SIGN LANGUAGE TO TEXT CONVERTER

A MINI PROJECT REPORT
Submitted by

Rahul C Karthik (RET19CS173)

Thejus P.S (RET19CS208)

Tejas Ananthajith (RET19CS206)

Sleety Kottuviruthil George (RET19CS198)

Under the guidance of

Ms. Maria Vijoy

to the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology(Autonomous)

Rajagiri Valley, Kakkanad, Kochi, 682039
July 2022

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)

RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

*Certified that Mini Project work entitled "**Sign Language to Text Converter"** is a bonafide work done by **Rahul C Karthik (RET19CS173), Thejus P.S (RET19CS208), Tejas Ananthajith (RET19CS206), Sleety Kottuviruthil George (RET19CS198)** of Sixth semester in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of the APJ Abdul Kalam Technological University during the academic year 2021-2022.*

Ms. Maria Vijoy	**Mr. Paul Augustine**	**Dr. Dhanya P M**
Project Guide	**Project Coordinator**	**Head of Department**

Place : Kakkanad

Date :

ACKNOWLEDGEMENT

We are extremely honored to thank Prof. (Dr). P.S. Sreejith, Principal, RSET who has always made sure that our ladder to success was leaning against the right wall. We thank him for his constant help and support. We thank Dr. Dhanya P M, HoD, Department of Computer Science & Engineering, RSET whose help and guidance has been a major factor in completing our journey.

We express our gratitude to our Project Coordinator and Class-In-Charge, Mr. Paul Augustine, Asst. Professor, Dept. of Computer Science Engineering for his support and timely help whenever it was required. He has been a great mentor without whom this would have been incomplete.

We extend our sincere and heartfelt thanks to our guide Ms. Maria Vijoy, Asst. Professor, Dept. of Computer Science & Engineering, RSET for taking the time and effort to review our work and providing valuable advice and feedback from time to time. Last but not the least we would like to express our heartfelt and sincere gratitude to our family and friends for their constant support throughout this entire journey.

Rahul C Karthik, Thejus P.S, Tejas Ananthajith, Sleety Kottuviruthil George

ABSTRACT

Sign language is the only medium of communication between the people with disabilities in hearing and speaking. Even if they need to present their thoughts or ideas to any individual, they will do that using actions. It may not be understood accurately and efficiently by the individual, which may result in misunderstanding leading to greater problems. In this project, we have aimed at converting their actions, the sign language, to text. The actions will be understood by the system, depending upon the knowledge of the actions, and then translated accordingly to the required text. The text can be easily read and thus favoring communication.

The software also aims at saving time by encoding some signs to a text that would normally be large in size. We just need to use the desired sign and it will be converted automatically to the large text. This can also be used for security purposes, for developing one's own Sign language.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

HSV	Hue Saturation Value
CNN	Convolutional Neural Network
ASL	American Sign Language
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
MNIST	Modified National Institute of Standards and Technology
IoT	Internet of Things
CV	Computer Vision
Relu	Rectified Linear Unit
Convnet	Convolutional Neural Network
DFD	Data Flow Diagram

Chapter 1

INTRODUCTION

1.1 GENERAL BACKGROUND

Today, there are almost 2 million people classified as Deaf and Dumb. They have great difficulty in communicating with each other and with other individuals as the only means of communication is sign language. They need to learn this sign language. It is extremely difficult for a person who is unaware of this sign language to understand and decode their actions.

It is impossible to identify anything without its prior knowledge. Even for computers, they need to have information in their memory to identify and provide data related to any object. Now one particular object may differ from another similar type of object in shape, size, orientation or even visual effects may differ. But all the different forms of 1 type of object must be classified in the same category.

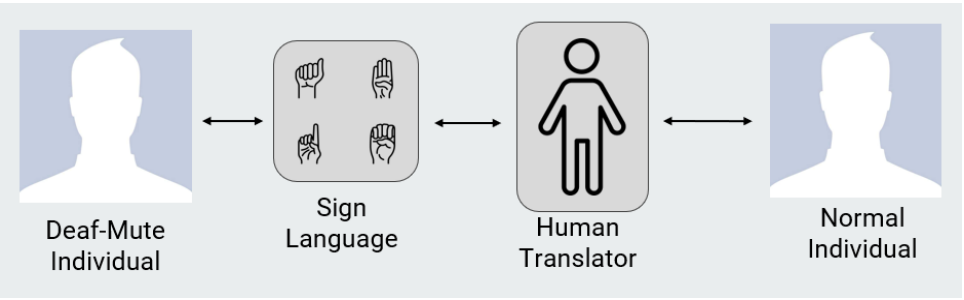

fig 1.1.1: Present Scenario

Making this our aim in abstracting, modifying, analyzing and identifying the various signals used by the Deaf and Mute to communicate, we have developed this model.

The major technologies used are IMAGE PROCESSING and CNN (Convolutional Neural Network).

A number of images of some gestures are taken and processed to make the dataset. The CNN model is then trained using Tensorflow on these captured and modified images. The signs to be translated are then fed to the software which matches it with the existing images and classifies it.

1.2 OBJECTIVE

The objective of this project are:

1. Deaf/dumb people can communicate with normal people using hand gestures; without the need of a human translator.
2. The program is portable, easy to use, understand and makes communication between a normal person and a deaf/ dumb person more convenient.

1.3 MOTIVATION

It is very difficult for the deaf people to communicate with the hearing person and there are not many options available to help them. And all of the alternatives have some major flaws. Interpreters are not usually available and are expensive. Pen and paper is also not a good idea, it is uncomfortable, messy and even time consuming, both for the deaf and the hearing person. With the evolution of IoT, everything around is getting automated. The demand for Machine Learning and its applications is very high. The accuracy and efficiency of any algorithm and the model developed must be very high to make it useful.The knowledge of Machine Learning thus becomes very important.

In the era of Machine Learning where everything is getting automated, the need for an Interpreter to translate Sign language to text is just a waste of resources. The classification of objects, object detection and image processing plays a very vital role.

Thus, the main aim is to bridge the gap between the normal and the deaf-andmute individuals by providing an automatic translation system.

1.4 Summary of Report

The primary goal of the project is to develop a sign language to text translator using concepts of computer vision, deep learning and object detection that translates an 2D image (gesture) to its corresponding display text. This is to ensure a more comfortable and convenient communication between a normal person and a person with disabilities. We have also developed a Graphical User Interface for users to perform operations such as creating new datasets and launching the translator.

Chapter 2

LITERATURE SURVEY

2.1 Zhi-Jie Liang, Sheng-Bin Liao, Bing-Zhang Hu: 3D Convolutional Neural Networks for Dynamic Sign Language Recognition (2018) [5]

2.1.1 INTRODUCTION

The model is built to remove the communication gap between deaf/dumb and others. It has to tackle the issue of detecting and tracking the moving limbs under different backgrounds and lighting. Second, it is a nontrivial challenge to integrate the temporal and spatial features together.

2.1.2 DATA COLLECTION AND PREPROCESSING

This involves data collection by Kinect, Sign Language Temporal Segmentation and pre-processing.

2.1.3 MODELS

2.1.3.1 3D Convolution

3D operators are provided in the convolution and pooling stages so that discriminative features from both spatial and temporal dimensions are captured. The 3D convolution is achieved by convolving a 3D kernel to the cube of temporal formed by stacking multiple contiguous frames together.

2.1.3.2 3D-CNN Architecture

The proposed model is a data driven learning system that consists of two sub-networks: an Infrared-classifier network and a Contour-classifier network.

2.1.3.3 Initialization

The weights of the fully connected hidden layers and the softmax layer were initialized with random samples from a normal distribution.

2.1.3.4 Optimization

The process of training CNN architecture involves the optimization of the network parameters to minimize the cost function for the dataset and fine-tune by back-propagation algorithm.

2.1.3.5 Regularization

In order to reduce overfitting, Regularization was applied which was regarded as a key component to successful generalization of the network.

2.1.3.6 Multimodal Fusion

Fusion model combines the class-membership probabilities estimated from the separate single-modality networks by linear combination to compute the final probabilities for the sign language classifier.

2.1.3.7 GPU Acceleration

It is made possible by using a GPU-based calculation with the help of CUDA-Convnet for parallel processing to achieve real-time efficiency.

2.2 Sankit Ojha, Ayush Pandey, Shubham Maurya: Sign Language to Text and Speech: Translation in Real Time Using Convolutional Neural Network (2020) [1]

Real time signal language to textual content and speech translation, specifically: Reading human signal gestures, training the system learning model for image to textual content translation, forming words, forming sentences, forming the entire content, obtaining audio output.

2.2.1 IMAGE ACQUISITION

The gestures are taken through the web camera using OpenCV. The frames are extracted and processed as grayscale with dimension 50*50. This dimension is consistent throughout the project as the entire dataset is the same.

2.2.2 HAND REGION SEGMENTATION & HAND DETECTION AND TRACKING

The captured images are scanned for hand gestures. This is a part of preprocessing before the image is fed to the model to obtain the prediction. The segments containing gestures are made more pronounced. This increases the chances of prediction by many folds.

2.2.3 HAND POSTURE RECOGNITION

The preprocessed images are fed to the keras CNN model. The model that has already been trained generates the predicted label. All the gesture labels are assigned with a probability. The label with the highest probability is treated to be the predicted label.

2.2.4 DISPLAY AS TEXT AND SPEECH

The model accumulates the recognized gesture to words. The recognized words are converted into the corresponding speech using the pyttsx3 library. The text to speech result is a simple work around but is an invaluable feature as it gives a feel of an actual verbal conversation.

2.2.4.1 Convolutional Neural Network for Detection

CNNs are a class of neural networks that are highly useful in solving computer vision problems. They found inspiration from the actual perception of vision that takes place in the visual cortex of our brain.. The CNN is equipped with layers like convolution layer, max pooling layer, flatten layer, dense layer, dropout layer and a fully connected neural network layer. These layers together make a very powerful

tool that can identify features in an image. The starting layers detect low level features that gradually begin to detect more complex higher-level features.

2.2.4.2 Recognition of Alphabets and number

To discover bounding packing containers of various objects, we used the Gaussian historical past subtraction which used a technique to version each history pixel with the resource of a mixture of K Gaussian set distributions (k varies from 3 to 5). The possibly historical past colorations are those that stay longer are greater the static. On those fluctuating pixels, we design a square bounding field. After Obtaining all the gesture and heritage, a Convolutional NN model has been designed using those photos to separate the gesture symptoms and signs from the historical beyond. These function maps explain that the CNN can understand the common unexposed structures of some of the gesture indicators within the training set.

2.2.5 SUMMARY

This research paper sheds light on how CNN is used for conversion of hand gestures in real time.This is done by using the multiple layers of the CNN which start by detecting low level features and then climbs to higher level features and once the model is trained to identify these features it will display the corresponding output to the sign.

2.3 Mehreen Hurroo, Mohammad Elham Waliza: Sign Language Recognition System using Convolutional Neural Network and Computer Vision (2020) [2]

This paper has achieved a recognition of 99% in the common vocabulary dataset.

2.3.1 DATA COLLECTION

They created their own dataset of ASL having 2000 images of 10 static alphabet signs. Two datasets have been made by 2 different signers. Each of them has performed one alphabetical gesture 200 times in alternate lighting conditions.. Out of the 2000 images captured, 1600 images are used for training and the rest for testing.

2.3.2 DATA PROCESSING

2.3.2.1 HSV color space and background elimination

Since the images obtained are in RGB color spaces, they are transformed into HSV color space. The region of the hand gesture undergoes dilation and erosion operations with an elliptical kernel. The first image is obtained after applying the 2 masks.

2.3.2.2 Segmentation

The first image is then transformed to grayscale. Non-black pixels in the transformed image are binarised while the others remain unchanged, therefore black. The hand gesture is segmented firstly by taking out all the joined components in the image and secondly by letting only the part which is immensely connecte. The frame is resized to a size of 64 by 64 pixels. At the end of the segmentation process, binary images of size 64 by 64 are obtained where the area in white represents the hand gesture, and the black coloured area is the rest.

2.3.2.3 Feature Extraction

Feature extraction reduces the data after having extracted the important features automatically. It also contributes in maintaining the accuracy of the classifier and simplifies its complexity. Scaling the images to 64 pixels has led to getting sufficient features to effectively classify the ASL gestures .

