Conversion of Sign Language to Text

Project Report

Submitted in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

Vijay Adhikari

University Roll NO.: 1900950100086

Vineet Kumar Sharma

University Roll NO.: 1900950100089

Navneet Kumar Singh

University Roll NO.: 1900950100047

Paras

University Roll NO.: 1900950100051

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MGM's College of Engineering & Technology, Noida

May, 2023

TABLE OF CONTENTS

Pages

CERTIFICATE

This is to certify that project report entitled “Conversion of Sign Language to Text” which is submitted by Vijay Adhikari, Vineet Kumar Sharma, Navneet Kumar Singh and Paras in the partial fulfillment of the requirement of the award of degree B. tech in department of Computer Science and Engineering of MGM’S College of Engineering and Technology which is affiliated by AKTU Lucknow is the record of the candidate own work carried out by them under my supervision. The matter embedded in this report is original and has been not submitted in the award of any other degree.

Date:

Name of Supervisor: Mrs. Deepti S. Deshmukh

Designation: HOD Supervisor Signature:

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person or material which to substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Signature:

Name of students: Vijay Adhikari, Navneet Kumar Singh, Paras and Vineet Kumar Sharma

Roll No.: 1900950100086, 1900950100047, 1900950100051, 1900950100089,

Date:

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Mrs. Deepti S. Deshmukh Department of Computer Science & Engineering, MGM's College of Engineering and Technology, Noida for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only her constant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Ms. Karamjeet Kaur Head, Department of Computer Science & Engineering, MGM's College of Engineering and Technology, Noida for her full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature: Date:

Name: Vijay Adhikari

Roll No.: 1900950100086

Signature:

Name: Vineet Kumar Sharma

Roll No.: 1900950100089

Signature:

Name: Navneet Kumar Singh

Roll No.:1900950100047

Signature:

Name: Paras

Roll No.: 1900950100051

ABSTRACT

Sign Language is a language in which we make use of hand movements and gestures to communicate with people who are mainly deaf and dumb. This paper proposes a system to recognize the hand gestures using a Deep Learning Algorithm, Convolution Neural Network (CNN) to process the image and predict the gestures. This paper shows the SLR of hand gestures of American Sign Language. The proposed system contains modules such as pre-processing and feature extraction, training and testing of model and sign to text conversion. Different CNN architecture and pre-processing techniques such as greyscale, thresholding, skin masking and Canny Edge Detection were designed and tested with our dataset to obtain better accuracy in recognition.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

%	Percentage
&	And
*	Multiplication
+	Plus
=	Equal's to
^	Power
[x]	Integer value of x.

LIST OF ABBREVATION

CNN	Convolutional Neural Network
HMM	Hidden Markov Models
ASL	American Sign Language
D&M	Dumb and Mute
RGB	Red Green Blue
ROI	Region of Interest
ReLu	Rectified Logical Unit
ADAGRAD	Adaptive Gradient Algorithm
RMSProp	Root Mean Square Propagation
SLR	Sign Language Recognition
GUI	Graphical User Interface
Fig	Figure

Chapter 1
INTRODUCTION, BACKGROUND OF THE PROBLEM, STATEMENT OF PROBLEM etc.

1) Literature Review

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.

1.1) Data Acquisition: The different approaches to acquire data about the hand gesture be done in the following ways:

i) Use of sensor: 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.

ii) Vision based approach: In vision based methods 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 7 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 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.

1.2) Data preprocessing and Feature extraction for vision based approach:

i) In the approach for hand detection combines threshold based color detection with background subtraction. We can use Ad boost face detector to differentiate between faces and hands as both involve similar skin-color.

ii) 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.

iii) For extracting necessary image which is to be trained we can use instrumented gloves as mentioned in. 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.

iv) 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.

1.3) Gesture Classification

i) In Hidden Markov Models (HMM) is used for the classification of the gestures. This model deals with dynamic aspects of gestures. Gestures are extracted from a sequence of video images by tracking the skin-color blobs corresponding to the hand into a body–face space 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 color pixels are gathered into blobs. Blobs are statistical objects based on the location (x, y) and the calorimetry (Y, U, V) of the skin color pixels in order to determine homogeneous areas.

ii) In Naïve Bayes Classifier is used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation. Thus, unlike many other recognition methods, this method is not dependent on skin color. 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.

iii) 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.

2) Problem Definition

People, who are not deaf, never try to learn the sign language for interacting with the deaf people. This leads to isolation of the deaf people. But if the computer can be programmed in such a way that it can translate sign language to text format, the difference between the normal people and the deaf community can be minimized. 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. Our team would like to develop a project that would enable deaf people to get more involved. The problem statement revolves around the idea of a camera based SLR system that would be in use for the deaf for converting sign language gestures to text. Our objective is to design a solution that is intuitive and simple. Communication for the majority of people is not difficult. It should be the same way for the deaf.

3)Brief introduction about the project

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. 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.

Sign language is a visual language and consist of three major components: 1) Finger spelling 2) Word level sign vocabulary 3) Non manual feature.

Fingerspelling	Word Level sign vocabulary	Non-manual features
Used to spell words	Used for the majority of communication	Facial expression and tongue, mouth and body position.

Table 1.1

4) Proposed Modules

4.1) Data set creation: 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 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. Finally, we apply our Gaussian blur filter to our image which helps us extracting various features of our image.

Figure 1.1 Grayscale image of alphabet "A"

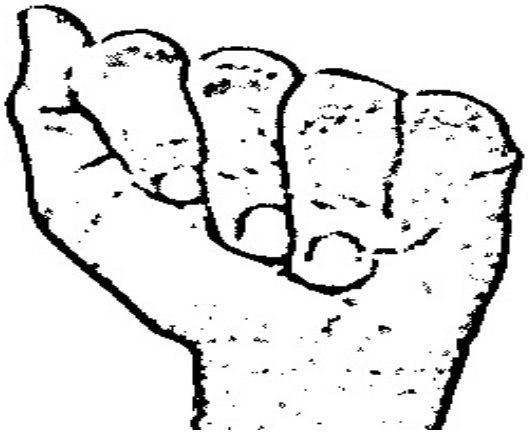


Fig 1.2 Image Obtained of Alphabet “A” After Applying Gaussian Filter

4.2) Training of data set: The entire data set that is prepared is trained to recognize different hand gesture. The hand gestures are trained using supervised learning that is we provide different hand gestures signs and train the program to recognize those sign. This is the most important part as few symbols or signs are similar to each other hence training those signs with a different set is important. This module defines how effective our project is. These are the steps used to training the CNN (Convolutional Neural Network):

Step 1) Upload Dataset: In this step in which the dataset collected is uploaded so that it can be trained using CNN it involves providing path where all the dataset is stored.

Step 2: The Input layer: This is the first layer in CNN which takes the data present in the dataset as input for the training purpose it is used to give input to the CNN layer.

Step 3: Convolutional layer: This is the layer where all the training part takes place it is a combination of weights and other data where all the data training takes place. It applies 14 5x5 filters (extracting 5x5-pixel sub-regions). Apply the number of filters to the feature map. After convolution, we need to use a relay activation function to add non-linearity to the network. The first convolutional layer has 18 filters with the kernel size of 7x7 with equal padding. The same padding has both the output tensor and input tensor have the same width and height. TensorFlow will add zeros in the rows and columns to ensure the same size.

Step 4: Pooling layer: The next step after the Convention is to down sampling the maximum facility. The objective is to reduce the mobility of the feature map to prevent

overfitting and improve the computation speed. Max pooling is a traditional technique, which splits feature maps into subfields and only holds maximum values. This will perform max pooling with a 2x2 filter and stride of 2 (which specifies that pooled regions do not overlap). The next step after the convolutional is pooling computation. The pooling computation will reduce the extension of the data. We can use the module max_pooling2d with a size of 3x3 and stride of 2. We use the previous layer as input.

Step 5: Convolutional layer and Pooling Layer: All neurons from the past layers are associated with the other next layers. The CNN has classified the label according to the features from convolutional layers and reduced with any pooling layer. There are ten neurons, one for each digit target class (0-9).

Step 6: Dense layer: The dense layer will connect **1764** neurons. We add a Relu activation function and can add a Relu activation function. We add a dropout regularization term with a rate of 0.3, meaning 30 percent of the weights will be 0. The dropout takes place only along the training phase.

Step 7: Logits Layer: Finally, we define the last layer with the prediction of model. The output shape is equal to the batch size 12, equal to the total number of images in the layer. We only want to return the dictionary prediction when the mode is set to prediction. We add these codes to display the predictions. The next step consists of computing the loss of the model. The final step is to optimizing the model, which is to find the best values of weight. For that, we use a gradient descent optimizer with a learning rate of 0.001. The objective is to reduce losses.

4.3) Creating trained models: The trained images are used to create a dataset of trained model which is capable of identifying images that are given as input by the user. The trained images are stored in the forms of models and more the number of trained models the easier it is to identify the signs and more efficient is the project. 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.

4.4) Gesture recognition: 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

4.4.1) 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 down sampled using max pooling of 2x2 i.e. we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.

3. 2nd Convolution Layer: Now, these 63 x 63 from the output of the first pooling layer 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.

