the labeled 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.

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

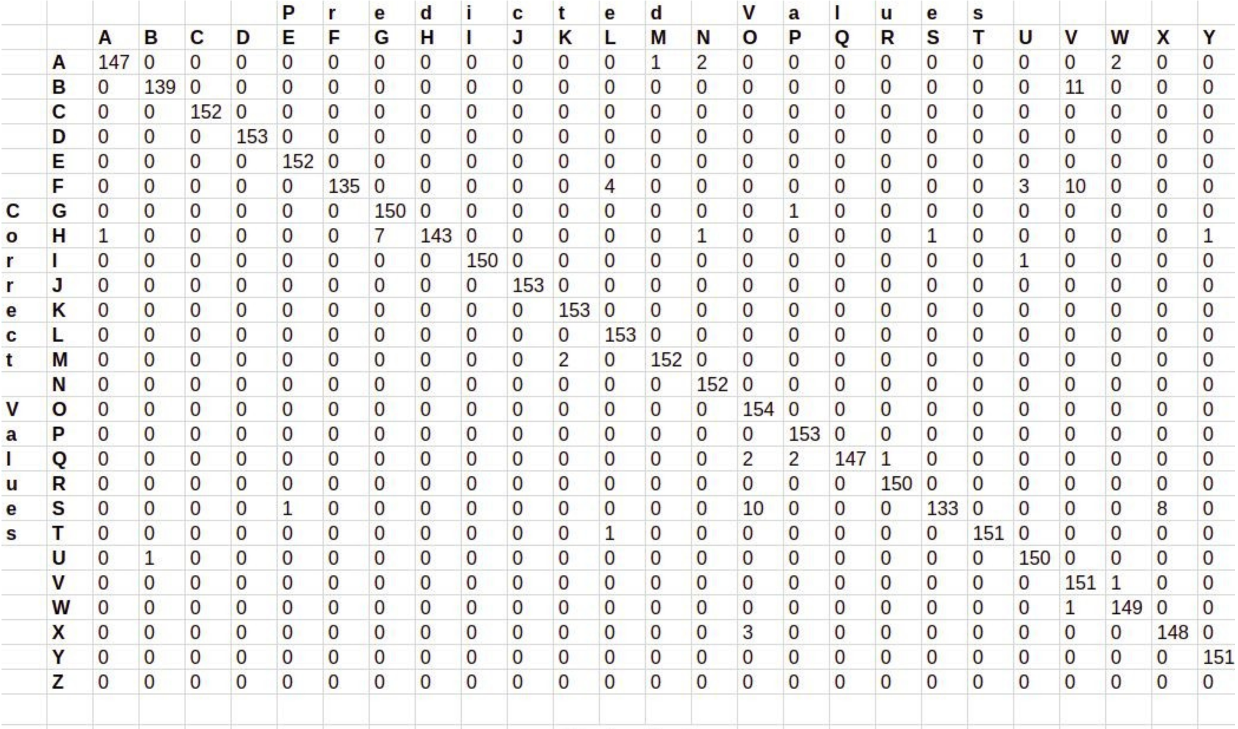
RESULT

We have achieved an accuracy of 95.8% 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 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 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 uses any background subtraction algorithm whiles some of the models present above do that. So once we 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 audience to buy and our model uses a normal webcam of the laptop hence it is great plus point.. Below are the confusion matrices for our results.



					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	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



					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

CONCLUSION

In this report, a functional real time vision based american sign language recognition for D&M people have been developed for asl alphabets.We achieved final accuracy of 98.0% on our dataset. We are able to improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other.

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

FUTURE SCOPE AND ADVANCEMENTS

We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. We are also thinking of improving the preprocessing to predict gestures in low light conditions with a higher accuracy.

