SIGN LANGUAGE DETECTION

A PROJECT REPORT

Submitted by

Sadhavi Thapa [Reg No: RA1911027010050]
Kanishka Gaur [Reg No: RA1911027010027]

Under the Guidance of

Dr. M.Sangeetha
(Assistant Professor, Department of Data Science and Business Systems)

In partial fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY
COMPUTER SCIENCE ENGINEERING
with specialization in Big Data Analytics

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603203

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

Annexure II

Department of Data Science and Business Systems

SRM Institute of Science & Technology

Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course: B.Tech Computer Science Engineering

Student Name: Sadhavi Thapa, Kanishka Gaur

Registration Number: RA1911027010050, RA1911027010027

Title of Work: Sign Language Detection

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

BONAFIDE CERTIFICATE

Certified that this project report titled "**SIGN LANGUAGE DETECTION"** is the bonafide work of **"Sadhavi Thapa [Reg No: RA1911027010050] and Kanishka Gaur [Reg No: RA1911027010027]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. M.Sangeetha
GUIDE
Assistant Professor
Dept. of DSBS

Dr. M.Lakshmi
HEAD OF THE DEPARTMENT
Dept. of DSBS

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

Sign language is a lingua among the speech and hearing-impaired community. It is hard for most people unfamiliar with sign language to communicate without an interpreter. Sign language recognition pertains to tracking and recognizing the meaningful emotion of humans made with the head, arms, hands, fingers, etc. The technique that has been implemented here, transcribes the gestures from sign language to a spoken language which the listener easily understands. The gestures that have been translated include alphabets and words from static images. This becomes more important for people who rely entirely on gestural sign language to communicate with someone who does not understand sign language. Most of the systems that are under this use face a recognition problem with the skin tone, hence introducing a filter will identify the symbols irrespective of the skin tone.

The aim is to represent features that will be comprehended by the system known as convolutional neural networks (CNN), which contains four types of layers:
convolution layers, pooling/subsampling layers, nonlinear layers, and fully connected layers.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V.Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department**, Dr. M. Lakshmi** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinator **Dr.G.Vadivu,** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her input during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor**, Dr.T.Karthick,** Assistant Professor, Department of Data Science and Business Studies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. M.Sangeetha,** Assistant Professor, Department of Data Science and Business Studies, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under his/her/their mentorship. He/She/They provided me with the freedom and support to explore the research topics of my interest. Her/His/Their passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Data Science and Business Systems staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Sadhavi Thapa

Kanishka Gaur

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN Convolutional Neural Network

ANN Artificial Neural Network

D&M Deaf and Dumb

GUI Graphical User Interface

CHAPTER 1

INTRODUCTION

1.1 General

American sign language is a predominant sign language, Since the only disability Deaf and Dumb (hereby referred to as D&M) people have is communication-related and since they cannot use spoken languages, the only way for them to communicate is through sign language. Communication is the process of the exchange of thoughts and messages in various ways such as speech, signals, behavior, and visuals. D&M people make use of their hands to express different gestures to express their ideas to other people. Gestures are non-verbally exchanged messages and these gestures are understood with vision. This nonverbal communication between deaf and dumb people is called sign language. Sign language is a language that uses gestures instead of sound to convey meaning by combining hand shapes, orientation, movement of the hands, arms, or body, facial expressions, and lip patterns. Contrary to popular belief, sign language is not international. These vary from region to region.

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

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

Figure 1

Minimizing the verbal exchange gap between D&M and non-D&M people turns into an obligation to make certain effective conversations among all. Sign language translation is among the growing lines of research and it enables the maximum natural manner of communication for those with hearing impairments. According to the recent developments in the area of deep learning, neural networks may have profound implications and implementations for sign language analysis. In the proposed system, Convolutional Neural Network (CNN) is used to classify images of sign language because convolutional networks are faster in feature extraction and classification of images over other classifiers. The environment may also recognize a sign as a compression technique for information transmission, which is then reconstructed by the receiver. The signs are divided into two categories: static and dynamic signs. The movement of body parts is frequently included in dynamic signs. Depending on the meaning of the gesture, it may also include emotions. A hand gesture recognition system offers an opportunity for deaf people to talk with vocal humans without the need for an interpreter. The system is built for the automated conversion of ASL into textual content and speech.

1.2 Purpose

• Communication has always had a great impact in every domain and how it is considered the essence of thoughts and expressions that captivated the researchers to bridge this gap for every mortal being.