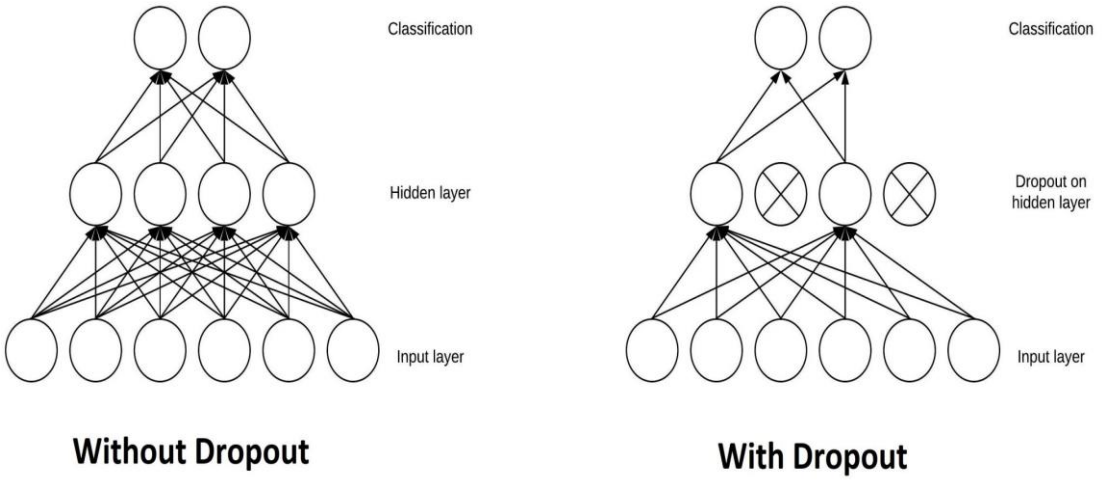
5. 1st Densely Connected Layer: Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer is an array of 28800 values. The output of these layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.

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).

4.4.2) 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.

4.4.3) 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].



1.3 Solving the problem of overfitting using dropout

4.4.4) 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).

5) Hardware and software requirement

5.1) HARDWARE

The most basic set of requirements defined by any software application or operating system is physical computer with specification:

Sr. No.	Parameter	Minimum Requirement	Justification
1)	CPU	Core i5 2.6GHz	For Training Model
2)	RAM	4 GB	To load large dataset
3)	GPU	GTX 1050 2GB	To train model with GPU parallelism

Table 1.2) Hardware Requirement

5.2) SOFTWARE

1) Operating System: Windows 8 and Above

2) Programming Language: Python 3.6

Chapter 2
SYSTEMS ANALYSIS AND SPECIFICATION

1) Functional Model:

i) This project used American sign language to convert into speech and text using the techniques of image segmentation and feature detection. The system goes through various phases such as data capturing, sensor, image segmentation, feature detection and extraction etc.

ii) This project is based on Creating a desktop application that captures a person signing gestures for American sign language (ASL), and translate it into corresponding text and speech in real time.

iii) This project builds a machine learning model which can classify the various hand gestures used in sign language. In this model, classification machine learning algorithms are trained using a set of image data.

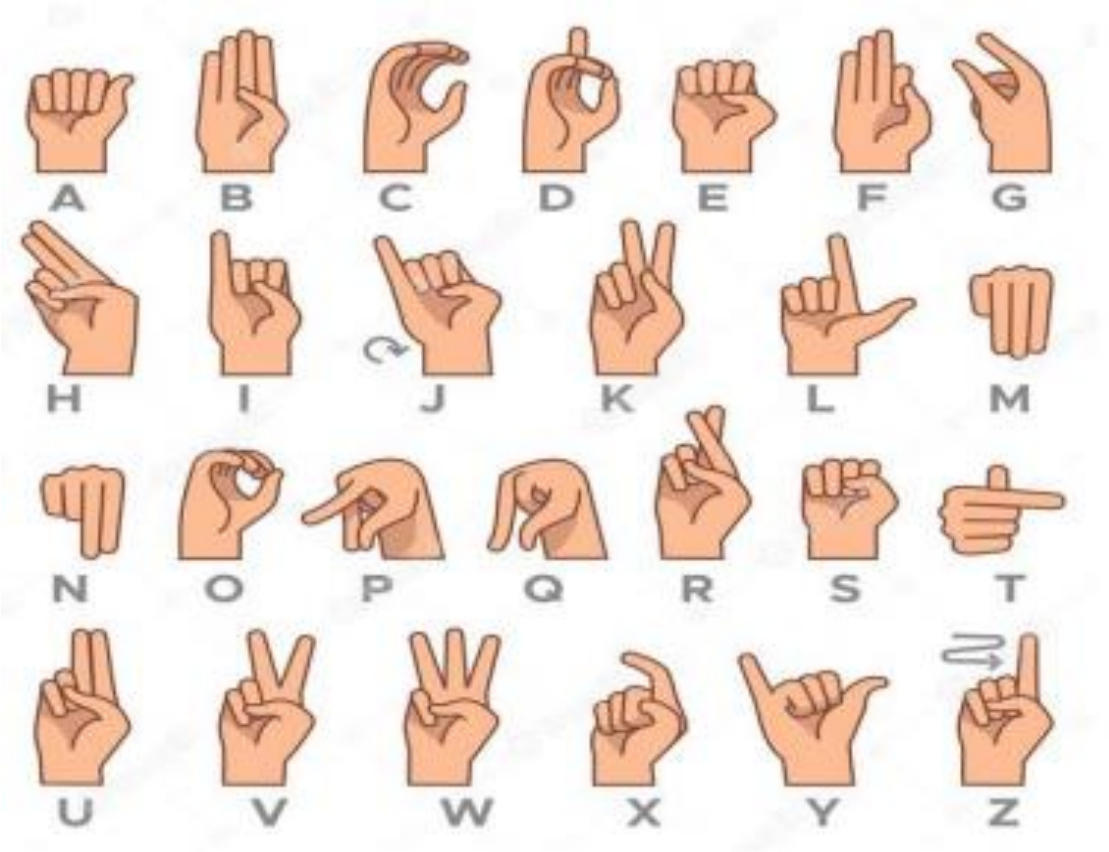


Fig 2.1 All Signs and Their Corresponding Letters

Our dataset is a large database of drawn representations of different gestures for American Sign Letters…. [4] The database contains 27,455 training images and 7172 testing images each of size 28x28. In our data, we have 784 columns of pixel1, pixel2…pixel784 which represents a single 28X28 image. These images contain matrices of pixel values and each pixel value is corresponding to English Alphabets A to Z. Our dataset does not contain any data for 9 = J and 25 = Z because in sign language these alphabets need motion. All these pixels' values can be presented directly to our model but this can result in challenges during modeling such as slower than expected training of the model. Instead, we believe it can be of great benefit in preparing pixel values before doing a modeling such as standardization.

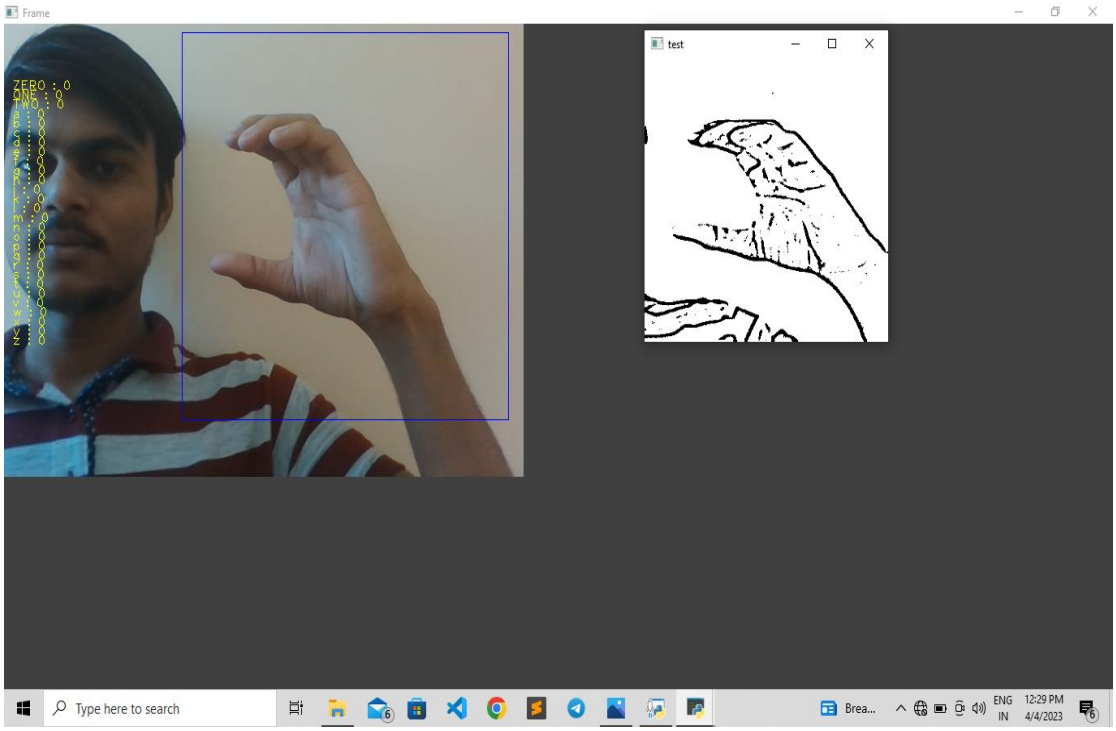


Fig 2.2 Dataset Image Collection

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance). Standardization scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one. Each pixel value is the feature of the dataset and since each pixel is important for an image so we need not do any feature selection/reduction. Our dataset does not have any missing, NaN, noisy or inconsistent value so we feel we need not do any kind of Data-cleaning.