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 ¨ıve 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. http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php
6. aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/
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. 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).
10. N.Mukai, N.Harada and Y.Chang, "Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning," *2017Nicograph International (NicoInt)*, Kyoto, Japan, 2017, pp. 19-24. doi:10.1109/NICOInt.2017.9

11. https://opencv.org/
12. https://en.wikipedia.org/wiki/TensorFlow
13. https://en.wikipedia.org/wiki/Convolutional_neural_network

CODE

1) Data pre-processing

```
import os
import cv2
from image_preprocessing import processing
path=os.getcwd() + "\\asl_dataset"
path1=os.getcwd()+"\\data_set"

if not os.path.exists("data_set"):
    os.makedirs("data_set")
if not os.path.exists("data_set/train"):
    os.makedirs("data_set/train")
if not os.path.exists("data_set/test"):
    os.makedirs("data_set/test")

for (dirpath,dirnames,filenames) in os.walk(path):
    for dirname in dirnames:
        print(dirname)
        for(subDirpath,subDirname,subDirfile) in os.walk(path+"\\"+dirname):
            if not os.path.exists(path1+"/train/"+dirname):
                os.makedirs(path1+"/train/"+dirname)
            if not os.path.exists(path1+"/test/"+dirname):
                os.makedirs(path1+"/test/"+dirname)

            n=0.80*(len(subDirfile))

            i=0;
            for file in subDirfile:
                actual_path=path+"/"+dirname+"/"+file
                actual_path1=path1+"/"+"train/"+dirname+"/"+file
                actual_path2=path1+"/"+"test/"+dirname+"/"+file
                final_img=processing(actual_path)
                if i<n:
                    cv2.imwrite(actual_path1,final_img)
                else:
                    cv2.imwrite(actual_path2,final_img)

                i=i+1
```

2) Model training

```
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense , Dropout
import os
import math
os.environ["CUDA_VISIBLE_DEVICES"] = "1"
sz = 128
# Step 1 - Building the CNN

# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(sz, sz, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=96, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=64, activation='relu'))
classifier.add(Dense(units=27, activation='softmax')) # softmax for more than 2

# Compiling the CNN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) #
categorical_crossentropy for more than 2


# Step 2 - Preparing the train/test data and training the model
classifier.summary()
# Code copied from - https://keras.io/preprocessing/image/
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('dataset/train',
                                                 target_size=(sz, sz),
                                                 batch_size=10,
                                                 color_mode='grayscale',
                                                 class_mode='categorical')

test_set = test_datagen.flow_from_directory('dataset/test',
                                            target_size=(sz , sz),
                                            batch_size=10,
                                            color_mode='grayscale',
                                            class_mode='categorical')

compute_step_epoch=lambda x:int(math.ceil(1.* x/10))
steps_of_epochs_training=compute_step_epoch(12845)
steps_of_epoch_test=compute_step_epoch(4268)
# Training the CNN on the Training set and evaluating it on the Test set
classifier.fit_generator(training_set,
                   steps_per_epoch = steps_of_epochs_training,#no. of images in our training set
                   epochs = 5,
                   validation_data =test_set,
                   validation_steps = steps_of_epoch_test)


model_json = classifier.to_json()
with open("model-bw.json", "w") as json_file:
    json_file.write(model_json)
print('Model Saved')
classifier.save_weights('model-bw.h5')
print('Weights saved')
```