2.3.2.4 Classification

The convolution layers of 2D CNN scan the images with a filter of size 3 by 3. The dot product between the frame pixel and the weights of the filter are calculated. The pooling layers are then applied after each convolution layer. One pooling layer decrements the activation map of the previous layer. It merges all the features that were learned in the previous layers' activation maps. The max pool layer has a size of 2×2. The dropout is set to 50 percent and the layer is flattened. The last layer of the network is a fully

connected output layer with ten units, and the activation function is Softmax. Then compile the model by using category cross-entropy as the loss function and Adam as the optimiser.

2.3.3 EVALUATION

The model is evaluated based on 10 alphabetic American sign languages including : A, B, C, D, H, K, N,O,T and Y. We have used a total of 2000 images to train the Convolutional Neural Network. The dataset is split in the ratio of 80:20 for training and testing respectively. The results used in this paper gives us an accuracy of over 90.0%.

2.4 Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, Indra R: Real-time Conversion of Sign Language to Text and Speech [3]

2.4.1 INTRODUCTION

Sign Languages allow the dumb and deaf people to communicate with each other and the rest of the world. American Sign Language has been created to reach the wider public and acts as the primary sign language of the Deaf populations in the United States and much of Anglophone Canada, also including most of West Africa and areas of Southeast Asia. An android app is developed to convert sign language gestures to text and speech.

2.4.2 METHODOLOGY

In this paper, 26 ASL alphabets are used along with 1 customized symbol for ‘Space’ which is to be recognized in real-time using a smartphone. The algorithm is developed on top of a Java-based OpenCV wrapper. The entire system was developed using images that are of 200 x 200 pixels in RGB format.

Detection Method - Contour Mask, Canny Edges, Skeleton
Kernel - Linear, Radial Basis Function (RBF)
Dimensionality Reduction Type - None, Principal Component Analysis

2.4.3 IMPLEMENTATION

2.4.3.1 Calibration

Here, color-based segmentation has been implemented using libraries present by OpenCV. This can be done by understanding all the different skin tones and their HSVA (Hue, Saturation, Value, Alpha) configurations. The following lower and upper bounds define all the skin tones possible. Only if the image possesses pixel values in this range, the frame will be considered for classification else it will be discarded.

2.4.3.2 Processing of Frame

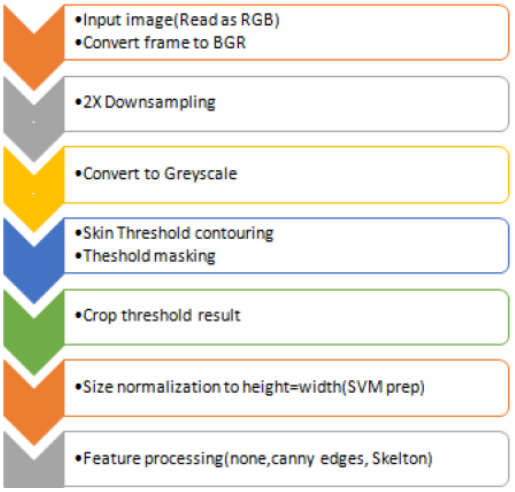

fig 2.4.3.2.1 Frame Processing Diagram

OpenCV converts the BGR that is taken from the camera to RGB. It is then downsampled by throwing away unnecessary information. Different feature preprocessing algorithms can now be applied, which consist of contour mask, canny edges, skeleton explained further in the paper.

2.4.3.3 Detection Methods

2.4.3.3.1 Contour Masking

Contours may be defined precisely as a curve that connects all the continuous points (along the boundary), with the same color or intensity.

2.4.3.3.2 Skeletonization

Skeletonization is a method for reducing foreground regions to a skeletal remnant in a binary picture that essentially retains the magnitude and continuity of the original area while removing much of the original foreground pixels.

2.4.3.3.3 Canny Edges

A Gaussian filter is applied to make the image smooth and remove the noise. Intensity gradients of the image are calculated. Non-maximum suppression is applied to remove the possibility of a false response. Double thresholding is done to detect or determine the possible edges. Edges are finalized by identifying and removing all other edges that are weak and not linked to strong edges.

2.4.3.3.4 Kernel

Kernels in SVM classification refer to the function that is responsible for defining the decision boundaries between the classes. The SVM software has been used with the Linear Kernel and the RBF (radial basis function) kernel.

2.4.3.3.5 Dimensionality Reduction

PCA uses a list of the principal axes to identify the underlying dataset before classifying it according to the amount of variance identified by each axis. PCA makes the maximum variability of the data set more visible by rotating the axes. The number of feature combinations is equal to the number of dimensions of the dataset and, in general, the maximum number of PCAs that can be constructed.

2.4.3.3.6 Classification

SVM (Support Vector Machine) is a supervised learning technique. The objective is to find a hyperplane that distinctly classifies data points into classes. 27 classes have been in our model each corresponding to letters in the English language. The SVM model will classify the images into the 27 classes to yield a result.

2.4.3.3.7 Post Processing

User interface string writing is used to print a message for an error. This is to make the UI user friendly to the users. Debugging is mainly used for inspection of strings. This will help in removing errors and increasing the work efficiency of the app.

2.5 Rahul Chauhan, R.C Joshi, Kamar Kumar Ghanshala: Convolutional Neural Networks for Image Detection and Recognition (2018) [4]

2.5.1 INTRODUCTION:

CNN is a powerful deep learning algorithm capable of dealing with millions of parameters and saving the computational cost by inputting a 2D image and convolving it with filters/kernel and producing output volumes. The MNIST dataset is a dataset containing handwritten digits and tests the performance of a classification algorithm.

2.5.2 CNN MODEL FOR MNIST DATASET:

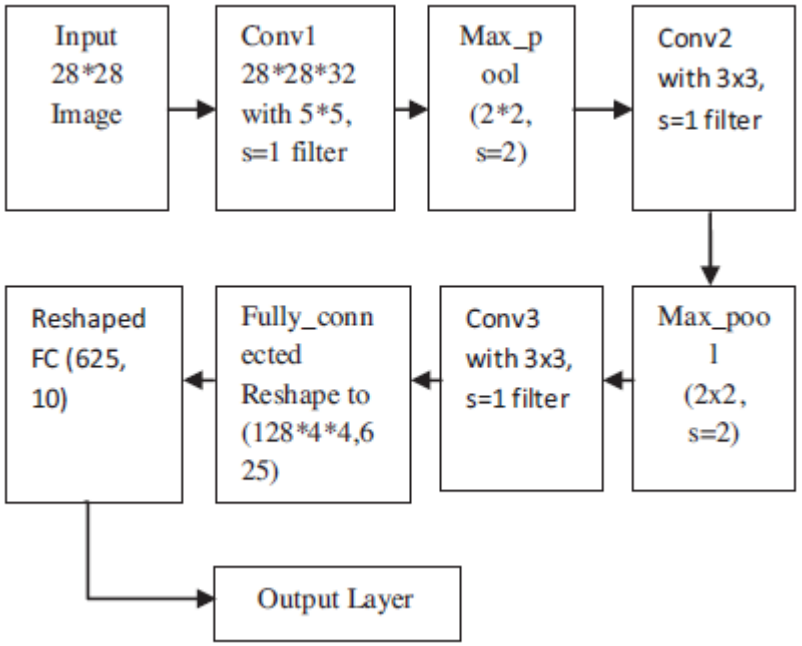

fig 2.5.2.1: CNN Model of MNIST dataset

2.5.3 RELU NONLINEARITY ACTIVATION FUNCTION:

Relu non linearity works several times faster than the tanh non – linearity, and results in the output of the activation as either 0 or a positive number. With relu it is easier to train larger neural networks.

2.5.4 Learning Rate:

The learning rate determines the steps the algorithm will take to converge to the global minimum. Thus the learning rate is a hyper parameter that needs to be finely tuned and can be done with the help of learning rate decay. In learning rate decay, the learning rate decays exponentially after every epoch.

2.5.5 Result Analysis

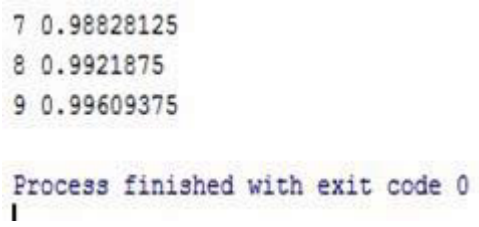

fig 2.5.5.1: Accuracy of CNN model on MNIST in 10 epochs

The result shows that as the number of epochs increases, the best accuracy achieved in recognizing the digits on the MNIST data set is 99.6 % with 10 epochs.

2.5.6 Conclusion:

The calculated accuracy on MNIST is 99.6%. The accuracy on training set may also be improved further by adding more hidden layers. And this system can be implemented as a assistance system for machine vision for detecting nature language symbols,

Chapter 3

PROPOSED METHOD

3.1 OBJECTIVE

Deaf people do not have that many options for communicating with a hearing person, and all of the alternatives have some major flaws. Interpreters are not available easily, and also can be expensive. Affordable and always available interpreter services are in huge demand in the deaf community. Every day thousands of local businesses around the globe face problems with providing their services to deaf. According to the National Deaf Association (NAD), 18 million people are estimated to be deaf in India.

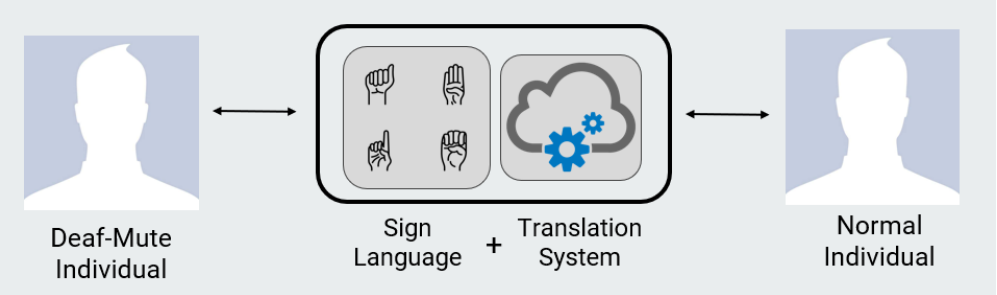

fig 3.1.1: Objective of the Solution

The main objective is to design a software that will help drive inclusivity at the workplace by removing communication barriers between the disabled and able. The new application can find use in a B2B setting, where businesses who want to employ deaf and mute employees can use it to convey employee messages to the end consumer.

An easy to use innovative digital translator that is compellingly fast, easy, comfortable and economical is the need of the hour.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1 Purpose

The aim of this document is to provide a detailed description of the translator of Sign language to text. It will cover the applications and features of the system, the interfaces of the system, what the system is expected to do, the constraints that the project will work under and how it behaves in response to external stimuli. This is intended for both the developers and the users of this system.

3.2.2 Scope

This system is primarily intended for making an interpreter. This will have applications in business who want to employ deaf and mute employees can use it to convey employee messages to the end consumer. It will be used majorly by the deaf and mute to communicate.

3.2.3 Glossary

1. Feature: Features are individual measurable properties or characteristics of a phenomenon being observed. These require classification.
2. Label: Labels are the final output. We can also consider the output classes to be the labels.
3. Model: A machine learning model is a mathematical portrayal of a real-life problem. There are various algorithms that perform different tasks with different levels of accuracy.
4. Regression: Regression is a statistical method that is used to predict real and continuous valued functions.
5. Classification:In classification, we will need to categorize data into a finite number of predefined classes.
6. CNN: It is a Machine Learning unit algorithm, for supervised learning, which is used in classification of large amounts of data.
7. Image Processing: The various modifications done on a raw image to make it suitable for the training model.
8. Training-set: This is the data set over which CNN model is trained. The predictions are completely dependent on the training-data set.

3.2.4 Overview of Document

The next section, the Overall Description section, of this document gives an overview of the functionality of the project. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next section. The third section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product. Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different languages.

3.2.5 Overall Description

3.2.5.1 System Architecture

The main objective of the software is to classify the gestures and label them with one of the various categories already defined while training the dataset. We have then tested the same for some data with the help of deep learning.

The system has been trained on previously captured gestures which have been labeled with the text associated with them. Multiple copies of each gesture image have been created to extract the features efficiently, then the CNN model is trained. The new gesture which has to be translated, has to be signaled in front of the camera connected to the system, which will be recorded and matched and classified using CNN algorithm.

3.2.6 Requirement Specification

3.2.6.1 Functional Requirements

The Prime objective of the software is to translate the sign language into text. Initially, one character was translated at a time and later, the software was trained and developed to translate even words. Words can be formed by concatenating various characters as well and the formed word will be displayed on the output window.

The captured images need to be pre-processed. The system modified the images captured and trained the model to classify the signals in one of these defined labels.

