## https://lh4.googleusercontent.com/wdFd5XalclUdvw3R4qHqQqSTyROdREQpx09rLm_Hiu8dai7R13UvGQrxphZ29b_oTAVr3kU_ZwEtA_w2VFL2XPgvg0pCiPoOP28ZcO2ibUmY56YSnlDk0jJd2g2utlyj5HAJ1UUy9D5UctbJddxD2tQKAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A PROJECT REPORT ON

**“MACHINE LEARNING BASED REGRESSION FRAMEWORK TO PREDICT SIGN DETECTION**

SUBMITTED TO

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA**

**A Project Report Submitted in partial fulfilment of the requirements Forthe Award of degree of**

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted By

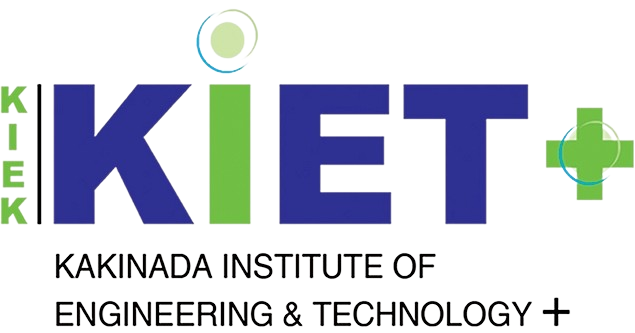
|  |  |
| --- | --- |
| **JONNALAGADDA SAMPATH** | **(206Q1A4401)** |
| **M.V.MALLESWARA REDDY** | **(206Q1A4414)** |
| **GURUJALA DINESH** | **(206Q1A4409)** |
| **KAKARLA REKHA SRI** | **(206Q1A4416)** |
|  |  |

#### Under the Esteemed Guidance of

#### Mr. V.SUBASH, M.Tech.

**Assistant Professor**

**Department of Computer Science and Engineering**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II

**(Approved by AICTE, Govt. of AP. & Affiliated to JNT University Kakinada)**

**Yanam Road, Korangi- 533461 E.G Dist. (A.P) 2020-2024**

## KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A PROJECT REPORT ON

**“MACHINE LEARNING BASED REGRESSION FRAMEWORK TO SIGN LANGUAGE DETECTION**

SUBMITTED TO

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA**

**A Project Report Submitted in partial fulfilment of the requirements Forthe Award of degree of**

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

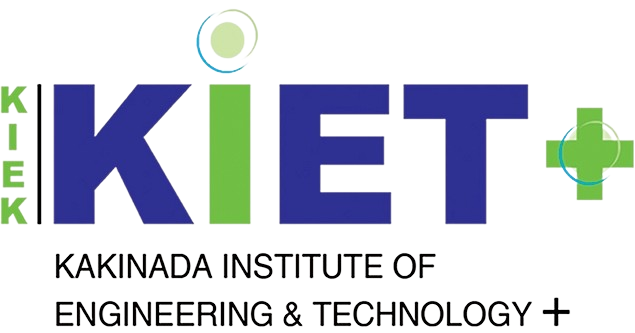
Submitted By

|  |  |
| --- | --- |
| **JONNALAGADDA SAMPATH** | **(206Q1A4401)** |
| **M.V.MALLESWARA REDDY** | **(206Q1A4414)** |
| **GURUJALA DINESH** | **(206Q1A4409)** |
| **KAKARLA REKHA SRI** | **(206Q1A4416)** |

#### Under the Esteemed Guidance of Mr. B. V.SUBASH, M.Tech.

**Assistant Professor**

**Department of Computer Science and Engineering**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

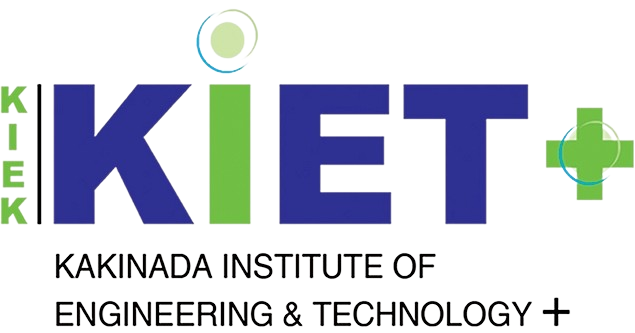
## KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II