3) GUI(using tkInter)

```
from PIL import Image, ImageTk
import tkinter as tk
import cv2
```

```
from keras.models import model_from_json
import operator
from string import ascii_uppercase

class Application:
    def __init__(self):
        self.directory = 'model'
        self.vs = cv2.VideoCapture(0)
        self.current_image = None
        self.current_image2 = None

        self.json_file = open(self.directory+"/model-bw.json", "r")
        self.model_json = self.json_file.read()
        self.json_file.close()
        self.loaded_model = model_from_json(self.model_json)
        self.loaded_model.load_weights(self.directory+"/model-bw.h5")



        self.ct = {}
        self.ct['blank'] = 0
        self.blank_flag = 0
        for i in ascii_uppercase:
          self.ct[i] = 0
        print("Loaded model from disk")
        self.root = tk.Tk()
        self.root.title("Sign language to Text Converter")
        self.root.protocol('WM_DELETE_WINDOW', self.destructor)
        self.root.geometry("900x700")
        self.panel = tk.Label(self.root)
        self.panel.place(x = 135, y = 100, width = 640, height = 360)
        self.panel2 = tk.Label(self.root) # initialize image panel
        self.panel2.place(x = 465, y = 100, width = 310, height = 210)

        self.T = tk.Label(self.root)
        self.T.place(x=150,y = 0)
        self.T.config(text = "Sign Language to Text",font=("courier",30,"italic"))
        self.panel3 = tk.Label(self.root) # Current SYmbol
        self.panel3.place(x = 200,y=600)
        self.T1 = tk.Label(self.root)
        self.T1.place(x = 10,y = 600)
        self.T1.config(text="Character :",font=("Courier",20,"bold"))
        self.panel4 = tk.Label(self.root) # Word
        self.panel4.place(x = 200,y=630)
        self.T2 = tk.Label(self.root)
        self.T2.place(x = 10,y = 630)
        self.T2.config(text ="Word :",font=("Courier",20,"bold"))
        self.panel5 = tk.Label(self.root) # Sentence
        self.panel5.place(x = 200,y=660)
        self.T3 = tk.Label(self.root)
        self.T3.place(x = 10,y = 660)
        self.T3.config(text ="Sentence :",font=("Courier",20,"bold"))

        self.btcall = tk.Button(self.root,command = self.action_call,height = 0,width = 0)
        self.btcall.config(text = "About",font = ("Courier",14))
        self.btcall.place(x = 0, y = 0)

        self.str=""
        self.word=""
        self.current_symbol="Empty"
        self.photo="Empty"
        self.video_loop()





    def video_loop(self):
        ok, frame = self.vs.read()
        if ok:
            cv2image = cv2.flip(frame, 1)
            x1 = int(0.5*frame.shape[1])
            y1 = 10
            x2 = frame.shape[1]-10
            y2 = int(0.5*frame.shape[1])
            cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
            cv2image = cv2.cvtColor(cv2image, cv2.COLOR_BGR2RGBA)
            self.current_image = Image.fromarray(cv2image)
            imgtk = ImageTk.PhotoImage(image=self.current_image)
            self.panel.imgtk = imgtk
            self.panel.config(image=imgtk)
            cv2image = cv2image[y1:y2, x1:x2]
            gray = cv2.cvtColor(cv2image, cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray,(5,5),2)
            th3 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINAR
Y_INV,11,2)
            ret, res = cv2.threshold(th3, 70, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
            self.predict(res)
            self.current_image2 = Image.fromarray(res)
            imgtk = ImageTk.PhotoImage(image=self.current_image2)
            self.panel2.imgtk = imgtk
            self.panel2.config(image=imgtk)
            self.panel3.config(text=self.current_symbol,font=("Courier",20))
            self.panel4.config(text=self.word,font=("Courier",20))
            self.panel5.config(text=self.str,font=("Courier",20))

        self.root.after(30, self.video_loop)
    def predict(self,test_image):
        test_image = cv2.resize(test_image, (128,128))
        result = self.loaded_model.predict(test_image.reshape(1, 128, 128, 1))

        prediction={}
        prediction['blank'] = result[0][0]
        inde = 1
        for i in ascii_uppercase:
            prediction[i] = result[0][inde]
            inde += 1
        #LAYER 1
        prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