2) Data Model

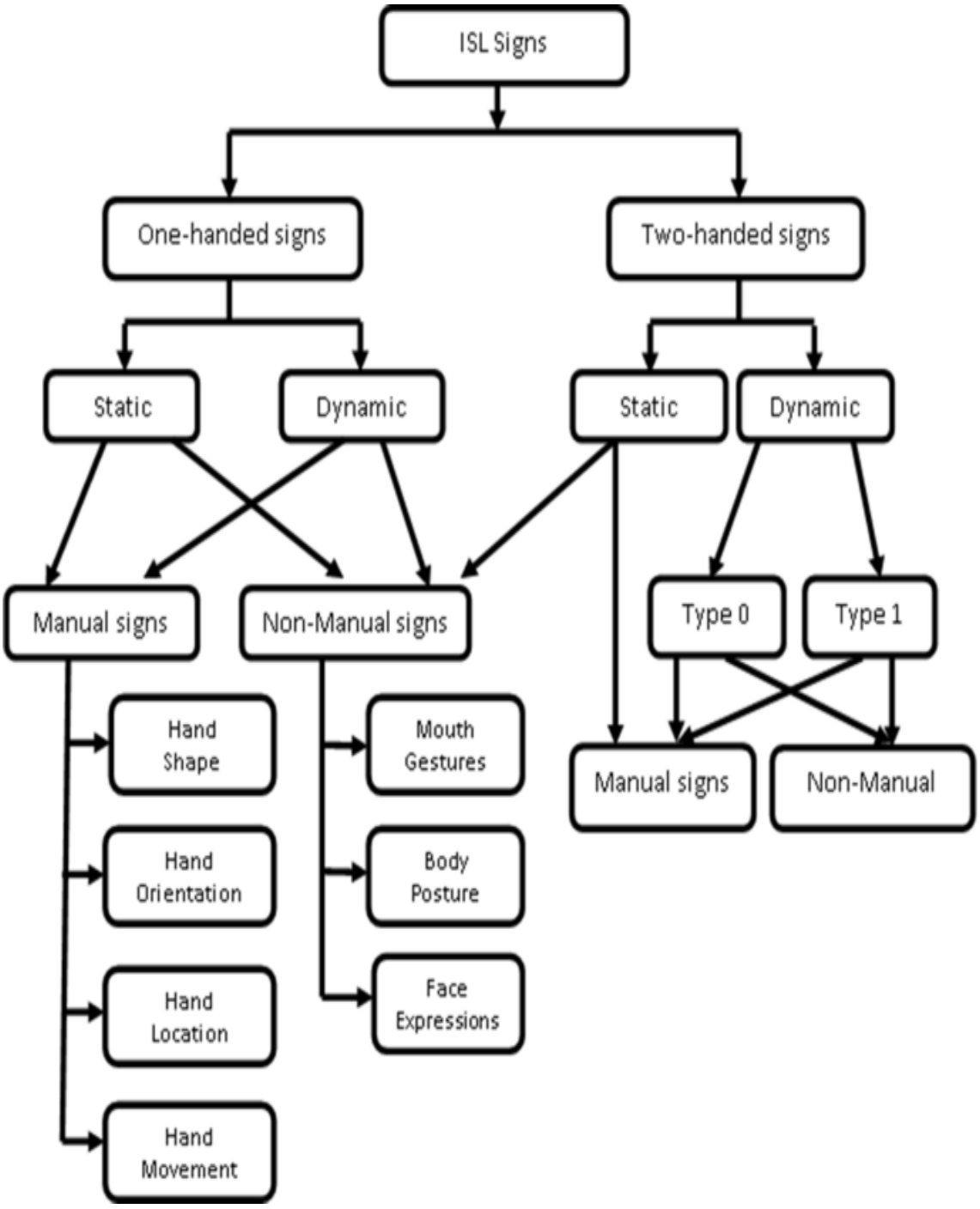


Fig 2.3 Data model chart

Sign languages (also known as signed languages) are languages that use the visual manual modality to convey meaning. Sign languages are expressed through manual articulation in combination with non-manual markers. Sign languages are full-fledged natural languages with their own grammar and lexicon.

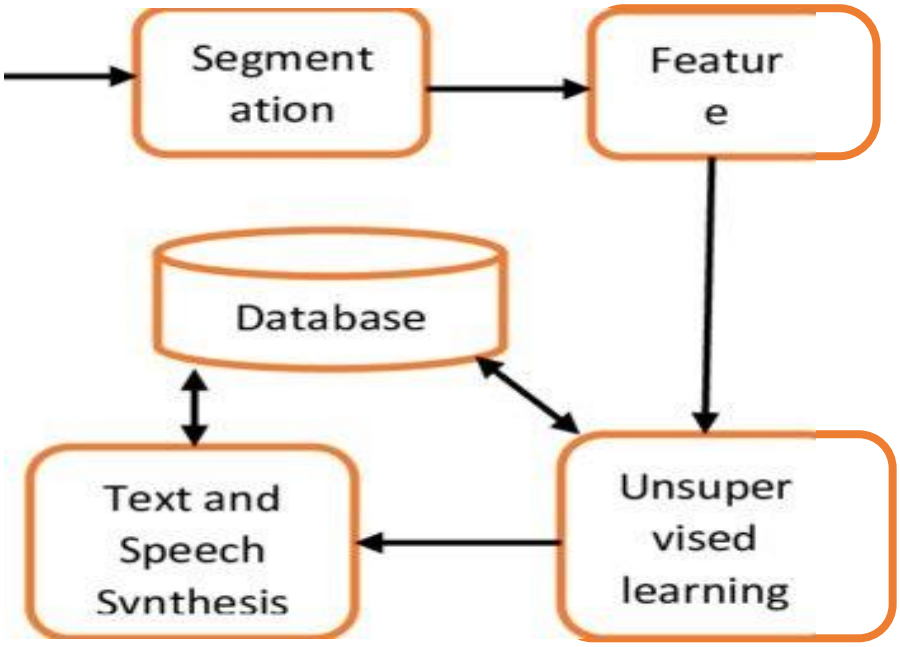


Fig 2.4 Data Model Diagram

The development of computer based SLR system, for enabling communication with hearing impaired people, is an important research area that faces different challenges in the pre-processing stage of image processing, particularly in boundary detection stage. In edge detection, the possibility of achieving high quality images sign. By helping to define and structure data in the context of relevant business processes, models support the development of effective information systems. They enable business and technical resources to collaboratively decide how data will be stored, accessed, shared, updated and leveraged across an organization.

3) A process flow model:

A process flow is a formal way of representing how a business operates. A process flow diagram is one method of representing a process model. The flow diagram/chart includes the following elements:

i) Activity: a step in the process

ii) Decision: where there are two or more options

iii) Flow: direction of the steps

iv) Terminator: beginning and end process

A process flow diagram is a way to visualize each subsequent task team need to complete to hit a goal. While they were originally designed for industrial engineering, process flowcharts have become an integral tool for business project management.

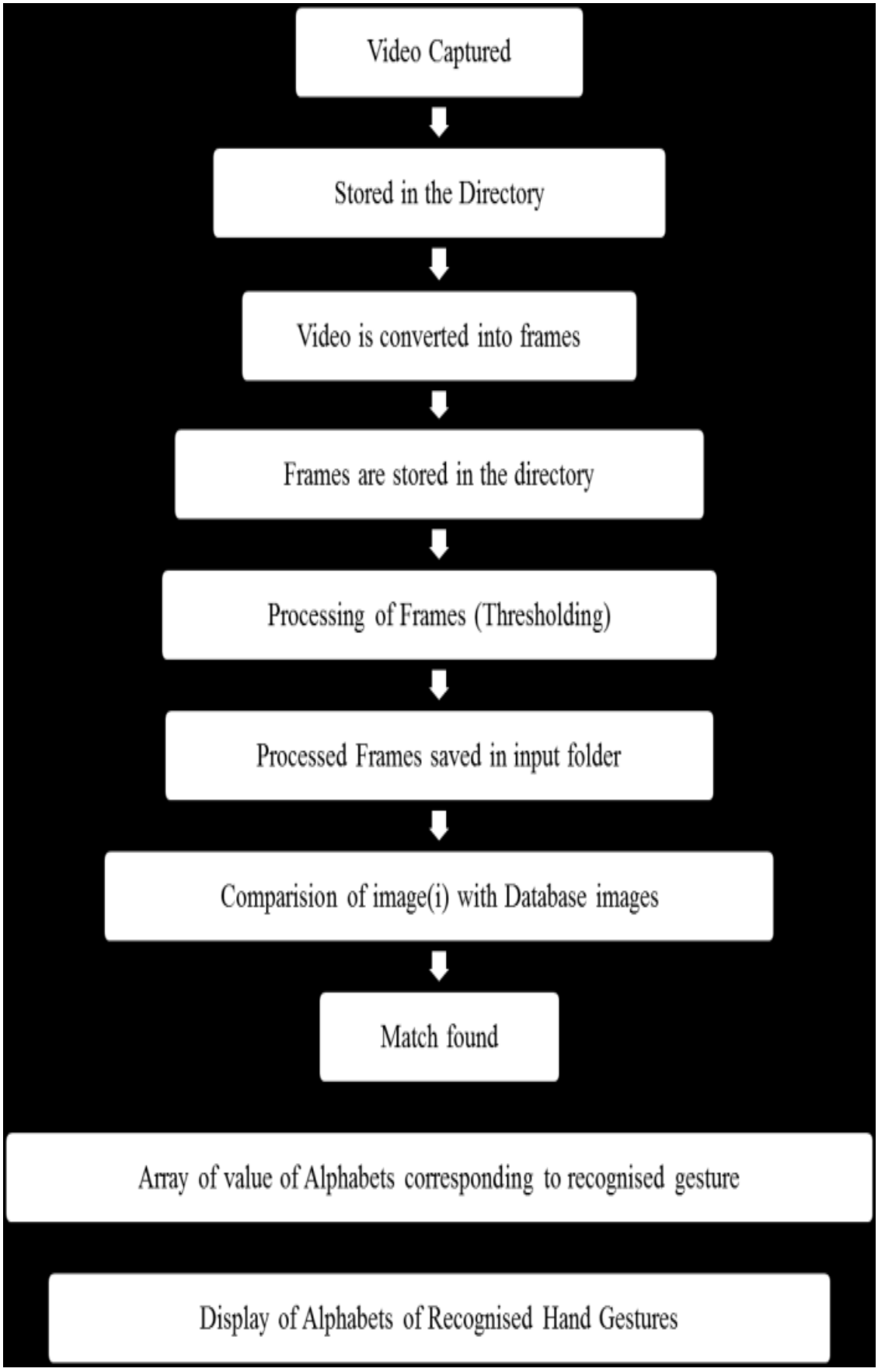


Fig 2.5 Process flow model

4) A behavioral model:

We conclude that SVM+HoG and Convolutional Neural Networks can be used as classification algorithms for SLR.