**(Approved by AICTE, Govt. of AP. & Affiliated to JNT University Kakinada)**

**Yanam Road, Korangi- 533461 E.G Dist. (A.P).**

**2020-2024**

## KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY-II

**(Approved by AICTE, Govt. of AP. & Affiliated to JNT University Kakinada)Yanam Road, Korangi- 533461 E.G Dist. (A. P) DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the project entitled **“MACHINE LEARNING BASED TO UNDERSTAND THE SIGN LANGUAGE**” is bonafide work of **JONNALAGADDA SAMPATH (206Q1A4401), M.V.MALLESWARA REDDY (206Q1A4414), GURUJALA DINESH (206Q1A4409), KAKARLA REKHA SRI (206Q1A4416**

in Partial Fulfillment of the requirements for the award of the Degree Of Bachelor of Technology in Computer Science & Engineering in the academic year 2023-2024 from Kakinada Institute of Engineering And Technology-II affiliated to Jawaharlal Nehru Technological University Kakinada, is a record of Bonafide work carried out by them under my guidance and supervision.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

|  |  |
| --- | --- |
| **INTERNAL SUPERVISIOR** | **HEAD OF THE DEPARTMENT** |

**EXTERNAL EXAMINER**

## DECLARATION

We hereby declare that the project entitled **MACHINE LEARNING BASED TO UNDERSTAND THE SIGN LANGUAGE**

is the work done by us during the academic year 2023-2024, and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** from **KAKINADA INSTITUTE OF**

### ENGINEERING AND TECCHNOLOGY-II affiliated to Jawaharlal Nehru Technological University Kakinada,

We also declare that this project is the outcome of our own effort, that it has not been submitted to any other university for the award of any degree

BY

|  |  |
| --- | --- |
| **JONNALAGADDA SAMPATH** | **(206Q1A4401)** |
| **M.V.MALLESWARA REDDY** | **(206Q1A4414)** |
| **GURUJALA DINESH** | **(206Q1A4409)** |
| **KAKARLA REKHA SRI** | **(206Q1A4416)** |

## ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to our honorable chairman, SRI. P. VENKATA VISWAM, for the guidance and advice that has been given and for providing sufficient resources.

We wish to take this opportunity to thank Dr. CH BHAVANARAYANA, Ph.D., Principal, Kakinada Institute of Engineering and Technology, for his continuous support and giving valuable suggestions during the entire period of the project work.

We extend our sincere thanks and deep sense of gratitude to Mr. B. Mahalakshmi Rao, M.Tech., Associate Professor, Head of the Department CSE, for his valuable guidance inthe successful completion of this project.

We express a great deal of pleasure to acknowledge my profound sense of gratitude to our project guide, Mr. B. V. Subash, M.Tech., Assistant Professor in the CSE Department, for his valuable guidance, comments, suggestions, and encouragement throughout the course of this project.

We also extend our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

**PROJECT ASSOCIATES**

|  |  |
| --- | --- |
| **JONNALAGADDA SAMPATH** | **(206Q1A4401)** |
| **M.V.MALLESWARA REDDY** | **(206Q1A4414)** |
| **GURUJALA DINESH** | **(206Q1A4409)** |
| **KAKARLA REKHA SRI** | **(206Q1A4416)** |

**Contents**

[Abstract 3](#_TOC_250008)

[Introduction 4](#_TOC_250007)

[Motivation 6](#_TOC_250006)

Literature Review 7

Keywords and Definition 10

Methodologies 14

[Challenges faced 21](#_TOC_250005)

[Results 21](#_TOC_250004)