3.2.6.2 Non-Functional Requirements

The sole purpose of the software is to facilitate communication for the disabled.The previous available devices were slow and inefficient. Thus, the software is built to translate the signs accurately and at a relatively faster rate. The software is designed efficiently such that it can be modified easily making it easy to maintain.

3.3 OVERVIEW OF APPROACH

We have created an User Interface which allows the User to add new gestures with some specific meaning, and the option to train the model with the added features. The UI also directs the user to ”hand recognition” window, which will further be used to do gestures. Then finally two options are provided for translating sign language to Text, one allowing one character at a time, and second to concatenate the characters to form a complete word.

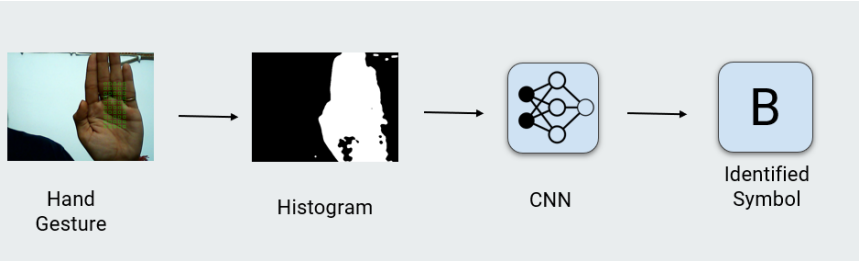

fig 3.3.1: Flowchart of the solution

There are a total of 29 gestures for which the model is trained, including 26 English alphabets and 3 other commonly used symbols. 2700 images of each gesture have been captured.

A histogram has been created which identifies the skin of the hand of the speaker and separates it from the background. The Neural Network is trained using Tensorflow Object Detection on each gesture. Whenever the translator is to be used, the person uses the sign language to be recorded in the camera of the device with the software, which then fetches it into the Neural Network and the sign is classified.

3.4 PRE-PROCESSING THE DATA

3.4.1 Histogram creation

The OpenCV library has been used to generate a histogram that will separate the hand gestures from the background. For this purpose 50 squares in the form of 5*10 are displayed and the hand must cover all the squares. Then the image is captured and a histogram is plotted of the area covering the squares.

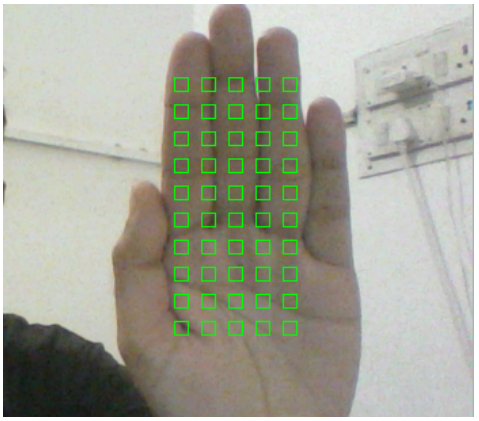

fig 3.4.1.1: Hand detection

3.4.2 Image Processing

fig 3.4.2.1: Histogram of the hand

The images captured of the gestures using the intensities obtained from the histogram are then processed. The images are resized to 50*50 pixels and then converted to gray scale.Each image is then flipped along the vertical axis.

3.4.3 Convolution

The image obtained is then convoluted with the feature detector to form the feature map. This is the most important part in feature detection.The image is convoluted with a number of features and hence a number of feature maps are present. Larger the number of features, the better it is to classify the image.

The main aim of convolution is to detect features.

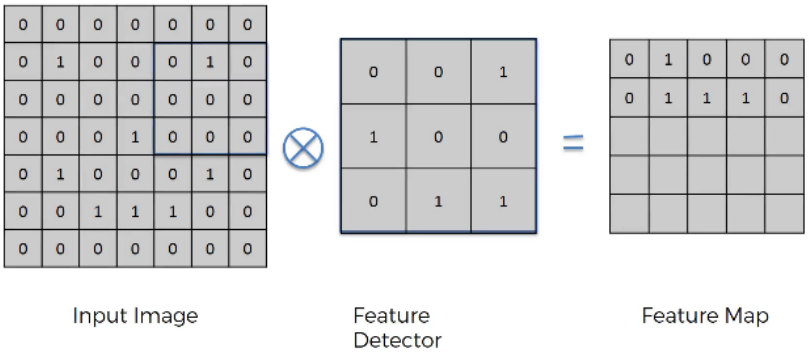

fig 3.4.3.1: Convolution of a matrix

3.4.4 ReLu Layer

The Convoluted image is then passed through the Activation function. The activation function used is the RECTIFIER FUNCTION.This is used to increase the non-linearity in the image.

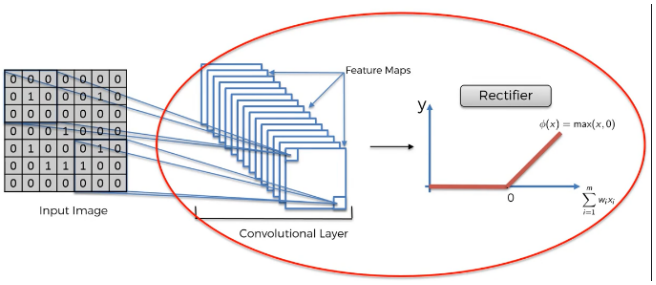

Fig 3.4.4.1: Activation function: Rectifier function

3.4.5 Max pooling

Now the feature map contains the convoluted result. There are a large number of such maps, hence enormous data. Further, one feature may differ in size, orientation in different images. Both these issues are resolved using Max Pooling. A small grid is selected and then the maximum value is preserved, reducing the size of data as well.

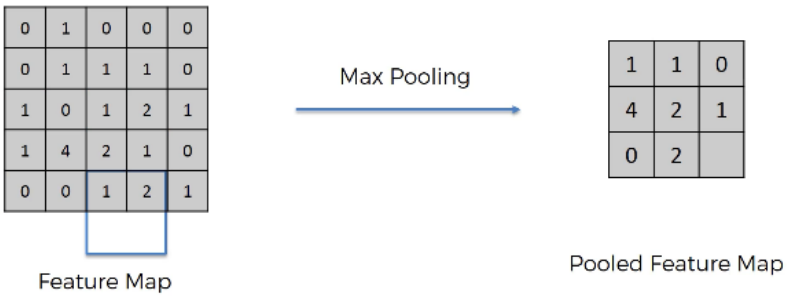

fig 3.4.5.1: Max Pooling

3.4.6 Flattening

The pooled image features need to be flattened so that they can be used as input in the next Artificial Neural Network. These form the input layer of the Artificial Neural Network.The max pooling output is transformed into a column.

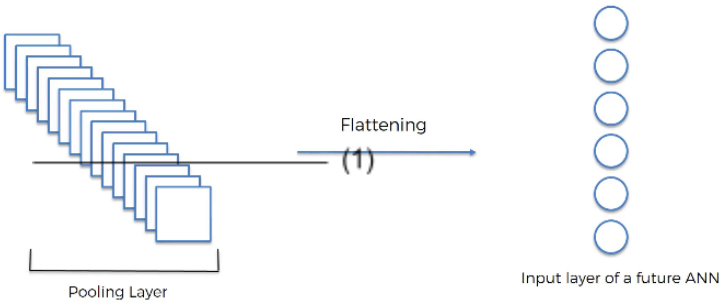

fig 3.4.6.1: Flattening

3.5 BUILDING THE MODEL USER INTERFACE

The CNN is a type of Artificial Neural Network that is used in Image recognition and processing. It detects specific features in the images and then classifies them accordingly based on the presence of those features.In our model we have 5 layers in total including 3 Convolutional layers and 2 fully connected Dense layers.
The first CNN layer consists of 16 neurons, the second layer consists of 32 neurons
and the third layer consists of 64 neurons.

The first dense layer consists of 128 neurons and uses the Rectifier function as the Activation function. The second dense layer uses Softmax function as the Activation function

3.5.1 Activation functions .

Activation Function is the function that is applied to the sum of weighted inputs to the neuron. This is where calculations happen. In our neural network, we have used two different activation functions

Relu Activation Function: ReLU stands for Rectified Linear Unit. Its cheap to compute as there is no complicated math. The model can therefore take less time to train or run. This is used to increase the non-linearity of the images.It is especially useful when dealing with small values as in our case.The formula for ReLu function is given by:
y = max(0, x)

Softmax Activation function:Softmax is an activation function. It is frequently used in classifications.

Softmax output is large if the score is large. Its output is small if the score is small. The proportion is not uniform. Softmax is exponential and enlarges differences. Formula for Softmax function is given by (1).

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad \text{——————} \quad (1)$$