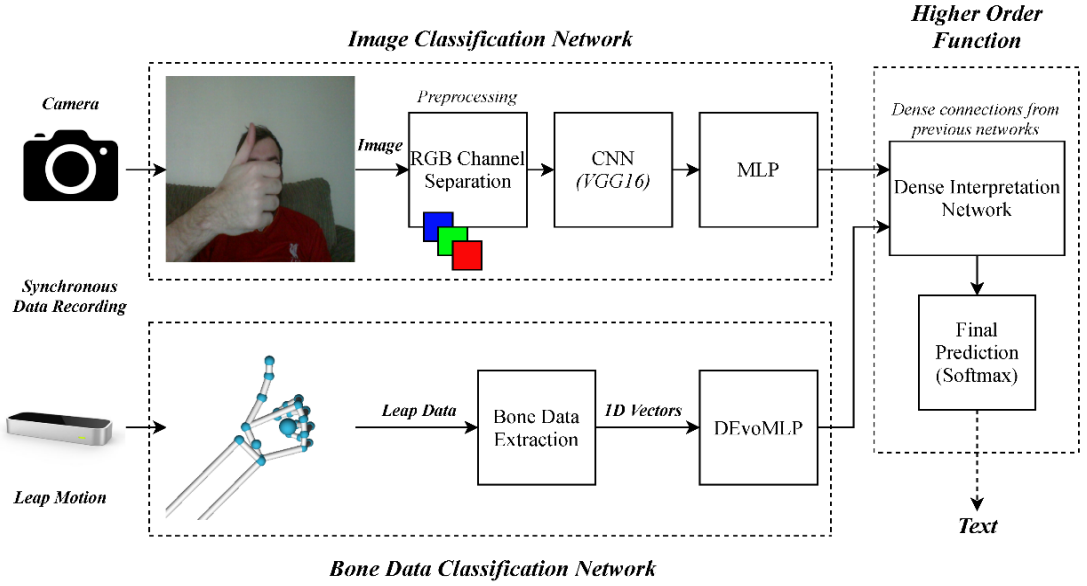


Fig 2.6 Behavioral Model

The definition of sign languages as systems arises out of traditional linguistic theory which has featured at its core the relationship between sound and language. Given such an emphasis, it is not surprising that the similarities between spoken and signed languages were, for many years, overlooked or refuted.

5) System Design:

It includes designing a system which is feasible and efficient in every possible way. The project designs cover various feasibility so that it can be called a good system design.

System design is the process of defining the architecture, interfaces, and data for a system that satisfies specific requirements. System design meets the needs of the business or organization through coherent and efficient systems. Once business or organization determines its requirements, one can begin to build them into a physical system design that addresses the needs of customers. The way a person design system will depend on whether he want to go for custom development, commercial solutions, or a combination of the two.

5.1) Technical feasibility:

The project is a working model and can run on different operating system. A 4gb ram system is good enough to run the project. The inbuilt camera of the laptop is used to

capture images and hence quite feasible. Two factors play a key role in the performance evaluation of continuous SLR, which include feature extraction from frame sequences of the input video and alignment between the features of each video segment and the corresponding sign label.

5.2) Operational feasibility:

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The project aims to achieve 95% accuracy so that it can be quite good operational feasibility. The project will recognize the hand gestures or signs with a good accuracy and convert them into text. The overall performance of the project will be enough to be operationally feasible.

5.3) Economic feasibility:

Devices like kinect are used for hand detection but our main aim is 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 economically feasible. A project is economically viable if the economic benefits of the project exceed its economic costs, when analyzed for society as a whole. The economic costs of the project are not the same as its financial costs—externalities and environmental impacts should be considered.

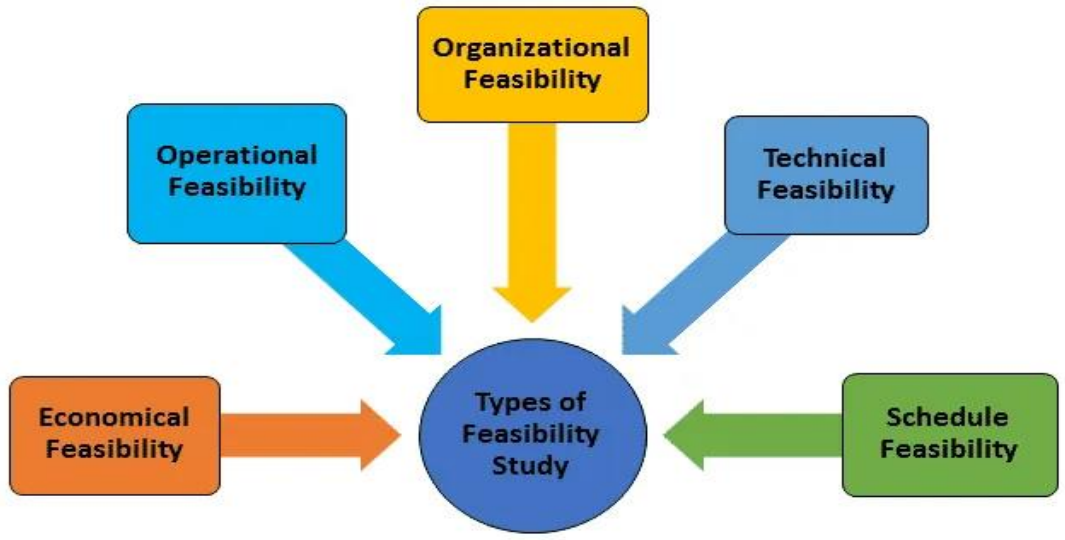


Fig 2.7 Feasibility Study

Chapter 3
Module Implementation and System Integration

1) OVERVIEW OF PROJECT MODULES

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.

1.1) 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 700 images of each of the symbol in ASL for training purposes and around 300 images per symbol for testing purpose. First we capture each frame shown by the webcam of our machine. In each frame we define a region of interest (ROI) which is denoted by a blue bounded square as shown in the image below. 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.

1.2) Gesture Classification; -

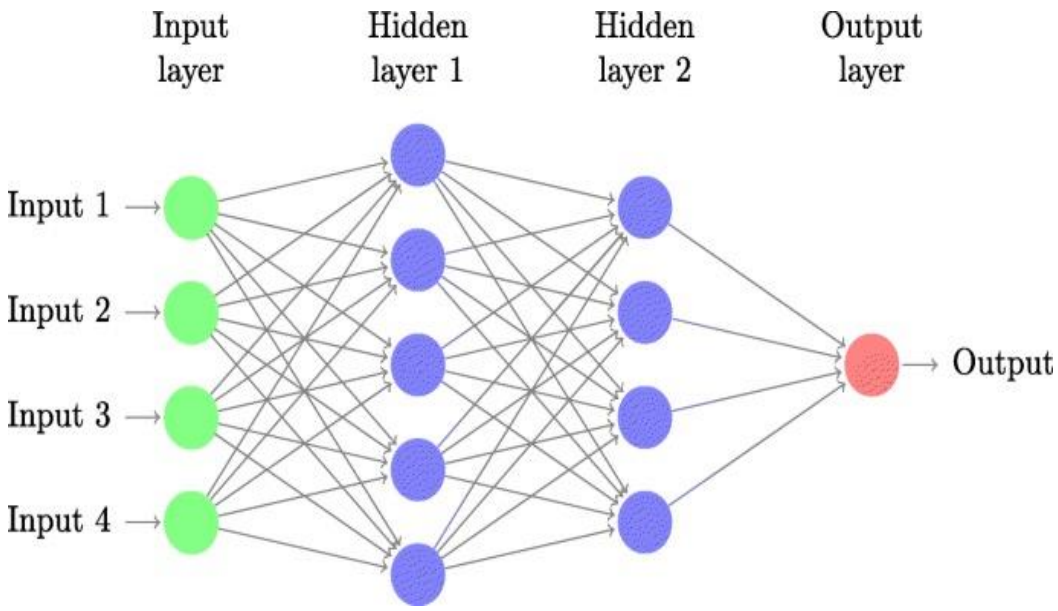


Fig 3.1 Artificial Neural Network

i) Layer 1; - CNN Model

(a) 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.

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

(c) 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.

(d) 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.

(e) 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.

(f) 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.

(g) 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).

1.3) Testing and Training: - 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.

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

1) Unsupervised Learning

2) Supervised Learning

3) Reinforcement Learning

1.3.1) Unsupervised Learning:

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. Two of the main methods used in unsupervised learning are principal component and cluster analysis. The only requirement to be called an unsupervised learning strategy is to learn a new feature space that captures the characteristics of the original space by maximizing some objective function or minimizing some loss function. Therefore, generating a covariance matrix is not unsupervised learning, but taking the eigenvectors of the covariance matrix is because the linear algebra Eigen decomposition operation maximizes the variance; this is known as principal component analysis.