[Conclusion 24](#_TOC_250003)

[Future Scope 24](#_TOC_250002)

[References 25](#_TOC_250001)

[Appendix 27](#_TOC_250000)

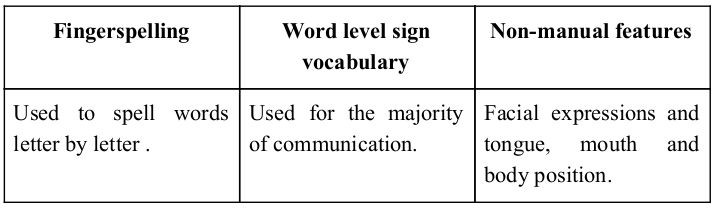
# Abstract

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, we have come up with a real-time method using neural networks for finger-spelling-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.

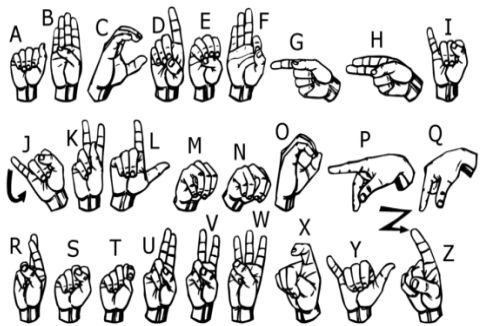
# Introduction

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

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



In our project, we basically focus on producing a model that can recognize finger-spelling-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.



# Motivation

For interaction between normal people and D&M people, a language barrier is created by the 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 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 interface (HCI) where the computer understands 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, and Japanese Sign Language, and work has been done on other languages all around the world.

**Literature survey**

##### In recent years, there has been tremendous research done on hand gesture recognition.

##### With the help of a literature survey, 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:

### 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.

### Vision based approach

In vision-based methods, a computer camera is the input device for observing the information of hands 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. 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 the human hand’s appearance due to a huge number of hand movements, different skin-color possibilities, and variations in view points, scales, and speed of the camera capturing the scene.

### Data preprocessing and Feature extraction for vision based approach:

* In [1], the approach for hand detection combines threshold-based color detection with background subtraction. We can use Adaboost face detectors to differentiate between faces and hands, as both involve similar skin colors..
* 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 described in [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 video extraction.
* 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 digit ‘2’, hence we decided that in order to produce better accuracies for our large number of symbols, rather than

segmenting the hand out of a random background we keep background of hand a stable single 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 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 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 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.

.

### 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.

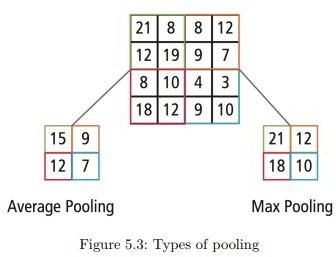
### Artificial Neural Networks :

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.

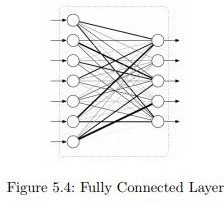
1. **Convolution Layer :** In convolution layer we take a small window size [typically of length 5\*5] that extends to the depth of the input matrix. The layer consist of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product 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 as an edge of some orientation or a blotch of some color
2. **Pooling Layer :** We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two type of pooling :
   1. **Max Pooling :** In max pooling we take a window size [for example window of size 2\*2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get a activation matrix half of its original Size.
   2. **Average Pooling :** In average pooling we take average of all

values

in a window.



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.



1. **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.