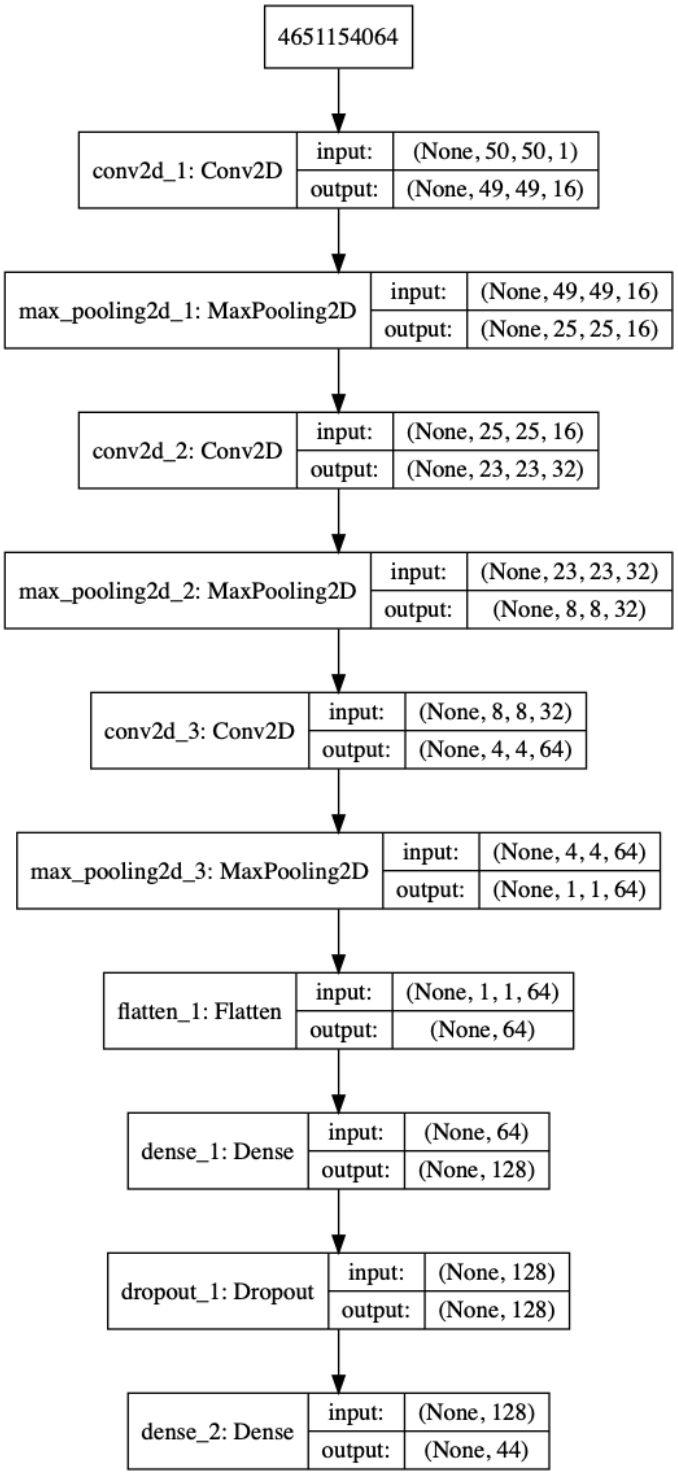

fig 3.5.1.1: Model consisting of 5 layers

3.6 DATA FLOW DIAGRAM

3.6.1 Zero level DFD

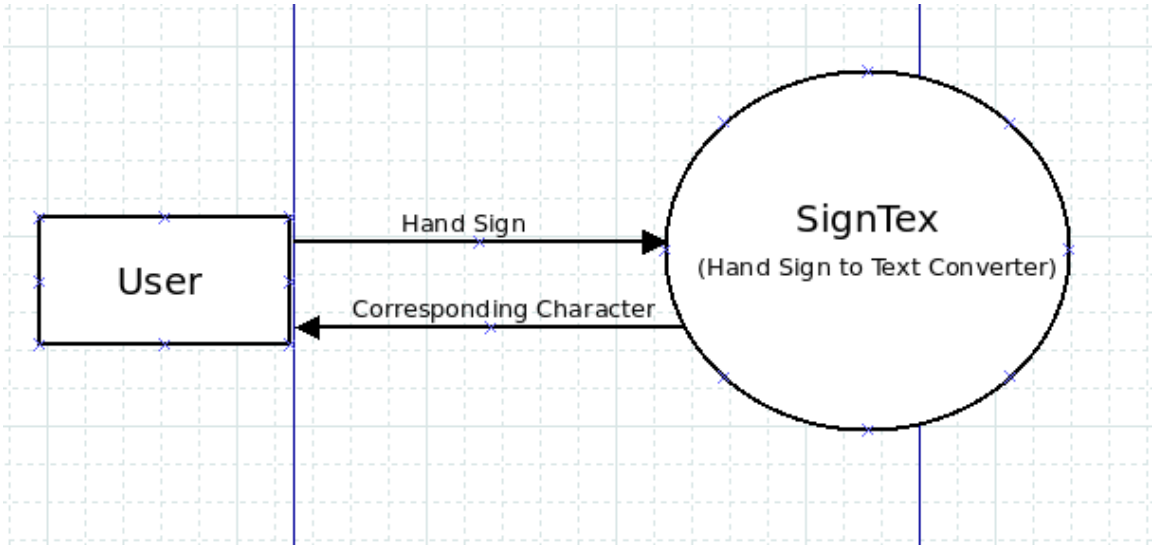

fig 3.6.1.1: Zero Level DFD

3.6.2 First level DFD

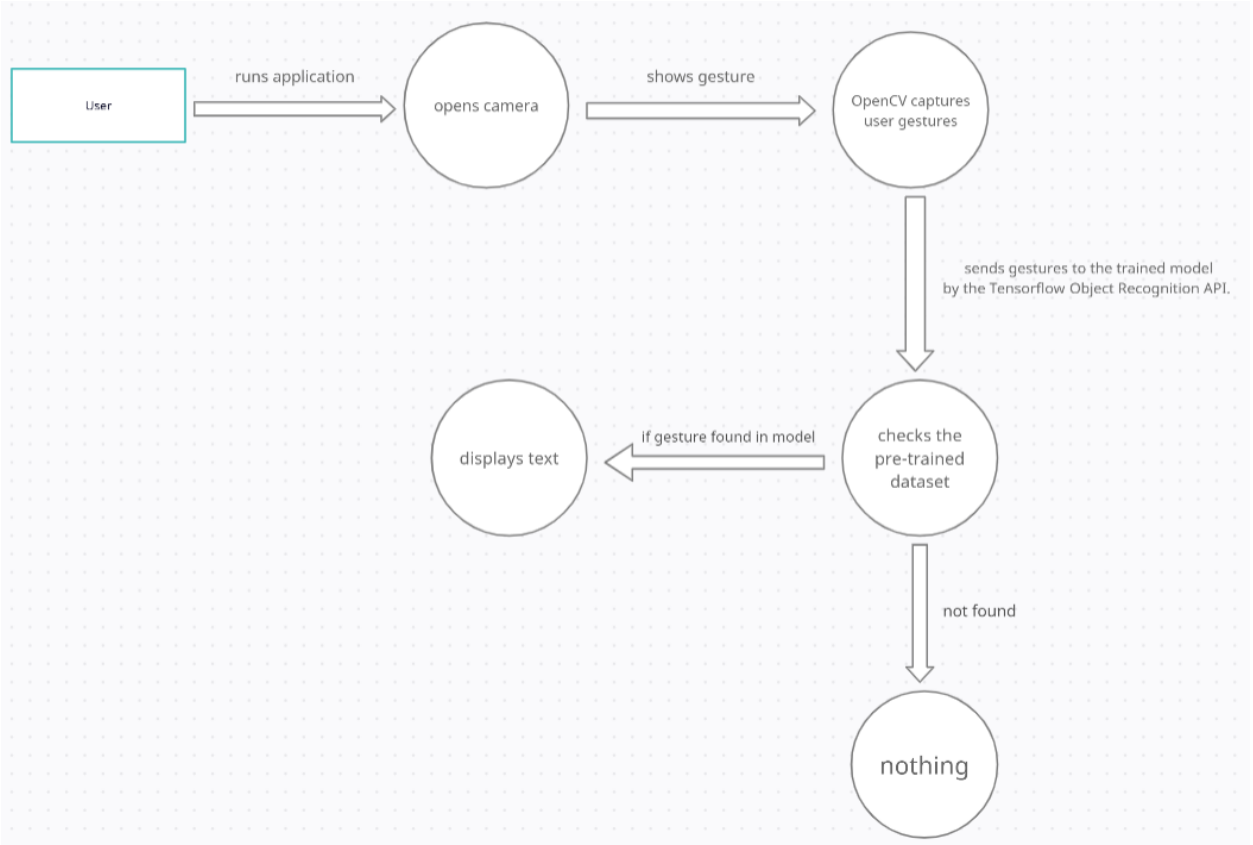

fig 3.6.2.1: First Level DFD

3.7 USER INTERFACE

An user-friendly User Interface is created using Tkinter. It provides buttons for various operations including Adding more gestures, Setting the histogram,Recognize Gesture as a Character.

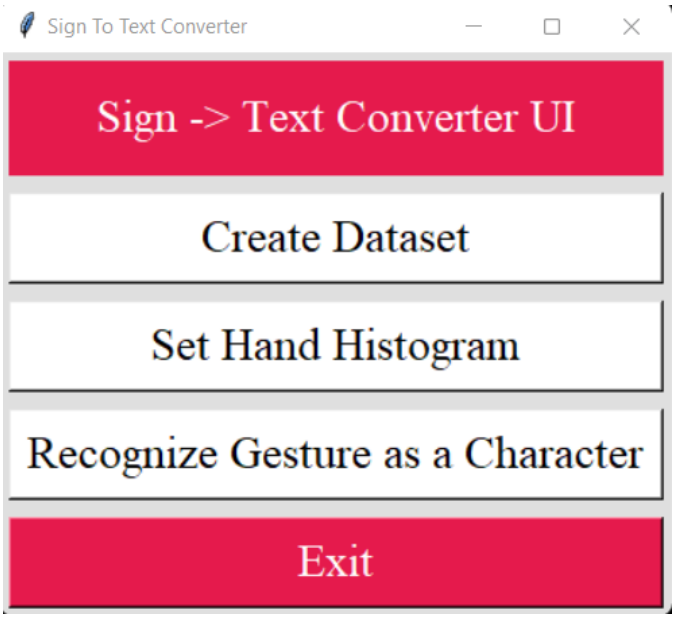

fig 3.7.1: User Interface

Chapter 4

RESULTS AND DISCUSSIONS

4.1 RESULTS

The model is translating the sign language to the English alphabet characters precisely. Some of the translated outputs are:

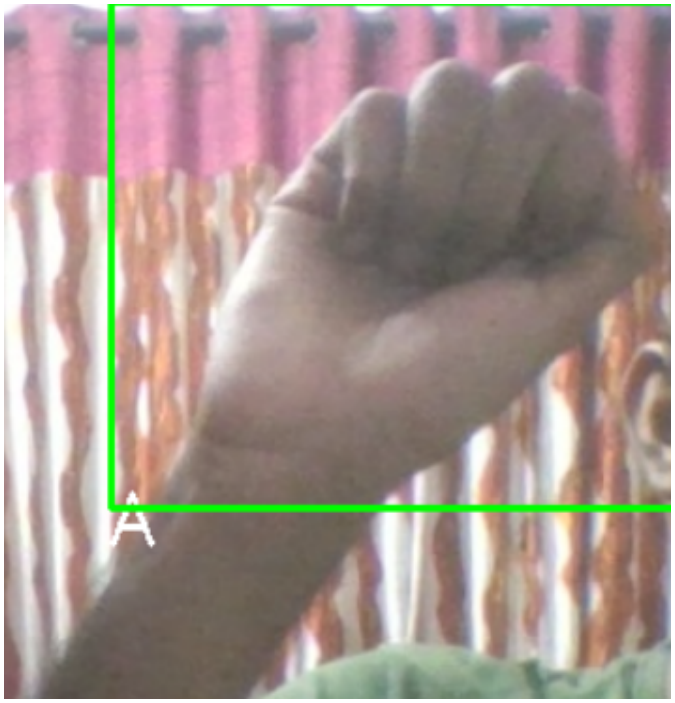

fig 4.1.1:Sign Language To Text for ‘A’

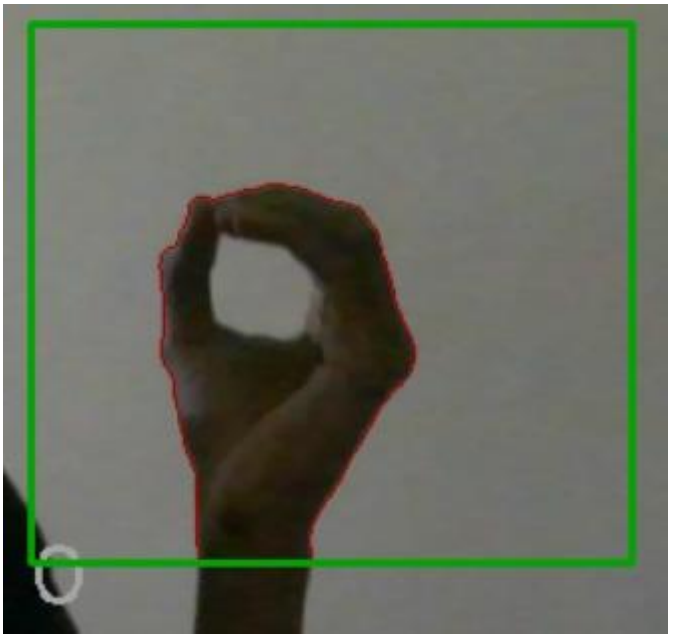

fig 4.1.2: Sign Language To Text for ‘O’

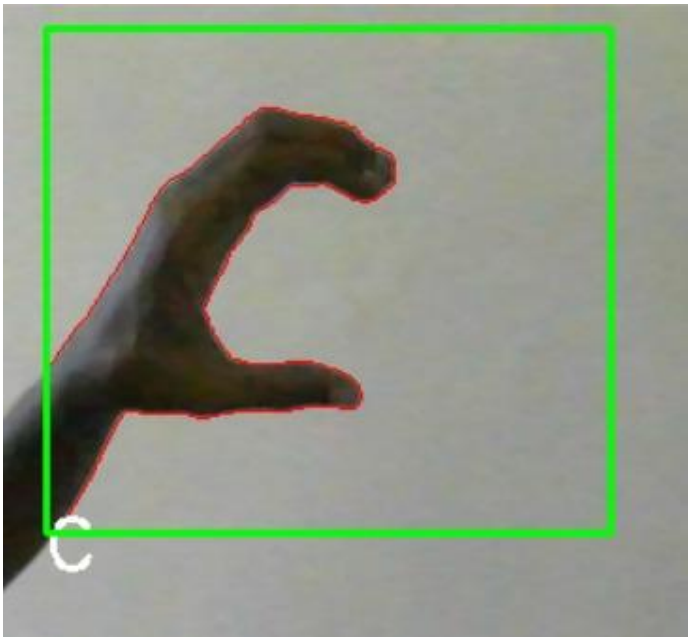

fig 4.1.3: Sign Language To Text for ‘C’