• The objective of this project is to identify the symbolic expression through images so that the communication gap between a normal & a hearing-impaired person can be easily transversed.

1.2.1 Scope

- We can develop a model for ISL word and sentence level recognition. This will require a system that can detect changes with respect to the temporal space.
- We can develop a complete product that will assist speech and hearing-impaired people, and thereby reduce the communication gap.

1.3 Motivation and Problem Statement

- Sign language is learned by the deaf and dumb, and usually it is not known to normal people, so it becomes a challenge to communicate between a normal and hearing-impaired person.
- It strikes our mind to bridge the gap between hearing impaired and normal people to make communication easier.
- Sign language recognition (SLR) system takes an input expression from the hearing-impaired person and gives output to the normal person in the form of text or voice.

Problem Statement

Sign Language is the main means of communication for deaf and dumb people. They use hand gestures/signs to communicate, hence normal people face problems in recognizing their language by signs made. Hence there is a need for systems that recognize the different signs and conveys the information to normal people.

CHAPTER 2

LITERATURE STUDY

In recent years there has been monumental research done on hand gesture recognition.

With the help of a literature survey, we acquired that the basic steps in hand gesture recognition are: -

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

2.1 Data acquisition:

The different approaches to acquire data about hand gestures can be done in the following ways:

1. Use of sensory devices:

   It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user-friendly.

2. Vision-based approach:

   In vision-based methods, the computer webcam is the input device for observing the information of hands and/or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices, thereby reducing cost. These systems tend to complement biological vision by describing artificial vision systems that are implemented in software and/or hardware. The main challenge of vision-based hand detection ranges from coping with the large variability of the human hand's appearance due to a huge number of hand movements, to different skin-color possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

2.2 Data Pre-Processing and 2.3 Feature extraction for vision-based approach:

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

- We can also extract the necessary image which is to be trained by applying a filter called Gaussian Blur (also known as Gaussian smoothing). The filter can be easily applied using open computer vision (also known as OpenCV) and is described in [3].

- For extracting the necessary image which is to be trained we can use instrumented gloves as mentioned in [4]. This helps reduce computation time for Pre-Processing and gives us more concise and accurate data compared to applying filters on data received from video extraction.

- We tried doing the hand segmentation of an image using color segmentation techniques but skin color and tone are highly dependent on the lighting conditions due to which the output we got for the segmentation we tried to do were not 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 the 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 the 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.

2.4 Gesture Classification:

- In [1] Hidden Markov Models (HMM) are 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-lookup indexing table. After filtering, skin color pixels are gathered into blobs. Blobs are statistical objects based on the location (x, y) and the colorimetry (Y, U, V) of the skin color pixels in order to determine homogeneous areas.

- In [2] Naïve Bayes Classifier(**a probabilistic 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 extract geometric invariants from them. The next step is the classification of gestures by using a K nearest neighbor algorithm aided with a distance weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes" classifier.

- According to the paper "Human Hand Gesture Recognition Using a Convolution Neural Network" by Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen (graduates of the Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan), they have constructed a skin model to extract the hands out of an image and then apply a 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 the axis. They input this image into a convolutional neural network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using this model they produced an accuracy of around 95% for those 7 gestures.