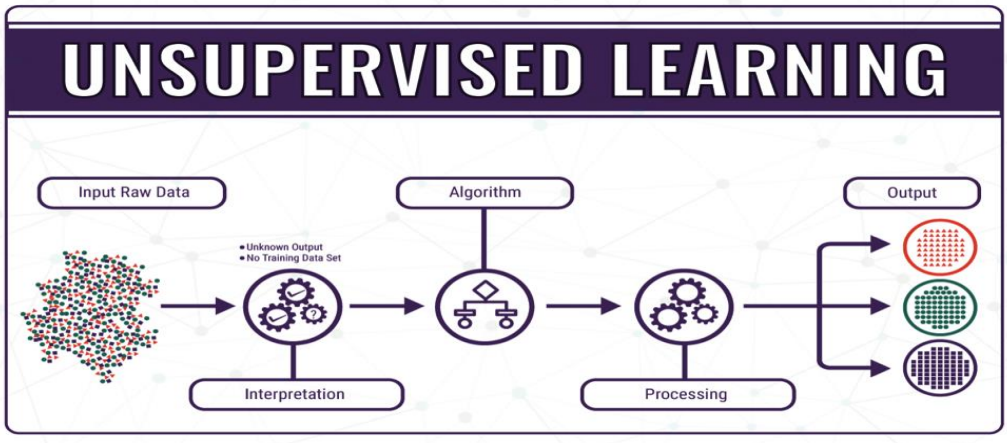


Fig 3.2) Unsupervised Learning

1.3.2) Supervised Learning:

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

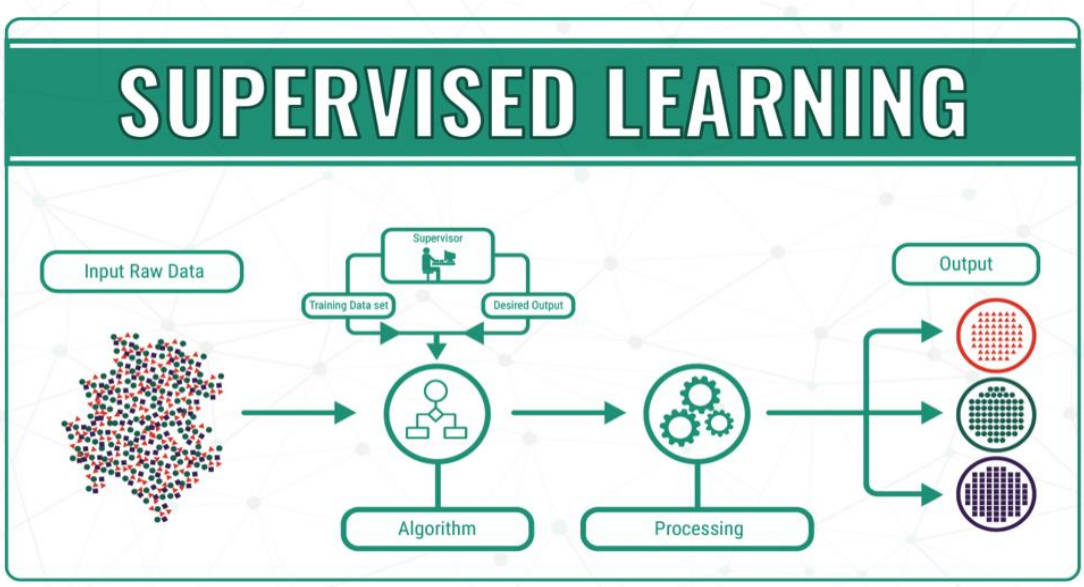


Fig 3.3 Supervised Learning

1.3.3) Reinforcement Learning:

Reinforcement learning is a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones. Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing 18sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

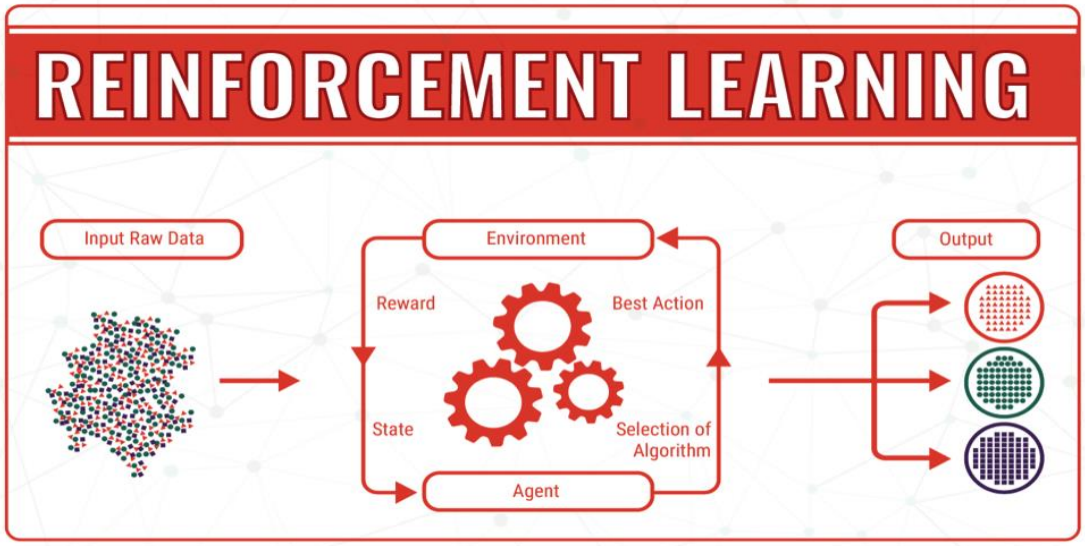


Fig 3.4 Reinforcement Learning

1.4) Model Deployment: - Going to deploy our model using inbuilt system camera.

2) TOOLS AND TECHNOLOGY USED

i) Python for implementation.

ii) Visual studios and Google collab for model creation.

iii) Adam optimizer for fast Gradient Decent calculations and adjusting the learning rate effectively.

iv) Image Data Generator for creating the wide verity of data from existing data with data augmentation.

v) Open-CV for sign language image collection.

vi) Keras for Deep Learning.

vii) Tkinter for frontend

3) ALGORITHM DETAILS

3.1) Convolutional neural networks: - Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

(a) Convolution Layer: - convolution layer we take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we 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

b) 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:

i) Max Pooling Layer: - 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 an activation matrix half of its original Size.

ii) Average Pooling: In average pooling we take average of all values in a window.

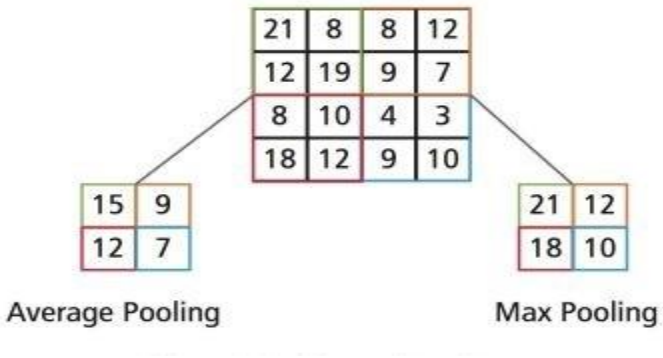


Fig 3.5 Polling Layer

(c) Final Output 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. 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.

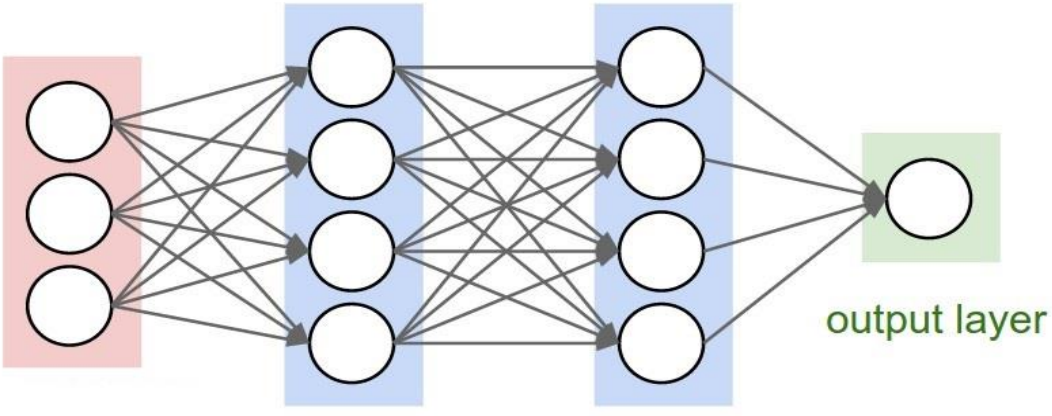


Fig 3.6 Output Layer

3.2) 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 algorithm namely adaptive gradient algorithm(ADAGRAD) and root mean square propagation(RMSProp).

4) Implementation

1 Whenever the count of a letter detected exceed output connected layers 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 60).

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 current gets appended to the sentence.

5) 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 29to append it to the current sentence. This helps in reducing mistakes committed in spellings and assists in predicting complex words.

4.1) TRAINING AND TESTING

i) 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.

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

iii) 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.

iv) 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.

v) 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.

vi) 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.