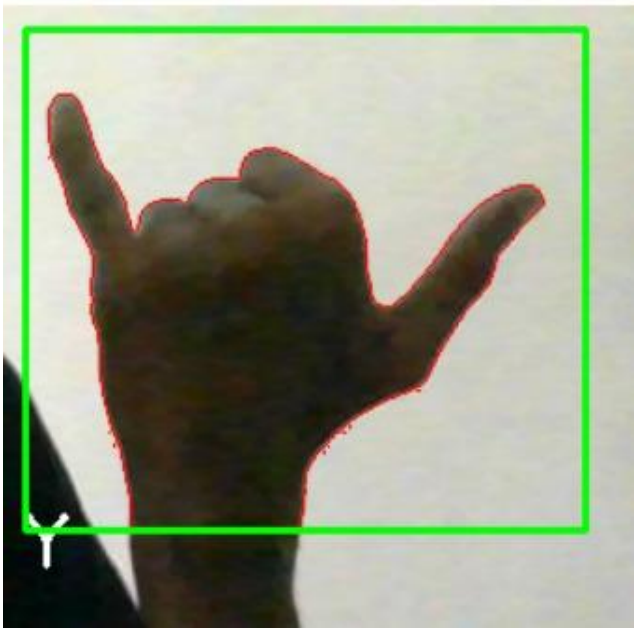

fig 4.1.4: Sign Language To Text for ‘Y’

fig 4.1.5: Creating Histogram

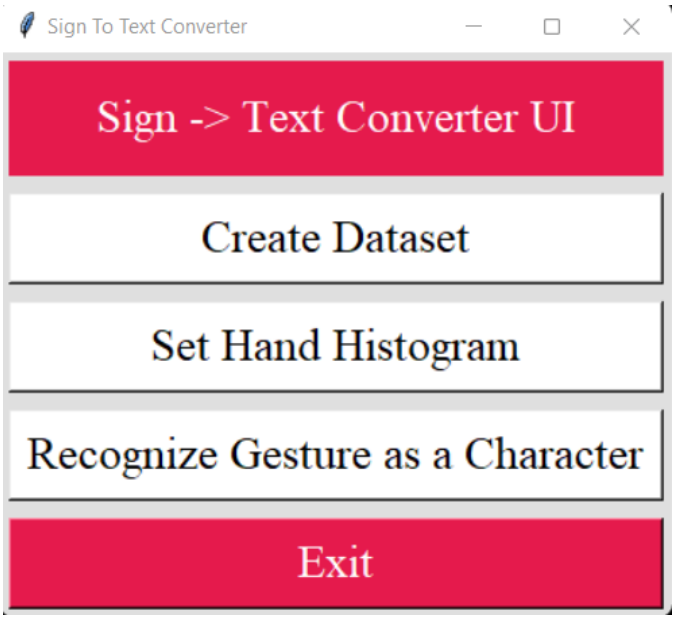

fig 4.1.6 :Sign Language To Text GUI

4.1.1 Confusion Matrix

Classification was using Convolutional Neural Network. The accuracy obtained on the test data was 0.9997 and misclass was 0.0003

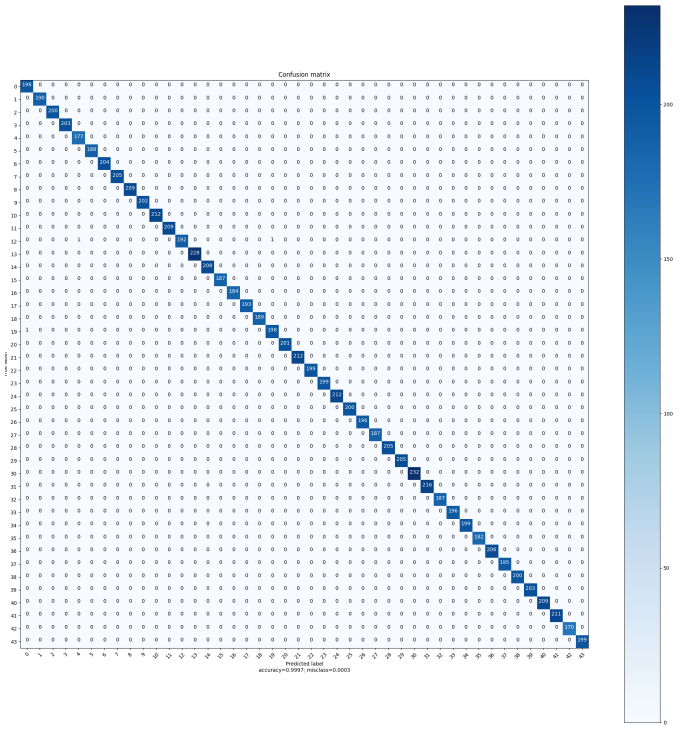

fig 4.1.3.1: Confusion Matrix

4.2 ANALYSIS

The accuracy of the translator is increasing as the number of epochs are increased while training the model and when the number of images for each gesture are increased. Epochs : It is the number of times the model is trained over the same data-set.

4.2.1 Recall

It is the fraction of the total number of relevant instances that were retrieved.

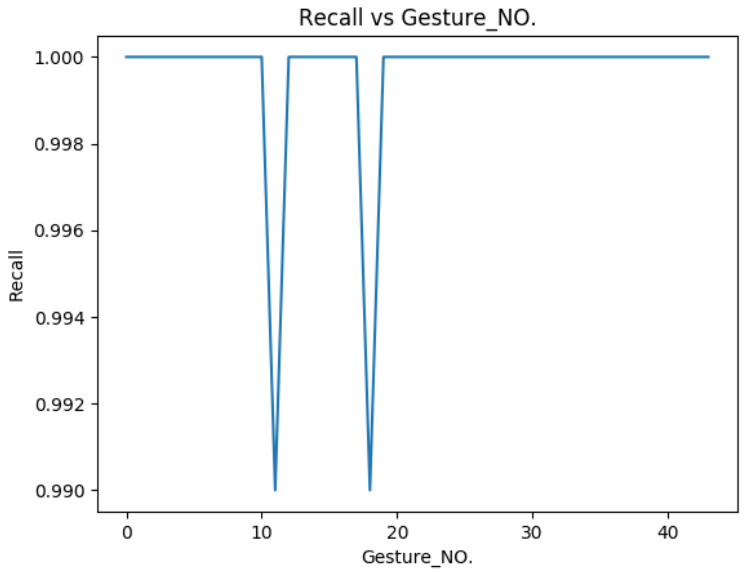

fig 4.2.1.1Recall vs Gesture_No

4.2.2 Precision

It is defined as the fraction of the relevant instances among the instances retrieved.

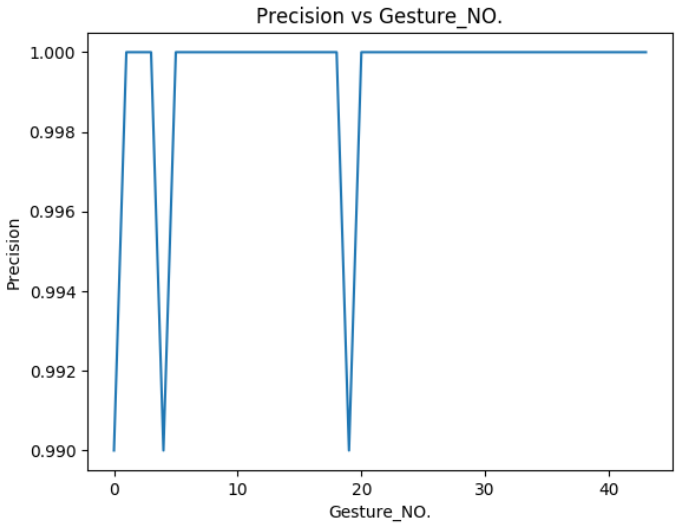

fig 4.2.2.1:Precision vs Gesture_No

4.2.3 F1 - score

It is defined as the harmonic mean of the Recall and the Precision. In this, even the false negative and false positive are crucial. It tells the accuracy of the classifier in classifying the data points in that particular class compared to all other classes.

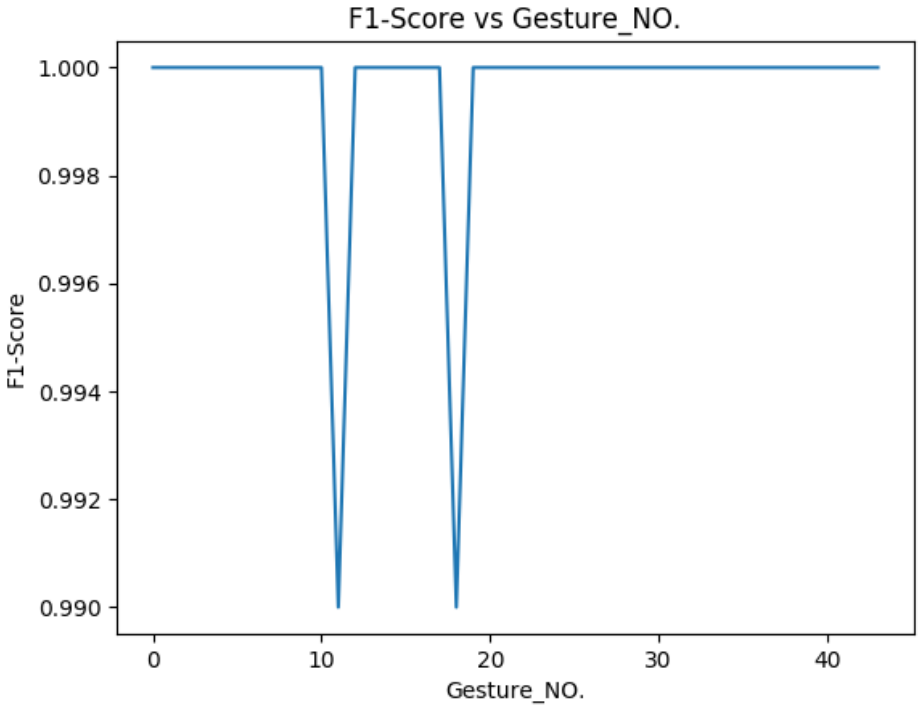

fig 4.2.3.1:F1-Score vs Gesture_No

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

A translator that can translate 44 gestures to Text and even concatenate the translated characters to form words was successfully developed. We have presented our results and have plotted the graphs for the precision, recall and F1 scores.The translator can be made more effective by adding more gestures. As suggested and guided by our project mentor, a lot of measures were taken and modifications made to tackle all the problems coming up in the project that required our undivided attention.

5.2 APPLICATIONS

1. Deaf/dumb people will be able to convey their thoughts with hand gestures without a translator.
2. Sensor based gloves can work even in dark and areas with high noise images.

5.3 LIMITATIONS

1. System will fail to work in accelerated motions.
2. Image processing based implementation would fail in the dark.
3. Complicated gestures involving motion cannot be identified.
4. Object prediction will be inaccurate when more than one hand is within the fixed background.
5. Object detection will be inaccurate when there is noise in the background.

5.4 FUTURE SCOPE

1. Adjustment to count for system in motion.
2. Identify complicated gestures involving two hands and motion.
3. Use hand gestures to control and automate other devices.
4. Convert sign language to both text and audio

REFERENCES

1. Sankit Ojha, Ayush Pandey, Shubham Maurya: Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network (2020)

2. Mehreen Hurroo, Mohammad Elham Walizad :Sign Language Recognition System using Convolutional Neural Network and Computer Vision (2020)

3. Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, Indra R: Real-time Conversion of Sign Language to Text and Speech (2020)

4. Rahul Chauhan, Kamal Kumar Ghanshala, R.C Joshi:Convolutional Neural Network (CNN) for Image Detection and Recognition (2018)

5. Zhi-jie Liang, Sheng-bin Liao, Bing-zhang Hu: 3D Convolutional Neural Networks for Dynamic Sign Language Recognition (2018)

6. https://www.tensorflow.org/object_detection: Tensorflow Object Detection

7. https://machinelearningmastery.com: To get more information about machine learning

APPENDIX A

BASE PAPER

2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)

Convolutional Neural Network (CNN) for Image Detection and Recognition

Rahul Chauhan
Graphic Era Hill University
Dehradun, India
chauhan14853@gmail.com

Kamal Kumar Ghanshala
Graphic Era University
Dehradun, India
kamalghanshala@gmail.com

R.C Joshi
Graphic Era University
Dehradun, India
rcjoshi.geu@gmail.com

***Abstract*- Deep Learning algorithms are designed in such a way that they mimic the function of the human cerebral cortex. These algorithms are representations of deep neural networks i.e. neural networks with many hidden layers. Convolutional neural networks are deep learning algorithms that can train large datasets with millions of parameters, in form of 2D images as input and convolve it with filters to produce the desired outputs. In this article, CNN models are built to evaluate its performance on image recognition and detection datasets. The algorithm is implemented on MNIST and CIFAR-10 dataset and its performance are evaluated. The accuracy of models on MNIST is 99.6 %, CIFAR-10 is using real-time data augmentation and dropout on CPU unit.**

Keywords*- *Deep Learning, Handwritten digit Recognition, Object Detection, Convolutional neural networks, MNIST, CIFAR-10, Dropout, Overfitting, Data Augmentation, Relu

I INTRODUCTION

Image Recognition and detection is a classic machine learning problem. It is a very challenging task to detect an object or to recognize an image from a digital image or a video. Image Recognition has application in the various field of computer vision, some of which include facial recognition, biometric systems, self-driving cars, emotion detection, image restoration, robotics and many more[1]. Deep Learning algorithms have achieved great progress in the field of computer vision. Deep Learning is an implementation of the artificial neural networks with multiple hidden layers to mimic the functions of the human cerebral cortex. The layers of deep neural network extract multiple features and hence provide multiple levels of abstraction. As compared to shallow networks, this cannot extract or work on multiple features. Convolutional neural networks is a powerful deep learning algorithm capable of dealing with millions of parameters and saving the computational cost by inputting a 2D image and convolving it with filters/kernel and producing output volumes.