### 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**

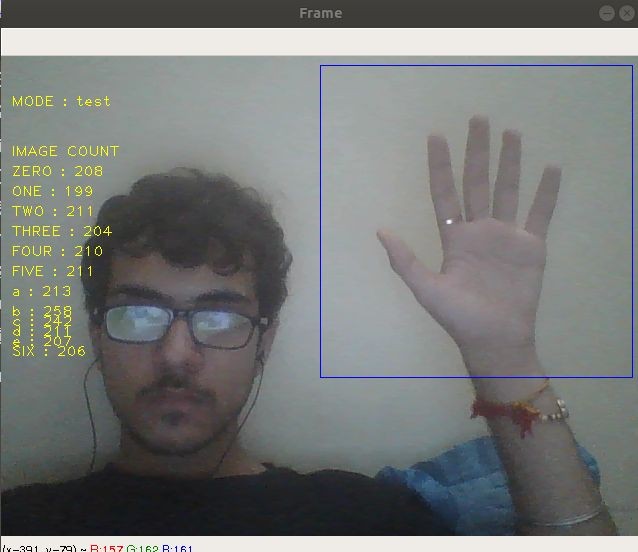
The system is a vision based approach. All the signs are represented with bare hands and so it eliminates the problem of using any artificial devices for interaction.

### Data Set Generation

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

We used Open computer vision(OpenCV) library in order to produce our dataset.Firstly we captured around 800 images of each of the symbol in ASL for training purposes and around 200 images per symbol for testing purpose.

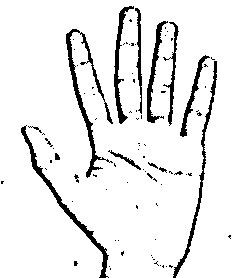
First we capture each frame shown by the webcam of our machine. In the each frame we define a region of interest (ROI) which is denoted by a blue bounded square as shown in the image below.



From this whole image we extract our ROI which is RGB and convert it into gray scale Image as shown below.



Finally we apply our gaussian blur filter to our image which helps us extracting various features of our image. The image after applying gaussian blur looks like below.



### 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, our picture is downsampled to 63x63 pixels.

###### 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.

###### 2nd Pooling Layer :

The resulting images are downsampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.

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

###### 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.

###### Final layer:

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

### Activation Function :

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

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

### Pooling Layer :

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

### Dropout Layers:

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn’t perform well when given new examples.This layer “drops out” a random set of activations in that layer by setting them to zero.The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out[5].

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

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

1. {D,R,U}

2. {T,K,D,I}

3. {S,M,N}

### 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).

1. 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.
2. Whenever the count of a blank(plain background) detected exceeds a specific value and if the current buffer is empty no spaces are detected.
3. In other case it predicts the end of word by printing a space and the current gets appended to the sentence below.

#### Autocorrect Feature :

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

### Training and Testing :

We convert our input images(RGB) into grayscale and apply gaussian blur to remove unnecessary noise.We apply adaptive threshold to extract our hand 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.

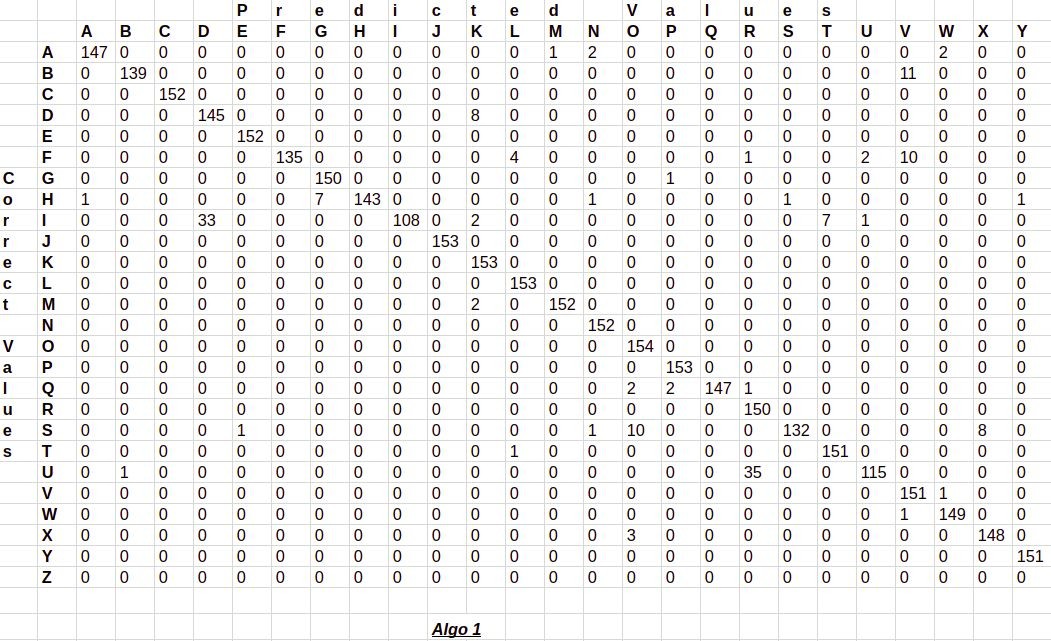
# Challenges Faced :