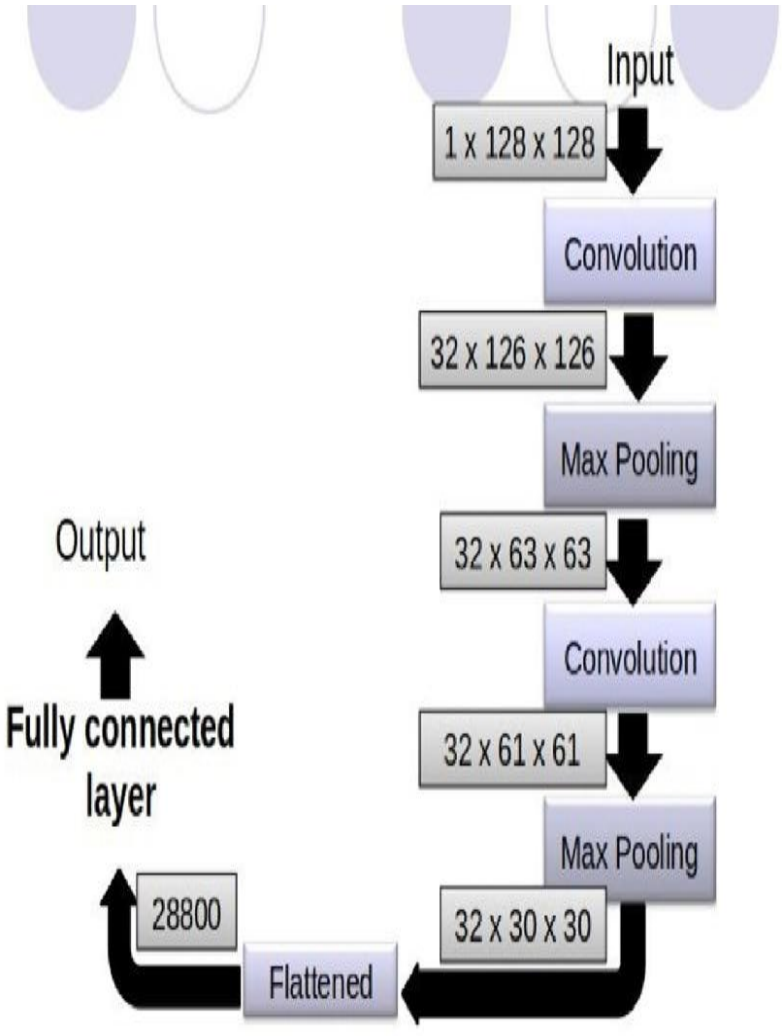


Fig 3.7 The CNN model used in the project

4.2) System Integrations: System integration is the process of connecting different sub- system(components) or modules into a single larger system or program that

function as one. With regards to software solution, system integration is typically defined as the process of linking together various modules. Systems integration consists of assuring that the pieces of a project come together at the right time and that it then functions as an integrated unit. However, to accomplish the integration process, all the various types of interfaces must be monitored and controlled, because integration, for the most part, is just another way of saying interface management. In addition, the number of interfaces can increase exponentially as the number of organizational unit's increase.

The dataset collection module is integrated with the system camera to access the images of the hand gestures shown by the user. The OpenCV library available in python is used to integrate the program with the camera.

In the next step the collected dataset is integrated with the preprocessing module where the dataset images are processed by performing thresholding and Gaussian blur. Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image_analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. Image thresholding is most effective in images with high levels of contrast.

Gaussian blur named after mathematician Carl Friedrich Gauss (rhymes with "grouse"), Gaussian ("gow-see-an") blur is the application of a mathematical function to an image in order to blur it. "It's like laying a translucent material like vellum on top of the image," says photographer Kenton Waltz. "It softens everything out." A type of low-pass filter, Gaussian blur smoothie's uneven pixel values in an image by cutting out the extreme outliers. Gaussian blur can mute that noise. If a person want to lay text over an image, a Gaussian blur can soften the image so the text stands out more clearly.

After preprocessing the dataset is trained using CNN. A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data. CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories. They can also be quite effective for classifying audio, time-series, and signal data.

The last step involves setting up an GUI which is used as the frontend to detect the signs using the images which are captured by the system camera and contain different hand gesture. A GUI is a visual friendly interface that allows us to interact with different devices.

Chapter 4
Testing and Evaluation

Test and Evaluation is the process by which a system or components are compared against requirements and specifications through testing. The results are evaluated to assess progress of design, performance, supportability, etc.

1) Testing

1.1) Pre Testing

In this phase the project was tested with the initial prepared models by training dataset. The program was getting confused among symbols or signs which have a lot of similarities. C&O, N&M and V&K were among those signs which had similar hand gestures. This was leading to serious problems in the classification of gesture to avoid it more training of the project over more dataset was important. The similar gestures were required to trained separately to avoid the confusion in detecting the hand gestures. The pre testing phase helped a lot to understand the drawbacks and lacking of the projects. It helped us to know the parts of the project we had to work on so that it could work effectively and deliver effective results. So that we can increase the efficiency of the project and it becomes operationally more feasible. We tested for every hand sign present in the ASL to check the efficiency of the project, though it failed to detect correct text for various gestures it was working effectively for other hand gestures.

1.2) Post Testing:

This was the testing which was carried out after the pre testing were the project failed to detect few of the hand gestures which were very similar such C&O,

N&M and V&K though it worked for other hand gestures. So we prepared more dataset for the similar hand gestures and trained them to get more trained models which would help to recognize the hand gesture easily and increase the project efficiency.

The separate dataset which was trained for similar gestures help the project to understand the similar gestures with more details and because of which it was possible for it to detect them. After the pre testing when the project was trained over similar gestures dataset it was able to detect the hand gestures effectively. We tested the project for every sign in the American sign language multiples times and the efficiency was almost 97% which helped us to know that the project was running effectively. In the post testing we checked every sign multiple times and it was able to differentiate among

the similar hand gestures and gave correct output for each gestures. The post training helped us to know about the operational efficiency of the project.

2) Results of testing:

1. An application interface that interprets American Sign Language to Text in real time to assist deaf and dumb for communicating with them effectively eliminating the requirement of a translating individual has been developed successfully.

2. The model devised achieved an accuracy of 96.5% which is greater than the many of the models that we researched. The sign had to be kept stable for at least 60 frames in front of the camera.

i) Other frame capturing values like 40 and 80 were also tried for prediction. For 40 frames, the predicted values were flickering much and for 80 frames the time taken to predict the correct alphabet was high. Hence frame value

ii) 60 proved to be the best optimized value for the sign detection.

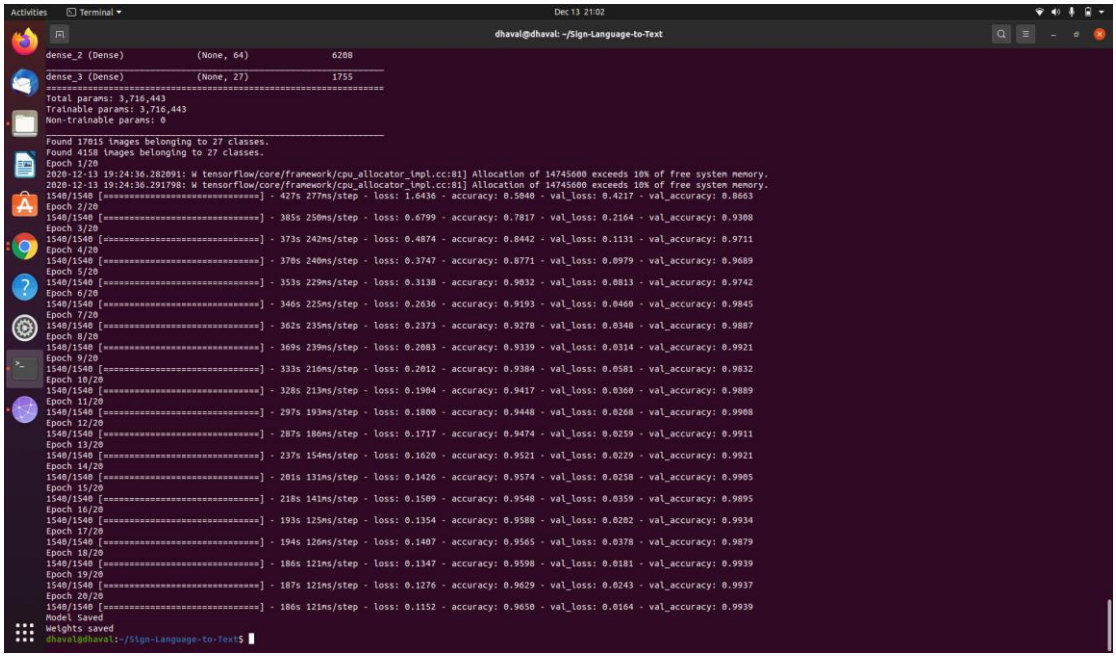


Figure 5.2 Accuracy achieved in our model is 96.5% as shown above

3. Our project is implemented using deep learning model that is implemented in python. The project requires only a webcam of a laptop and a computer system to be deployed. This is much cost effective and efficient than the hardware implemented system in [9] using flex sensors. The maintenance cost of our project is very less. We only need a functional webcam to detect the sign.

4. Our model provides suggestions based on the current word being translated from the word in the US dictionary:

For e.g.: If a speech or hearing impaired person wants to say “ELECTRON” to a normal person in sign language with the help of our project; all they have to do is start conversing in sign language for the initial alphabets (here ‘E’,’L’,’E’,’C’,’T’) and our project would provide 5 suggestions (like ‘ELECTOR’, ‘ELECTRON’, ‘ELECTRA’, ‘ELECTROCUTE’) accordingly that matches with the initials written so far.

5. The suggestions given by the hunspell library helps to detect the correct spelling mistakes that may be introduced due to wrong detections of an alphabet or wrong knowledge of spellings of a word.

6. We can make words by combining multiple characters and hence can also formulate sentences with the help of generated words. This is the highlighting feature of our project which is not available in any of the models that we researched.