The MNIST dataset is a dataset containing handwritten digits and tests the performance of a classification algorithm. Handwritten digit recognition has many applications such as OCR (optical character recognition), signature verification, interpretation and manipulation of texts and many more[2,3]. Handwritten digit recognition is an image classification and recognition problem and there have been recent advancements in this field [4]. Another dataset is CIFAR-10 which is an object detection datasets that classifies the objects into 10 classes and detects the objects in the test sets. It contains natural images and helps implement the image detection algorithms.

In this paper, Convolutional neural networks models are implemented for image recognition on MNIST dataset and object detection on the CIFAR-10 dataset. The implementation of models is discussed and the performance is evaluated in terms of accuracy. The model is trained on an only CPU unit and real-time data augmentation is used on the CIFAR-10 dataset. Along with that, Dropout is used to reduce Overfitting on the datasets.

The remaining sections of the paper are described as follows: Section 2 describes a brief literature survey; Section 3 describes the classifier models with details of the techniques implemented. Section 4 evaluates the performance of the model and describes the results. Section 5 summaries the work with future works.

II. LITERATURE SURVEY

In recent years there have been great strides in building classifiers for image detection and recognition on various datasets using various machine learning algorithms. Deep learning, in particular, has shown improvement in accuracy on various datasets. Some of the works have been described below:

Norhidayu binti Abdul Hamid et al. [3] evaluated the performance on MNIST datasets using 3 different classifiers: SVM (support vector machines), KNN (K-nearest Neighbor) and CNN (convolutional neural networks). The Multilayer perceptron didn't perform well on that platform as it didn't reach the global minimum rather remained stuck in the local optimal and couldn't recognize digit 9 and 6 accurately. Other classifiers, performed correctly and it was concluded that performance on CNN can be improved by implementing the model on Keras platform. Mahmoud M. Abu Gosh et al. [5] implement DNN (Deep neural networks), DBF (Deep Belief networks) and CNN (convolutional neural networks) on MNIST dataset and perform a comparative study. According to the work, DNN performed the best with an accuracy of 98.08% and other had some error rates as well as the difference in their execution time.

Youssouf Chherawala et al. [6] built a vote weighted RNN (Recurrent Neural networks) model to determine the significance of feature sets. The significance is determined by weighted votes and their combination and the model is an application of RNN. It extracts features from the Alex word images and then uses it to recognize handwriting. Alex krizhevsky [7] uses a 2-layer

978-1-5386-6373-8/18/$31.00 ©2018 IEEE

2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)

Convolutional Deep belief network on the CIFAR-10 dataset. The model built classified the CIFAR-10 dataset with an accuracy of 78.90% on a GPU unit. Elaborative differences in filters and their performance is described in the paper which differs with every model.

Yehya Abouelnaga et al. [8] built an ensemble of classifiers on KNN. They used KNN in combination with CNN and reduce the Overfitting by PCA (Principal Component Analysis). The combination of these two classifiers improved the accuracy to about 0.7%.

Yann le Cunn et al. [1] give a detailed introduction to deep learning and its algorithms. The algorithms like Backpropagation with multilayer perceptron, Convolutional neural networks, and Recurrent neural networks are discussed in detail with examples. They have also mentioned the scope of unsupervised learning in future in Artificial intelligence.

Li Deng [10] details a survey on deep learning, its applications, architectures, and algorithms. The generative, discriminative and hybrid architectures are discussed in detail along with the algorithms that fall under the respective categories. CNN, RNN, Autoencodes, DBN's, RBM's (Restricted Boltzmann machines) are discussed with their various applications.

III. CLASSIFIER MODELS

A. Datasets

MNIST is the dataset used for image recognition i.e. for recognition of handwritten digits [11]. The dataset has 70,000 images to train and test the model. The training and test set distribution is 60,000 train images and 10,000 test images. The size of each image is 28x28 pixels (784 pixels) which are given as input to the system and has 10 output class labels from (0-9). Fig.1 shows a sample picture from MNIST dataset [13].

CIFAR-10 is the dataset used for object detection which is labeled a subset of 80 million tiny images [12]. The dataset has 60,000 32x32 pixel color images with 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Each class has 6000 images. The train batch has 50,000 images and test batch has 10,000 images. The test batch for each class has 1000 images which are randomly selected. Fig.2 shows sample pictures from CIFAR-10 dataset [14].

B. CNN Models

Convolutional neural networks are deep learning algorithms that take input images and convolves it with filters or kernels to extract features. A NxN image is convolved with a fXf filter and this convolution operation learns the same feature on the entire image[18]. The window slides after each operation and the features are learnt by the feature maps. The feature maps capture the local receptive field of th eimage and work with shared

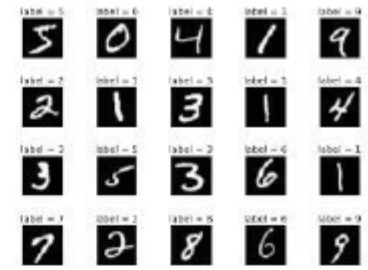

Fig.1 A sample MNIST Dataset

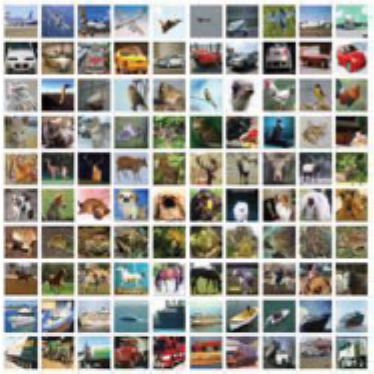

Fig.2 A CIFAR-10 dataset images

weights and biases [15,21]. Equation (1) shows the size of the output matrix with no padding and Equation (2) shows the convolution operation. In order to preserve the size of input image padding is used . In a 'SAME' padding the output image size is same as input image size and a "VALID" padding is no padding. The size of th eoutput matrix with padding is depicted in equation (3).

$$NXN * fXf = N - F + 1 \quad (1)$$

$$O = \sigma(b + \sum_{l=0}^{2}\sum_{j=0}^{2} w_{l,j}\, h_{a+l,b+j}) \quad (2)$$

$$NXN * f * f = (N + 2P - f)/(s + 1) \quad (3)$$

Here, O is the output , P is the padding, s i sthe stride, b is the bias, σ is the sigmoidal activation function, w is a 3x3 weight matrix of shared weights and $h_{x,y}$ is the input activation at position x, y. CNN has application in fields of large scale image recognition [17], Emotion detection through speech [9] [19] facial expression recognition [20], biometric systems, genomics and many others.

C. CNN model for MNIST dataset

The CNN model for MNIST dataset is shown in figure (c). The input image is a vector with 784 pixel values. It is input into the convolutional model where the convolution layers along with filters generate th efeature maps using the local receptive field. The pooling and fully connected layers follow the convolution layers. Dropout is

2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)

introduced after each convolutional layer. The pooling layers simplify the output after convolution. There are two types of pooling: Max pooling and L2 pooling. In max pooling the maximum activation output is pooled into a 2 x 2 input region and L2 pooling takes the square root of the sum of squares of the activation 2x2 region. Finally, the fully connected layers connect each layer of max pooling layer to the output neurons. The architecture of the developed model is as follows:

- Batch Size (training): 128, Batch size (test): 256, Number of epochs:10
- Dropout: Yes
- Optimizer: RMS prop, Learning rate=0.001, the parameter β : 0.9
- Keep_prob:0.8

The architecture of the model is :

Convolution_layer 1 → Relu → Max_pool → dropout → Convolution_layer 2 → Relu → Max_pool → dropout → Convolution_layer 3 → Relu → Max_pool → fully_connected → dropout → output_layer → Result

The input is a 28X28 image which passed to filters to generate fetaure maps. . The first filter is of size 5x5x1x32 (32 features to learn in the first hidden layer), 3x3x32x64 for the second convolution layer (64 features to learn from second hidden layer), 3x3x64x128 for the third layer, (128*4*4,625) for the fourth layer and (625,10) for the last layer. The stride is 1 for convolution layer and 2 for max-pooling layers. Stride defines the number of block to move forward after one calculation. Generally, the value of stride for convolution layer is 1 and for pooling layer is 2. The accuracy of this model is 99.6%.

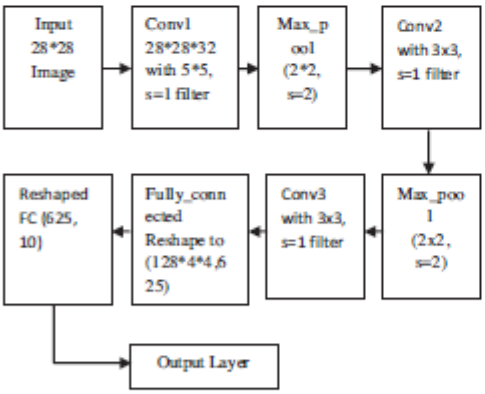

Fig. 3. CNN model for MNIST dataset

D. *CNN model for CIFAR-10 dataset*

The CNN model for CIFAR-10 dataset is as follows:

- Batch size: 32, Number of epochs:50
- Optimizer: RMS Prop, Decay rate= 1e-6
- Data augmentation: True, Rotation ; in range of 0-180, horizontal flip: TRUE, Vertical flip: FALSE
- Dropout : True

The architecture is as follows:

Conv1→Relu→Conv2→Relu→Max_pooling→Dropout →Conv3→Relu→Conv4→Relu→Max_pooling→Dropout→Conv5→Relu→Max_pooling→Dropout→Flatten→ Dense→Relu→Dropout→Dense→Softmax→Result

Fig.4 shows the architecture of the model. The 32X32 input image is given to the model where the first convolutional layer learns 32 features through a 5x5 filter and 'same' padding. After the activation, another convolutional layer is stacked up that learns 64 features through a 5x5 filter. Then Relu activation is added and forwarded to max pooling layer. After the max pooling layer a dropout is implemented with the dropout of 0.25. This entire layer is repeated again with 3x3 filters and the conv 3 layer this time learns 64 features similar to conv 4 features. All the remaining parameters are same. The conv 5 layer learns 64 features with a 3x3 filter followed by Relu, max-pooling and dropout. Then the output is flattened and we have a fully connected layer with 512 outputs. The dropout is again applied as 0.5 and denser to number of output classes i.e. 10 and passed to the softmax layer for the final output. The accuracy of the model is 80.01 % on a CPU unit on test dataset.

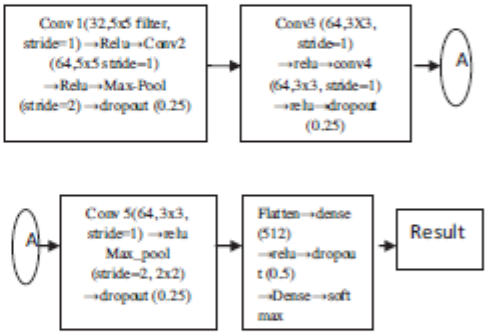

Fig.4 CNN Model for CIFAR-10

E. Relu non-linearity activation function

There is a wide range of activation function available when training neural network models. The mainly used activations are sigmoid, tanh, relu and leaky relu. The relu non linearity is a popular activation function used in deep learning algorithm sand has replaced the use of sigmoidal activation function which is generally used for binary classification techniques. Relu non linearity $f(x) = \max(0,x)$ works several times faster than the tanh non – linearity $(e^z - e^{-z})/(e^z + e^{-z})$, and results the output of the activation as either 0 or a positive number. With relu it is easier to train larger neural networks. Fig.5 shows the graph of relu non-linearity [16].

F. Overfitting and under fitting

Another aspect of training a deep neural network is the issue of high bias (resulting in underfitting) or high variance (resulting in overfitting). When the data is

2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)

underfitting is not generalizing or fitting the data points well. In case of high bias, which is on training batch the network needs to be trained longer and have much bigger network of hidden layers. In case of overfitting, the data has high variance i.e. it is generalizing too well on the test sets. To reduce high variance, regularization techniques and data augmentation techniques can be implemented.