        self.current_symbol = prediction[0][0]


        if(self.current_symbol == 'blank'):
            for i in ascii_uppercase:
                self.ct[i] = 0
        self.ct[self.current_symbol] += 1
        if(self.ct[self.current_symbol] > 60):
            for i in ascii_uppercase:
                if i == self.current_symbol:
                    continue
                tmp = self.ct[self.current_symbol] - self.ct[i]
                if tmp < 0:
                    tmp *= -1
                if tmp <= 20:
                    self.ct['blank'] = 0
                    for i in ascii_uppercase:
                        self.ct[i] = 0
                    return
            self.ct['blank'] = 0
            for i in ascii_uppercase:
                self.ct[i] = 0
            if self.current_symbol == 'blank':
                if self.blank_flag == 0:
                    self.blank_flag = 1
                    if len(self.str) > 0:
                        self.str += " "
                    self.str += self.word
                    self.word = ""
            else:
                if(len(self.str) > 16):
                    self.str = ""
                self.blank_flag = 0
                self.word += self.current_symbol

    def destructor(self):
        print("Closing Application...")
        self.root.destroy()
        self.vs.release()
        cv2.destroyAllWindows()

    def destructor1(self):
        print("Closing Application...")
        self.root1.destroy()

    def action_call(self) :

        self.root1 = tk.Toplevel(self.root)
        self.root1.title("About")
        self.root1.protocol('WM_DELETE_WINDOW', self.destructor1)
        self.root1.geometry("900x900")

        # img = cv2.imread("Pictures/sir.jpg", 1)
        # # img = cv2.resize(img, (300, 300))
        # cv2.imwrite("Pictures/sir.png", img)
        # return
        self.tx = tk.Label(self.root1)
        self.tx.place(x = 330,y = 20)
        self.tx.config(text = "Efforts By", fg="red", font = ("Courier",30,"bold"))
```

```
self.photo1 = tk.PhotoImage(file='Pictures/yash.png')
self.w1 = tk.Label(self.root1, image = self.photo1)
self.w1.place(x = 20, y = 105)
self.tx6 = tk.Label(self.root1)
self.tx6.place(x = 20,y = 250)
self.tx6.config(text = "Yash\n181114009", font = ("Courier",15,"bold"))

self.photo2 = tk.PhotoImage(file='Pictures/abhishek.png')
self.w2 = tk.Label(self.root1, image = self.photo2)
self.w2.place(x = 200, y = 105)
self.tx2 = tk.Label(self.root1)
self.tx2.place(x = 200,y = 250)
self.tx2.config(text = "Abhishek\n181114151", font = ("Courier",15,"bold"))

self.photo3 = tk.PhotoImage(file='Pictures/prakhar.png')
self.w3 = tk.Label(self.root1, image = self.photo3)
self.w3.place(x = 380, y = 105)
self.tx3 = tk.Label(self.root1)
self.tx3.place(x = 380,y = 250)
self.tx3.config(text = "Prakhar\n181114117", font = ("Courier",15,"bold"))

self.photo4 = tk.PhotoImage(file='Pictures/prerna.png')
self.w4 = tk.Label(self.root1, image = self.photo4)
self.w4.place(x = 560, y = 105)
self.tx4 = tk.Label(self.root1)
self.tx4.place(x = 560,y = 250)
self.tx4.config(text = "Prerna\n181114120", font = ("Courier",15,"bold"))

self.photo5 = tk.PhotoImage(file='Pictures/anas.png')
self.w5 = tk.Label(self.root1, image = self.photo5)
self.w5.place(x = 740, y = 105)
self.tx5 = tk.Label(self.root1)
self.tx5.place(x = 740,y = 250)
self.tx5.config(text = "Anas\n181114039", font = ("Courier",15,"bold"))

self.tx7 = tk.Label(self.root1)
self.tx7.place(x = 170,y = 360)
self.tx7.config(text = "Under the supervision of", fg="red", font = ("Courier",30,"bold"))

self.photo6 = tk.PhotoImage(file='Pictures/sir.png')
self.w6 = tk.Label(self.root1, image = self.photo6)
self.w6.place(x = 350, y = 420)
self.tx6 = tk.Label(self.root1)
```

```
        self.tx6.place(x = 230,y = 670)
        self.tx6.config(text = "Dr. R. K. Chaurasiya\nAsst. Professor, Dept. of ECE\nMANIT, Bhopal", 
font = ("Courier",20,"bold"))


print("Starting Application...")
pba = Application()
pba.root.mainloop()
```

CONTRIBUTION

- PRAKHAR TRIPATHI (181114117)- GUI, Model and Accuracy
- YASH MODI (181114009)- GUI, Model and Accuracy
- PRERNA SINGH (181114120)- Data Preprocessing and Report
- ABHISHEK AGRAWAL (181114151)- GUI, Model and Accuracy
- MOHAMMAD ANAS (181114039)- Data Mining and Presentation