3) APPLICATIONS

i) According to the National Deaf Association (NAD), 18 million people are estimated to be deaf in India. Hence a project like this could cater a huge percentage of such impaired community by providing a helping hand to communicate with the rest of the world using sign language. This leads to the elimination of the middle 58person who generally acts as a medium of translation. Due to the simplicity of the model, it can also be implemented in mobile application and is regarded as our future plan to do so.

ii) Deaf people do not have that many options for communicating with a hearing person, and all of the alternatives do have major flaws. Interpreters aren't usually available, and also could be expensive. Our project as mentioned before is quite economical and requires minimal amount of management. Hence is quite advantageous in terms of cost reduction.

iii) Pen and paper approach is just a bad idea: it's very uncomfortable messy, time-consuming for both deaf and hearing person. Our translator “ANUVADAK” thereby solves the problem by removing the need of any written approach for communication.

iv) This project idea can even be implemented for deaf and dumb community in various places like schools and colleges (for teaching purposes), airports (for security check or communicating during on boarding or in flight), community service agencies and courts (to adjudicate legal disputes between parties

Chapter 5
TASK ANALYSIS AND SCHEDULE OF ACTIVITIES

Task analysis is the process of learning about ordinary users by observing them in action to understand in detail how they perform their tasks and achieve their intended goals.

1) Task Decomposition

A task decomposition describes a particular task and then decomposes it to describe the task's features, such as devices and interface components used, duration, errors, and feedback.

1.1) Device and Interface Analysis

i) We used a normal laptop with 8gb RAM, Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz and on a 64-bit operating system, x64-based processor. The laptop has a 4-megapixel camera. The device also had a SSD of 256gb. The system was enough fast, it was easy to write code as it supported visual studios, sublime text and google collab. The device was fast because of good processor present and the we could easily manage different task due to availability of large RAM size. The device was small which was but the camera present on the system was good because of which we were able to get good pictures for the dataset creation. Moreover, due to clear images the project was able to identify the images easily and convert them into text.

RAM	Operating System	Processor	System Type	ROM
8GB	Windows and Linux	Intel- i7	64-bits	256

Table 5.1 Device Analysis

ii) Interface:

1) We used mainly used sublime text code the project. Sublime Text. Sublime Text is a shareware text and source code editor available for Windows, macOS, and Linux. It natively supports many programming languages and markup languages. Users can customize it with themes and expand its functionality with plugins, typically community-built and maintained under free-software licenses. To facilitate plugins, Sublime Text features a Python API. The editor utilizes minimal interface and contains features for programmers including configurable syntax highlighting, code folding,

search-and-replace supporting regular-expressions, terminal output window, and more. It is proprietary software, but a free evaluation version is available.

2) We used Python as the language to code our project. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3) The convolutional neural network was used to train the project over different hand gestures. A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data. CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories. They can also be quite effective for classifying audio, time-series, and signal data.

1.2) Duration Analysis

The project was completed in a duration of about ten months. We firstly began with project from the first month of our seventh semester. We had three presentations in our seventh semester. In the first we presentation we showed what our project was about and steps we necessary to completed the project. The overview of the project was given in this presentation. Till the next presentation we made the raw dataset for our project which was necessary to train the project for recognition of hand gestures. Till the third we finished with the preprocessing of the raw dataset and conversion of images to proper dataset was done. In the first moth of our eighth semester we had trained our dataset and the trained models were present with us. The last step was t prepare a GUI for the frontend so that we could show how the project runs and recognizes different hand gestures and converts them into text. The GUI was the most challenging thing do as few of the python libraries were not running on our device which was a big issue to handle it we tried different method and finally we were able to run GUI and project was completed in due duration. In the last presentation held in seventh semester we were able to show the complete project running with good efficiency.

1.3) Errors Analysis

There were many errors while preparing the project but the most difficult one was that the project was getting confused among similar hand gestures and signs so we had to prepare a separate dataset for the signs on which the results were wrong and we trained the project separately on those datasets and created more trained models. Thereafter the error was resolved.

1.4) Performance Analysis

Performance analysis is a measure of the success or failure of a project using various parameters. It helps in developing a positive culture of project management that yields excellent results. The projects perform with device which has a camera and can record continuous video as it is important to capture images of hand gestures. The performances of the project increases with increase in number of trained models as they are used to predict the gestures to text. The project can perform on low system as the requirement are very less. The project works in following steps:

i) The user collects raw images, processes them, and forms a good dataset.

ii) The user trains the dataset using CNN and creates well trained models which can be used to predict the gestures.

iii) The user creates a GUI which will act as the frontend to show and predict hand gestures.

iv) The users show hand gestures and the GUI shows correct text corresponding to that sign or hand gesture the auto correct feature is used to form sentence.

Performance feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The project achieves 97% (approx.) accuracy so that it can be quite good Performance feasibility. The project will recognize the hand gestures or signs with a good accuracy and convert them into text. The overall performance of the project will be enough to be said good Performance feasible.

Performance analysis involves planning, monitoring and reviewing. They are beneficial in the following ways:

i) Planning: the act or process of making or carrying out plans. specifically: the establishment of goals, policies, and procedures for a social or economic unit.

ii) Monitoring: It is important to keep monitoring the project as it helps to know the short coming and the drawbacks of our project. Proper monitoring helps in to develop an error free and efficient project.

iii) Reviewing: It is important to examine or consider something again in order to decide if changes are necessary. If changes are necessary, we can work on them or we can move ahead with the further work.

1.5) Content analysis

Using content analysis, researchers can quantify and analyze the presence, meanings, and relationships of such certain words, themes, or concepts. The content or the code written use be effective. It should not contain unnecessary line of codes and should have proper comments so that people can easily understand the working of the programs written. The Content should be less but good adding unnecessary line should be avoided and should cover edge cases and our code has all these qualities.

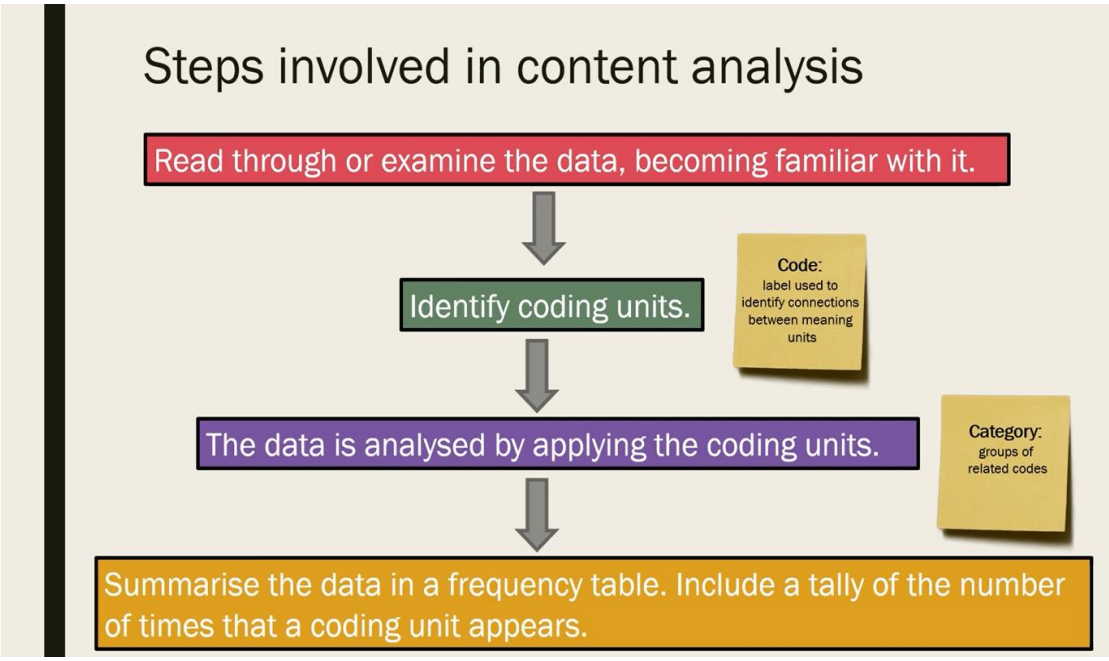


Fig 5.1 Content Analysis

2) Project Scheduling

It's a timetable that outlines start and end dates and milestones that must be met for the project to be completed on time. The project schedule is often used in conjunction with a work breakdown structure (WBS) to distribute work among team members.

Our project scheduling is shown in the following figure where task represent the different modules of project (which are also mentioned below) and period of time in which our project was created.

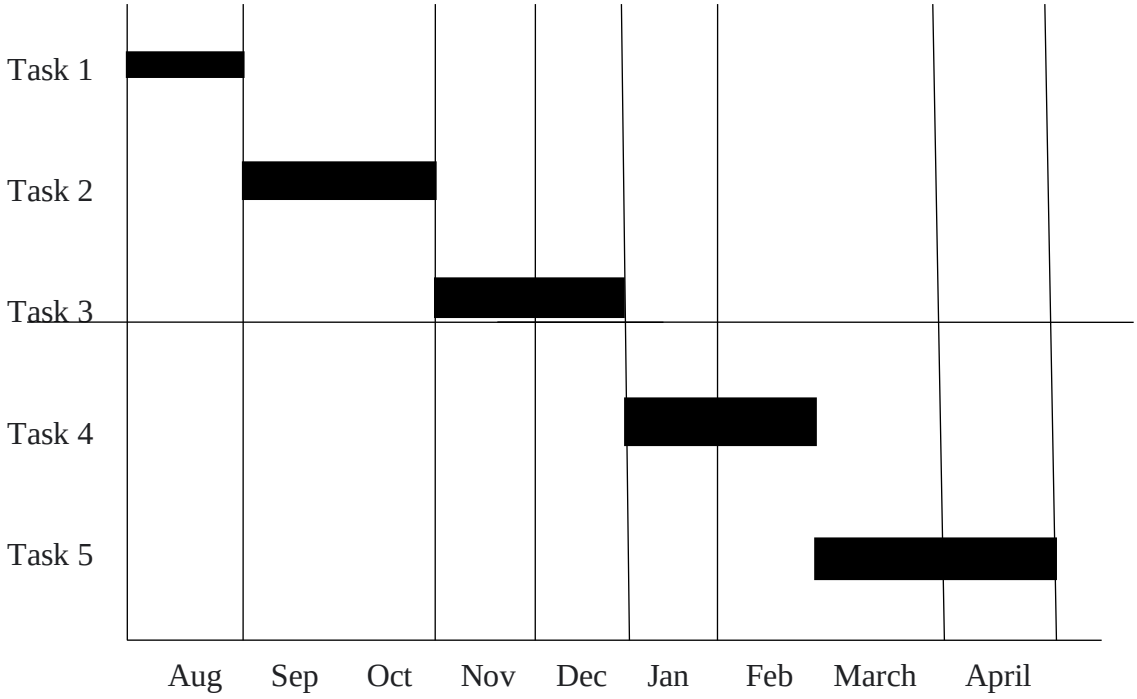


Fig 5.2 Project Scheduling

The different tasks are:

i) Task 1: Planning and gathering information about the project which was to be created. It involved detail research and study about the project and the steps that would help us to make the project.