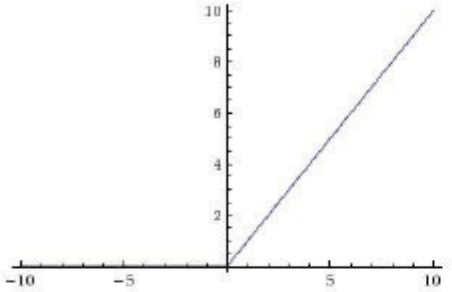

Fig.5. Relu non-linearity

G. Dropout to reduce overfitting

Dropout is a regularization technique which is used to reduce overfitting [7]. In dropout, the network deads some of its nodes randomly based on a parameter. The probability parameter determines whether the node should remain in the network or not. Keep_prob is the probability parameter to keep the hidden nodes in the network. The activation is unaffected during this process as it only determines whether to keep the node in the network or not.

H. Data Augmentation

Another technique to reduce overfitiing is to train the data on large datasets. If dataset is limited the dataset can be artificially created by data augmentation techniques [7]. The data augmentation techniques include distortion and altering of data images for processing to get more data. Some of the techniques are:

- Mirroring – The images are flipped and laterally inverted.
- Random cropping- Cropping some parts of the image and creating subsets from the main image.
- Rotation- This includes rotating the images in any direction at various angles and generating new images.
- Color shifting- Shifting the RGB pixel values of the image to get a new coloured image.

I. RMS prop optimizer and learning rate

RMS prop or root mean square prop is an optimizer which works on the root mean square value of the change in gradients. The change in weights and bias determine the gradient parameters with help of rms value. The learning rate determines the steps the algorithm will take to converge to the global minimum. If the learning rate is too high the algorithm faces exploding gradient problem i.e. it takes larger steps and fails to converge at the local minimum. If the learning rate is too small it faces a vanishing gradient problem i.e. the gradient takes smaller steps and ultimately becomes so small that the changes in weights are insignificant. Thus the learning rate is a hyper parameter that needs to be finely tuned and can be done with the help of learning rate decay. In learning rate decay, the learning rate decays exponentially after every epoch.

IV. RESULT ANALYSIS

The results of the experiments are as shown below:

- CNN Model for MNIST dataset: Accuracy 99.6% shown in figure 6

```
7 0.98828125
8 0.9921875
9 0.99609375

Process finished with exit code 0
```

Fig.6 Accuracy of CNN model on MNIST in 10 epochs

The result shows that as the number of epochs get increased and the best accuracy achieve in recognizing the digits on MNIST data set is 99.6 % with 10 epochs.

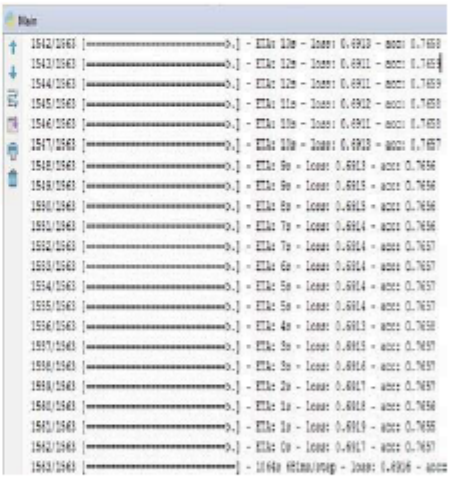

Fig. 7 Accuracy of the CNN model in 50 epochs

CNN model for CIFAR-10 dataset: Accuracy of 80.17% on test set as shown in figure 7.

V. CONCLUSION

The article discusses various aspects of deep learning, CNN in particular and performs image recognition and detection on MNIST and CIFAR -10 datasets using CPU unit only. The accuracy of MNIST is good but the accuracy of CIFAR-10 can be improved by training with larger epochs and on a GPU unit. The calculated accuracy on MNIST is 99.6% and on CIFAR-10 is 80.17%. The

training accuracy on CIFAR-10 is 76.57 percent after 50 epochs. The accuracy on training set may also be improved further by adding more hidden layers. And this system can be implemented as a assistance system for machine vision for detecting nature language sysmbols.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, Geoffery Hinton, "Deep Learning", Nature, Volume 521, pp. 436-444, Macmillan Publishers, May 2015.

[2] Nothidayu binti Abdul Hamid, Nilam Nur Binti Amir Sjarif, "Handwritten Recognition Using SVM, KNN and Neural Network", www.arxiv.org/ftp/arxiv/papers/1702/1702.00723

[3] Cheng-Lin Liu*, Kazuki Nakashima, Hiroshi Sako, Hiromichi Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques", ELSEVIER, Pattern Recognition 36 (2003) 2271 – 2285).

[4] Ping kuang, Wei-na cao and Qiao wu, "Preview on Structures and Algorithms of Deep Learning", 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP), IEEE, 2014.

[5] Mahmoud M. Abu Ghosh ; Ashraf Y. Maghari, "A Comparative Study on Handwriting Digit Recognition Using Neural Networks", IEEE, 2017.

[6] Youssouf Chherawala, Partha Pratim Roy and Mohamed Cheriet, "Feature Set Evaluation for Offline Handwriting Recognition Systems: Application to the Recurrent Neural Network," IEEE Transactions on Cybernetics, VOL. 46, NO. 12, DECEMBER 2016.

[7] Alex Krizhevsky, "Convolutional Deep belief Networks on CIFAR-10". Available: https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf.

[8] Yehya Abouelnaga , Ola S. Ali , Hager Rady , Mohamed Moustafa, " CIFAR-10: KNN-based ensemble of classifiers", IEEE, March 2017.

[9] Caifeng Shan, Shaogang Gong, Peter W. McOwan, "Facial expression recognition based on Local binary patterns: A comprehensive study", ELSEVIER, Image and Vision Computing 27, pp. 803-816, 2009.

[10] Li Deng, "A tutorial survey of architectures, algorithms, and applications of Deep Learning", APSIPA Transactions on Signal and Information Processing (SIP), Volume 3, 2014.

[11] Yann LeCun, Corinna Cortes and Christopher J.C. Burges, "The MNIST Database of handwritten digits". Available: http://yann.lecun.com/exdb/mnist/ - MNIST database

[12] The CIFAR-10 dataset. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[13] MNIST dataset introduction, 2017. Available: http://corochann.com/mnist-dataset-introduction-1138.html

[14] Robust Vision Benchmark. Available: https://rebust.vision/benchmark/about/

[15] Neural Network and Deep Learning. Available: http://neuralnetworksanddeeplearning.com/chap6.html

[16] Convolutional Neural Networks for Visual Recoginition. Available: http://cs231n.github.io/neural-networks-1/

[17] Krizhevsky, Sutskever and Hinton, "ImageNet classification with deep convolutional neural networks", Advances in Neural Information Processing Systems 25 (NIPS 2012), pp. 1106–1114, 2012.

[18] Zeiler, M. D. and Fergus, "Visualizing and understanding convolutional networks". European Conference on Computer Vision, vol 8689. Springer, Cham, pp. 818-833, 2014.

[19] Zhengwei Huang, Min Dong, qirong Mao and Yongzhao Zhan, "Speech Recognition using CNN", IEEE/ACM Transactions on Audio, Speech and Language Processing, pp. 1533-1545, Volume 22, Issue 10, 2014, http://dx.doi.org/10.1145/2647868.2654984.

[20] Shima Alizadeh and Azarr Fazel, "Convolutional Neuml networks for Facial Expression recognition", Computer Vision and Pattern Recognition, Cornell University Library, ArXiv:1704.06756v1, 22 April, 2017, arXiv.org.1704.06756.pdf.

[21] Christian szegedy, Wei liu, Yangqing Jia et al., "Going Deeper with Convolutions",Conference on Computer Vision and Pattern Recognition (CPVR) , IEEE explorer, Boston,MA, USA, 2015.

Appendix B

CODE

1. GUI

```
from tkinter import *     #importing tkinter
import os                 #to access the system directories
import subprocess
from datetime import datetime;
#creating instance of TK
root=Tk()
root.configure(background="#DFDFDF")
def function1(): #to create gestures
   os.system("py create_gestures.py")
def function2(): #to set histogram
   os.system("py set_hand_hist.py")
def function3(): #to start recognising sign language to text
   os.system("py recognize_gesture.py")
def function4(): #exit function
   root.destroy()
root.title("Sign To Text Converter") #to set window title
#creating a text label
Label(root,text="Sign->Text Converter UI",font=("times new roman",20),fg="white",bg="#e51a4c",
height=2).grid(row=0,rowspan=2,columnspan=2,sticky= N+E+W+S,padx=5,pady=5)
#creating first button
Button(root,text="Create Dataset",font=("times new roman",20),bg="#ffffff",fg='black', command=
```

```
function1) .grid(row=3,columnspan=2,sticky=W+E+N+S,padx=5,pady=5)

#creating second button

Button(root,text="Set Hand Histogram",font=('times new roman',20),bg="#ffffff",fg="#000000",
command=function2).grid(row=6,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)


#creating third button

Button(root,text="Recognize Gesture as a Character",font=('times new roman',20),bg="#ffffff",
fg="black",command=function3).grid(row=8,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

#creating  fourth button

Button(root,text="Exit",font=('times new roman',20),bg="#e51a4c",fg="white",command=function4

) .grid(row=11,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

root.mainloop()
```

2. CREATE GESTURES

```
import cv2
import numpy as np
import pickle, os, sqlite3, random

image_x, image_y = 50, 50

def get_hand_hist():
        with open("hist", "rb") as f:
                hist = pickle.load(f)
        return hist

def init_create_folder_database():
        # create the folder and database if not exist
        if not os.path.exists("gestures"):
                os.mkdir("gestures")
        if not os.path.exists("gesture_db.db"):
                conn = sqlite3.connect("gesture_db.db")
                create_table_cmd = "CREATE TABLE gesture ( g_id INTEGER NOT NULL PRIMARY
KEY AUTOINCREMENT UNIQUE, g_name TEXT NOT NULL )"
                conn.execute(create_table_cmd)
                conn.commit()

def create_folder(folder_name):
        if not os.path.exists(folder_name):
                os.mkdir(folder_name)

def store_in_db(g_id, g_name):
        conn = sqlite3.connect("gesture_db.db")
        cmd = "INSERT INTO gesture (g_id, g_name) VALUES (%s, \'%s\')" % (g_id, g_name)
        try:
                conn.execute(cmd)
        except sqlite3.IntegrityError:
```

```
                choice = input("g_id already exists. Want to change the record? (y/n): ")
                if choice.lower() == 'y':
                        cmd = "UPDATE gesture SET g_name = \'%s\' WHERE g_id = %s" % (g_name,
g_id)
                else:
                        print("Doing nothing...")
                        return
        conn.commit()

def store_images(g_id):
        total_pics = 500
        hist = get_hand_hist()
        cam = cv2.VideoCapture(0)
        if cam.read()[0]==False:
                cam = cv2.VideoCapture(0)
        x, y, w, h = 300, 100, 300, 300

        create_folder("gestures/"+str(g_id))
        pic_no = 0
        flag_start_capturing = False
        frames = 0
        while True:
                img = cam.read()[1]
                img = cv2.flip(img, 1)
                imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
                dst = cv2.calcBackProject([imgHSV], [0, 1], hist, [0, 180, 0, 256], 1)
                disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(10,10))
                cv2.filter2D(dst,-1,disc,dst)
                blur = cv2.GaussianBlur(dst, (11,11), 0)
                blur = cv2.medianBlur(blur, 15)
                thresh = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
                thresh = cv2.merge((thresh,thresh,thresh))
                thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
                thresh = thresh[y:y+h, x:x+w]
contours=cv2.findContours(thresh.copy(),cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)[0]
```

```
            if len(contours) > 0:
                contour = max(contours, key = cv2.contourArea)
                if cv2.contourArea(contour) > 10000 and frames > 50:
                    x1, y1, w1, h1 = cv2.boundingRect(contour)
                    pic_no += 1
                    save_img = thresh[y1:y1+h1, x1:x1+w1]
                    if w1 > h1:
                        save_img = cv2.copyMakeBorder(save_img, int((w1-h1)/2) , int((w1-h1)/2) , 0, 0, cv2.BORDER_CONSTANT, (0, 0, 0))
                    elif h1 > w1:
                        save_img = cv2.copyMakeBorder(save_img, 0, 0, int((h1-w1)/2) , int((h1-w1)/2) , cv2.BORDER_CONSTANT, (0, 0, 0))
                    save_img = cv2.resize(save_img, (image_x, image_y))
                    rand = random.randint(0, 10)
                    if rand % 2 == 0:
                        save_img = cv2.flip(save_img, 1)
                    cv2.putText(img, "Capturing...", (30, 60), cv2.FONT_HERSHEY_TRIPLEX, 2, (127, 255, 255))
                    cv2.imwrite("gestures/"+str(g_id)+"/"+str(pic_no)+".jpg", save_img)

            cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
            cv2.putText(img, str(pic_no), (30, 400), cv2.FONT_HERSHEY_TRIPLEX, 1.5, (127, 127, 255))
            cv2.imshow("Capturing gesture", img)
            cv2.imshow("thresh", thresh)
            keypress = cv2.waitKey(1)
            if keypress == ord('c'):
                if flag_start_capturing == False:
                    flag_start_capturing = True
                else:
                    flag_start_capturing = False
                    frames = 0
            if flag_start_capturing == True:
                frames += 1
            if pic_no == total_pics:
```

```
                    break

init_create_folder_database()
g_id = input("Enter gesture no.: ")
g_name = input("Enter gesture name/text: ")
store_in_db(g_id, g_name)
store_images(g_id)
```