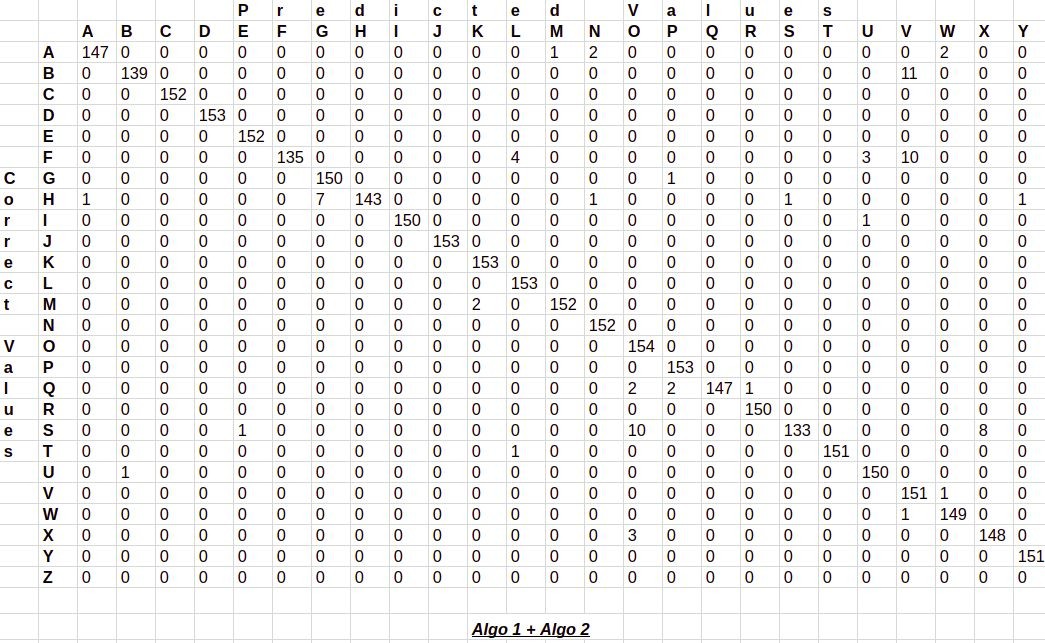
There were many challenges faced by us during the project. The very first issue we faced was of dataset. We wanted to deal with raw images and that too square images as CNN in Keras as it was a lot more convenient working with only square images. We couldn’t find any existing dataset for that hence we decided to make our own dataset. Second issue was to select a filter which we could apply on our images so that proper features of the images could be obtained and hence then we could provided that image as input for CNN model. We tried various filter including binary threshold, canny edge detection, gaussian blur etc. but finally we settled with gaussian blur filter. More issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by increasing the input image size and also by improving the dataset.

# Results :

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.





# 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 :

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.

* 1. 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.
  2. https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian\_median\_blur\_b ilateral\_filter/gausian\_median\_blur\_bilateral\_filter.html
  3. Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.
  4. aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convol utional-Neural-Networks-Part-2/
  5. <http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php>
  6. 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

* 1. 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)
  2. 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
  3. 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)
  4. <https://opencv.org/>
  5. <https://en.wikipedia.org/wiki/TensorFlow> [13]<https://en.wikipedia.org/wiki/Convolutional_neural_network>

# APPENDIX

### OpenCV

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 people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

### Convolution Neural network

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. 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.