ii) Task 2: Creating a dataset for the project which was necessary to train the project to be trained.

iii) Task 3: Processing the image and removing unnecessary information by applying Gaussian blur and thresholding

iv) Task 4: Training the prepared. dataset and preparing trained models so that detect hand gesture.

v) Task 5: Creating a GUI as frontend.

3) Task specification:

3.1) Task 1:

i) Goal: To gather knowledge about the project, and detail research about the project. It involved learning about all the steps involved in making the project, the difficulties and the scope of the project.

ii) Efforts and duration: Our every team member was involved in gathering information about the project, with the help of different sites we were able gather enough information about the project in about half a month.

3.2) Task 2:

i) Goal: Creating a raw image dataset for the project.

ii) Input: The input for this steps where the images captured by the laptops camera. These images involved the hand signs involved in the making of the project.

iii) Output: The out of this step was a raw image dataset which was to be preprocessed to remove the unnecessary or irrelevant details.

iv) Efforts and duration: Inorder to complete this task we created a file "collect data" which was used to collect raw images of different signs that where necessary to be trained. This task took more than a month as we had to study about different python modules used to perform this task.

3.3) Task 3:

i) Goal: Processing the raw images collected to remove unnecessary information by applying Gaussian Blur and thresholding.

ii) Input: The input to this step was the collected raw images in the previous step of collecting raw image data.

iii) Output: The output of this step was a proper dataset for the project which could be trained to generate trained models.

iv) Efforts and duration: This step involved creating a processing module in which we applied thresholding and Gaussian blur on the images so that the image could contain only relevant information. This task was completed in nearly two-month time.

3.4) Task 4:

i) Goal: The aim of this task was to train the prepared dataset so that proper trained models could be generated.

ii) Input: The input of the step was the dataset prepared in the previous step after processing the images.

iii) Output: The output of this step was well trained models which were generated after training the dataset.

iv) Efforts and duration: This step involved training the model using unsupervised learning for this a "training" file was created. This step took two months to complete.

3.5) Task 4:

Goal: Create a GUI which could show us the desired output when we show a sign.

Input and output: Input is image shown by user and output is corresponding text.

CHAPTER 6
PROJECT MANAGEMENT

6.1. Major risks and problem faced

Risk Identification forms an important part in Software Development Life Cycle. This gives a view of all the possible risks to the system and whether they can be controlled or not. We may define risks by presenting a questionnaire and answering all the questions relating to it or we also have a table giving risk details and probability of occurrence. The major risk and problem faced are:

i) The model works well only in good lighting conditions.

ii) Plain background is needed for the model to detect with accuracy.

iii) We couldn't find a dataset with raw images of all the asl characters so we made our own dataset.

iv) Second issue was to select a filter for feature extraction. We tried various filter including binary threshold, canny edge detection, Gaussian blur etc., of which Gaussian blur filter was giving better results.

v) 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.

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Underestimation of time required to train the model	Low	High	Medium	High
2	No datasets available	High	Medium	High	High

Table 6.1 Risk Table

Probability	Value	Description
High	Probability of occurrence is	> 75%
Medium	Probability of occurrence is	26 – 75%
Low	Probability of occurrence is	< 25%

Table 6.2 Risk Probability definitions

6.2. Principal learning outcomes

Conclusion

In this report, a functional real time vision based American sign language recognition for D&M people have been developed for asl alphabets.

i) We achieved an accuracy of **95.7%** on our dataset.

iii) Prediction has been improved after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other.

Future Scope

i) We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms.

ii) We are also thinking of improving the preprocessing to predict gestures in low light conditions with a higher accuracy.

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 Markova 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 21 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.

		Predicted Values																									
		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
Correct Values	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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

Fig 6.1 Algorithm 1

		Predicted Values																									
		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
Correct Values	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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
	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

Fig 6.2 Algo1+Algo2

```
Epoch 1/5
1285/1285 [==============================] - 190s 148ms/step - loss: 0.0691 - accuracy: 0.9811 - val_loss: 0.0046 - val_accuracy: 0.9986
Epoch 2/5
1285/1285 [==============================] - 167s 130ms/step - loss: 0.0539 - accuracy: 0.9853 - val_loss: 0.0069 - val_accuracy: 0.9977
Epoch 3/5
1285/1285 [==============================] - 164s 128ms/step - loss: 0.0433 - accuracy: 0.9889 - val_loss: 0.0207 - val_accuracy: 0.9946
Epoch 4/5
1285/1285 [==============================] - 164s 127ms/step - loss: 0.0503 - accuracy: 0.9863 - val_loss: 0.0059 - val_accuracy: 0.9988
Epoch 5/5
1285/1285 [==============================] - 168s 131ms/step - loss: 0.0370 - accuracy: 0.9900 - val_loss: 0.0018 - val_accuracy: 0.9993

<tensorflow.python.keras.callbacks.History at 0x21004d31730>
```

Fig 6.3: Training process

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4194432
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 96)                12384
_________________________________________________________________
dropout_1 (Dropout)          (None, 96)                0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                6208
_________________________________________________________________
dense_4 (Dense)              (None, 27)                1755
=================================================================
Total params: 4,240,859
Trainable params: 4,240,859
Non-trainable params: 0
```

Fig 6.4: Training output

APPENDIX

Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Fig 7.1 Python

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the 64quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Convolution Neural network: CNNs use a variation of multilayer perceptron’s 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.

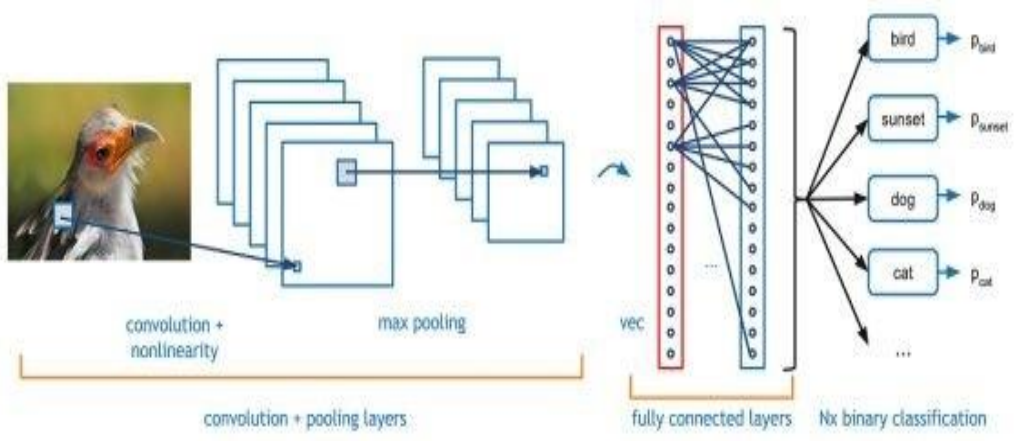


Fig 7.2 CNN

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 OpenCV, 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.

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.

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.

References

[1] M.M.Gharasuie, H.Seyedarabi, “Real-time Dynamic Hand Gesture Recognition using Hidden Markov Models”, 8th Iranian Conference on Machine Vision and Image Processing (MVIP), 2013.

[2] Pradumn Kumar, Upasana Dugal, “Tensorflow Based Image Classification using Advanced Convolutional Neural Network”, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-6, March 2020, March, 2020.

[3] Xin Jia, “Image Recognition Method Based on Deep Learning”, CCDC,

DOI: 978-1-5090-4657-7/17, 2017.

[4] Jiudong Yang, Jianping Li, “Application Of Deep Convolution Neural

Network”, IEEE, DOI: 978-1-5386-1010-7/17, 2017.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever,

Ruslan Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 15 (2014) 1929-1958, 2014.

[6] Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, Dr. Dayananda P, “Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network”, International Journal of Engineering Research & Technology(IJERT), ISSN: 2278-0181, NCAIT - 2020 Conference Proceedings.

[7] Nobuhiko MUKAI, Naoto HARADA, Youngha CHANG, “Japanese Fingerspelling Recognition based on Classification Tree and Machine Learning”, NICOGRAPH International, 2017. [8] Qi Wang, Zequn Qin, Feiping Nie, Yuan Yuan, “Convolutional 2D LDA for Nonlinear Dimensionality Reduction”, Proceedings of the Twenty

Sixth International Joint Conference on Artificial Intelligence (IJCAI-17).

[9] Aarthi M, Vijayalakshmi P, “Sign Language To Speech Conversion”, Fifth International Conference On Recent Trends In Information Technology, 2016.

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

[11] https://opencv.org/

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

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