3. SET HAND HISTOGRAM

```
import cv2
import numpy as np
import pickle


def build_squares(img):
        p,q,r,s = 420, 140, 10, 10
        d = 10
        imgCrop = None
        crop = None
        for a in range(10):
                for b in range(5):
                        if np.any(imgCrop == None):
                                imgCrop = img[q:q+s, p:p+r]
                        else:
                                imgCrop = np.hstack((imgCrop, img[q:q+s, p:p+r]))
                        #print(imgCrop.shape)
                        cv2.rectangle(img, (p,q), (p+r, q+s), (0,255,0), 1)
                        p+=r+d
```

```
            if np.any(crop == None):
                crop = imgCrop
            else:
                crop = np.vstack((crop, imgCrop))
            imgCrop = None
            p = 420
            q+=s+d
        return crop


def get_hand_hist():
        cam = cv2.VideoCapture(1)
        if cam.read()[0]==False:
            cam = cv2.VideoCapture(0)
        p,q,r,s = 300, 100, 300, 300
        flagPressedC, flagPressedS = False, False
        imgCrop = None
        while True:
            img = cam.read()[1]
            img = cv2.flip(img, 1)
            img = cv2.resize(img, (640, 480))
            hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
            keypress = cv2.waitKey(1)
            if keypress == ord('c'):
                hsvCrop = cv2.cvtColor(imgCrop, cv2.COLOR_BGR2HSV)
```

```
                    flagPressedC = True
                    hist = cv2.calcHist([hsvCrop], [0, 1], None, [180, 256], [0, 180, 0, 256])
                    cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX)
                elif keypress == ord('s'):
                    flagPressedS = True
                    break
                if flagPressedC:
                    dst = cv2.calcBackProject([hsv], [0, 1], hist, [0, 180, 0, 256], 1)
                    dst1 = dst.copy()
                    disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(10,10))
                    cv2.filter2D(dst,-1,disc,dst)
                    blur = cv2.GaussianBlur(dst, (11,11), 0)
                    blur = cv2.medianBlur(blur, 15)
                    ret,thresh =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
                    thresh = cv2.merge((thresh,thresh,thresh))
                    #cv2.imshow("res", res)
                    cv2.imshow("Thresh", thresh)
                if not flagPressedS:
                    imgCrop = build_squares(img)
                #cv2.rectangle(img, (p,q), (p+r, q+s), (0,255,0), 2)
                cv2.imshow("Set hand histogram", img)
        cam.release()
        cv2.destroyAllWindows()
        with open("hist", "wb") as f:
```

```
            pickle.dump(hist, f)
get_hand_hist()
```

4. RECOGNIZE GESTURES

```
import cv2
import imutils
import numpy as np
# global variables
bg = None
global camera
def run_avg(image, aWeight):
    global bg
    # initialize the background
    if bg is None:
        bg = image.copy().astype("float")
        return
# compute weighted average, accumulate it and update the background
cv2.accumulateWeighted(image, bg, aWeight)
def segment(image, threshold=25):
global bg
# find the absolute difference between background and current frame
diff = cv2.absdiff(bg.astype("uint8"), image)
# threshold the diff image so that we get the foreground
thresholded = cv2.threshold(diff,threshold,cv2.THRESH_BINARY)[1]
    # get the contours in the thresholded image
(cnts,_)=cv2.findContours(thresholded.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX
_SIMPLE)
    # return None, if no contours detected
    if len(cnts) == 0:
        return
    else:
        # based on contour area, get the maximum contour which is the hand
```

```
    segmented = max(cnts, key=cv2.contourArea)
    return (thresholded, segmented)
import imutils
def main():
# initialize weight for running average
Weight = 0.5

# get the reference to the webcam
camera = cv2.VideoCapture(0)

# region of interest (ROI) coordinates
top, right, bottom, left = 10, 350, 225, 590

# initialize num of frames
num_frames = 0
image_num = 0
val = 0

start_recording = False

# keep looping, until interrupted
while(True):
    # get the current frame
    (grabbed, frame) = camera.read()
    if (grabbed == True):

        # resize the frame
        frame = imutils.resize(frame, width=700)

        # flip the frame so that it is not the mirror view
        frame = cv2.flip(frame, 1)

        # clone the frame
        clone = frame.copy()
```

```
# get the height and width of the frame
(height, width) = frame.shape[:2]

# get the ROI
roi = frame[top:bottom, right:left]

# convert the roi to grayscale and blur it
gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
rgb = cv2.cvtColor(roi, cv2.COLOR_BGR2RGB)
# to get the background, keep looking till a threshold is reached
# so that our running average model gets calibrated
if num_frames < 30:
    run_avg(gray, aWeight)

else:
    # segment the hand region
    hand = segment(gray)

    # check whether hand region is segmented
    if hand is not None:
        # if yes, unpack the thresholded image and
        # segmented region
        (thresholded, segmented) = hand

        # draw the segmented region and display the frame
        if start_recording and num_frames%5==0:
            frame_topredict = cv2.resize(frame, (32, 32)).reshape(1, 32, 32, 3)
            preds = m.predict(frame_topredict)
            val = np.argmax(preds[0])
            print(val)

cv2.putText(clone, class_names[val], (right, bottom+15), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)
    # Mention the directory in which you wanna store the images followed by the image name
```

```
            # draw the segmented hand
            cv2.rectangle(clone, (left, top), (right, bottom), (0, 255, 0), 2)

            # increment the number of frames
            num_frames += 1

            # display the frame with segmented hand
            cv2.imshow("Video Feed", clone)

            # observe the keypress by the user
            keypress = cv2.waitKey(1) & 0xFF

            # if the user pressed "q", then stop looping
            if keypress == ord("q"):
                break

            if keypress == ord("s"):
                start_recording = True
                num_frames = 0
        else:
            print("[Warning!] Error input, Please check your(camera Or video)")
            break
    camera.release()
main()
 cv2.destroyAllWindows()
import cv2
videoCaptureObject = cv2.VideoCapture(0)
result = True
while(result):
    ret,frame = videoCaptureObject.read()
    cv2.imwrite("NewPicture.jpg",frame)
    result = False
videoCaptureObject.release()
cv2.destroyAllWindows()
get_ipython().run_line_magic('pwd', '')
```

5. TRAINING THE DATASET USING TENSORFLOW OBJECT DETECTION

```
import tensorflow.keras.models as Models

import tensorflow.keras.layers as Layers

import tensorflow.keras.activations as Actications

import tensorflow.keras.models as Models

import tensorflow.keras.optimizers as Optimizer

import tensorflow.keras.metrics as Metrics

import tensorflow.keras.utils as Utils

from keras.utils.vis_utils import model_to_dot

from keras.models import Sequential

model = Models.Sequential()


model.add(Layers.Conv2D(200,kernel_size=(3,3),activation='relu',input_shape=(32,32,3)))

model.add(Layers.Conv2D(100,kernel_size=(3,3),activation='relu'))

model.add(Layers.MaxPool2D())

model.add(Layers.Conv2D(50,kernel_size=(3,3),activation='relu'))

model.add(Layers.MaxPool2D())

model.add(Layers.Flatten())

model.add(Layers.Dense(180,activation='relu'))

model.add(Layers.Dense(29,activation='softmax'))


model.compile(optimizer=Optimizer.Adam(learning_rate=3e-4),loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
model.summary()
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=20,
)

model.save("1.h5")

m = tf.keras.models.load_model('1.h5')

m.summary()
```

6. DETERMINING THE F-SCORES, RECALL AND PRECISION OF TRAINED MODEL

```
scores = m.evaluate(test_ds)
scores
```

Appendix C

Mapping the Project Objectives with POs and PSOs

Course Outcome

Sl No.	Description	Bloom's Taxonomy Level
CSD 334.1	Analyze the current topic of professional interest knowledge (level 1) and present it before an audience.	Knowledge (Level 1) Analyze (Level 4)
CSD 334.2	Identify an engineering problem, analyze it and propose a work plan to solve it.	Evaluate (Level 5) Understand (Level 2)

CO-PO Mapping

	P01	P02	P03	P04	P05	P06	P07	P08	P09	P010	P011	P012
CSD 334.1	-	3	-	`3	-	-	2	-	3	2	-	2
CSD 334.2	3	2	3	2	2	-	-	2	3	2	-	2

CO-PSO Mapping

	PS02	PS02	PS03
CSD 334.1	3	-	1
CSD 334.2	3	3	2

Justifications for CO-PO/PSO Mapping

Mapping	Low/Medium/high	Justification
CSD 334.1-P02	H	Problem analysis : I was able to identify, formulate, review, research literature and analyze problems with the traditional learning environment, reaching substantiated conclusions using first principles of mathematics and natural sciences and engineering sciences.
CSD 334.1-P04	H	Conduct investigations of complex problems : I used research based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.

CSD 334.1-P07	M	Environment and sustainability: I understood the impact of the professional Engineering solutions and societal and environmental contexts and demonstrated the knowledge of the need for sustainable development.
CSD 334.1-P09	H	Individual: I was able to function effectively as an individual in multi-disciplinary settings.
CSD 334.1-P10	M	Communication: I was able to communicate effectively on complex engineering activities with the engineering community and society at large, such as, being able to write effective reports and being able to comprehend and design documentation, make effective presentations, give and receive clear instructions.
CSD 334.1-P11	L	Project management and finance: Demonstrated knowledge and understanding of the Engineering and Management principles and applied these 217 work to manage projects and in multi-disciplinary environments.
CSD 334.1-P12	H	Lifelong learning: Recognise the need for and have the preparation and ability to engage in Independent and lifelong learning in the broader context of technological change.
CSD 334.1-PS01	H	Computer science specific skills: Was able to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science.
CSD 334.1-PS03	L	Professional skills: Was able to apply the fundamentals of computer science to formulate competitive research proposals and to develop innovative products to meet societal needs thereby evolving as an eminent researcher and entrepreneur.
CSD 334.2-P01	H	Engineering knowledge: Applied the knowledge of mathematics, science, engineering fundamentals and engineering discipline to the

		solution of complex engineering problems.
CSD 334.2-P02	M	Problem analysis : I was able to identify, formulate, review, research literature and analyze problems with the traditional learning environment, reaching substantiated conclusions using first principles of mathematics and natural sciences and engineering sciences.
CSD 334.2-P03	H	Design/Development of solutions: Designed solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
CSD 334.2-P04	M	Conduct investigations of complex problems : I used research based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
CSD 334.2-P05	M	Modern tool usage: Created, selected and applied appropriate techniques, resources and modern engineering and IT tools including production and modeling to Complex engineering activities with an understanding of the limitations.
CSD 334.2-P08	M	Ethics: Applied ethical principles and committed to professional ethics and responsibilities and norms of the engineering practice.
CSD 334.2-P09	H	Individual: I was able to function effectively as an individual in multi-disciplinary settings.
CSD 334.2-P010	M	Communication: I was able to communicate effectively on complex engineering activities with the engineering community and society at large, such as, being able to write effective reports and being able to comprehend and design documentation, make effective presentations, give and receive clear instructions.

CSD 334.2-P012	M	Lifelong learning: Recognise the need for and have the preparation and ability to engage in Independent and lifelong learning in the broader context of technological change.

CSD 334.2-PS01	H	Computer science specific skills: Was able to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science.
CSD 334.2-PS02	H	Programming and software development skills: Acquired programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products.
CSD 334.2-PS03	M	Professional skills: Was able to apply the fundamentals of computer science to formulate competitive research proposals and to develop innovative products to meet societal needs thereby evolving as an eminent researcher and entrepreneur.