### 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.

**Test.py :**

import cv2

from cvzone.HandTrackingModule import HandDetector

from cvzone.ClassificationModule import Classifier

import numpy as np

import math

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)

classifier = Classifier("Model/keras\_model.h5", "Model/labels.txt")

offset = 20

imgSize = 300

folder = "Data/C"

counter = 0

labels = ["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"]

while True:

    success, img = cap.read()

    imgOutput = img.copy()

    hands, img = detector.findHands(img)

    if hands:

        hand = hands[0]

        x, y, w, h = hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) \* 255

        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        imgCropShape = imgCrop.shape

        aspectRatio = h / w

        if aspectRatio > 1:

            k = imgSize / h

            wCal = math.ceil(k \* w)

            imgResize = cv2.resize(imgCrop, (wCal, imgSize))

            imgResizeShape = imgResize.shape

            wGap = math.ceil((imgSize - wCal) / 2)

            imgWhite[:, wGap:wCal + wGap] = imgResize

            prediction, index = classifier.getPrediction(imgWhite, draw=False)

            print(prediction, index)

        else:

            k = imgSize / w

            hCal = math.ceil(k \* h)

            imgResize = cv2.resize(imgCrop, (imgSize, hCal))

            imgResizeShape = imgResize.shape

            hGap = math.ceil((imgSize - hCal) / 2)

            imgWhite[hGap:hCal + hGap, :] = imgResize

            prediction, index = classifier.getPrediction(imgWhite, draw=False)

        cv2.rectangle(imgOutput, (x - offset, y - offset-50),

                      (x - offset+90, y - offset-50+50), (255, 0, 255), cv2.FILLED)

        cv2.putText(imgOutput, labels[index], (x, y -26), cv2.FONT\_HERSHEY\_COMPLEX, 1.7, (255, 255, 255), 2)

        cv2.rectangle(imgOutput, (x-offset, y-offset),

                      (x + w+offset, y + h+offset), (255, 0, 255), 4)

        cv2.imshow("ImageCrop", imgCrop)

        cv2.imshow("ImageWhite", imgWhite)

    cv2.imshow("Image", imgOutput)

    cv2.waitKey(1)

dataCollection.py

import cv2

from cvzone.HandTrackingModule import HandDetector

import numpy as np

import math

import time

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)

offset = 20

imgSize = 300

folder = "Data/A"

counter = 0

while True:

    success, img = cap.read()

    hands, img = detector.findHands(img)

    if hands:

        hand = hands[0]

        x, y, w, h = hand['bbox']

        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) \* 255

        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        imgCropShape = imgCrop.shape

        aspectRatio = h / w

        if aspectRatio > 1:

            k = imgSize / h

            wCal = math.ceil(k \* w)

            imgResize = cv2.resize(imgCrop, (wCal, imgSize))

            imgResizeShape = imgResize.shape

            wGap = math.ceil((imgSize - wCal) / 2)

            imgWhite[:, wGap:wCal + wGap] = imgResize

        else:

            k = imgSize / w

            hCal = math.ceil(k \* h)

            imgResize = cv2.resize(imgCrop, (imgSize, hCal))

            imgResizeShape = imgResize.shape

            hGap = math.ceil((imgSize - hCal) / 2)

            imgWhite[hGap:hCal + hGap, :] = imgResize

        cv2.imshow("ImageCrop", imgCrop)

        cv2.imshow("ImageWhite", imgWhite)

    cv2.imshow("Image", img)

    key = cv2.waitKey(1)

    if key == ord("s"):

        counter += 1

        cv2.imwrite(f'{folder}/Image\_{time.time()}.jpg',imgWhite)

        print(counter)