CHAPTER 3

PROPOSED METHODOLOGY

3. 1 Framework

3.1.1 Artificial Neural Network (ANN):

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

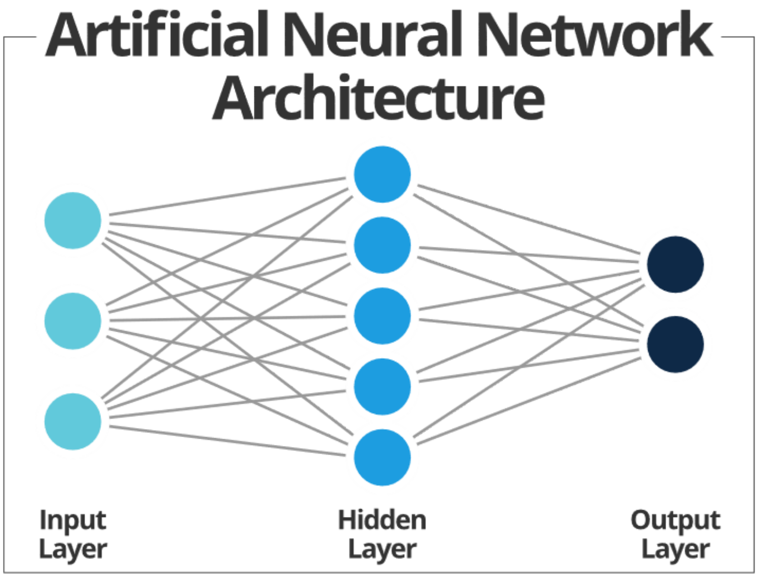


Figure 2

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

1. Unsupervised Learning
2. Supervised Learning
3. Reinforcement Learning

3.1.2 Convolutional Neural Network (CNN):

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

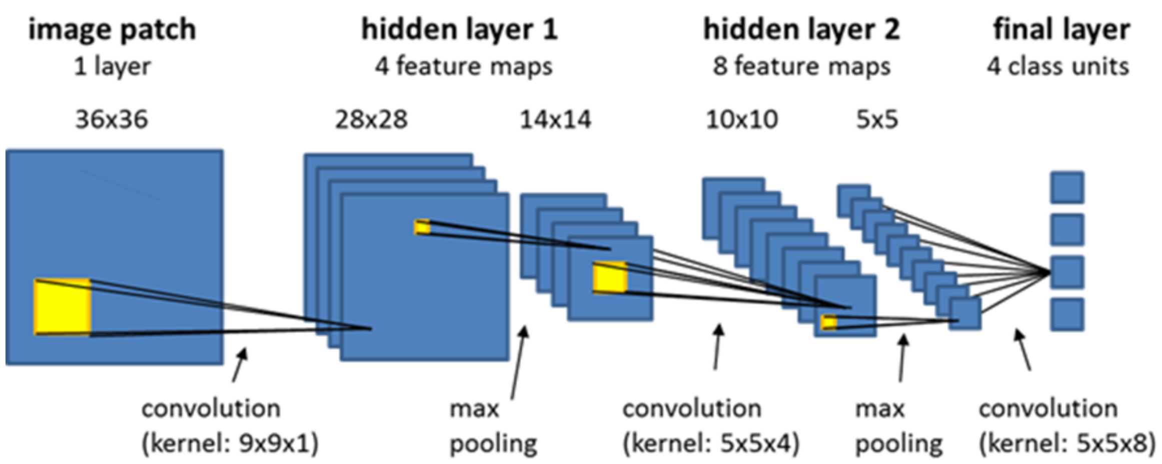


Figure 3

1. Convolution Layer:

In the 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 computed the dot product of filter entries and input values at a given position.

As we continue this process we will create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual features such as an edge of some orientation or a blotch of some color.

2. Pooling Layer:

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

a. Max Pooling: In max pooling, 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 we'll finally get an activation matrix half of its original Size.

b. Average Pooling: In average pooling, we take advantage of all Values in a window.

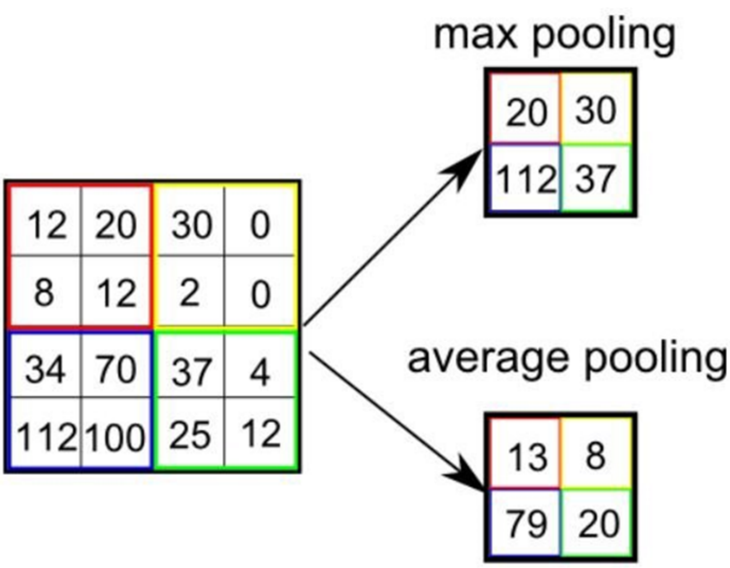


Figure 4

3. Fully Connected Layer:

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

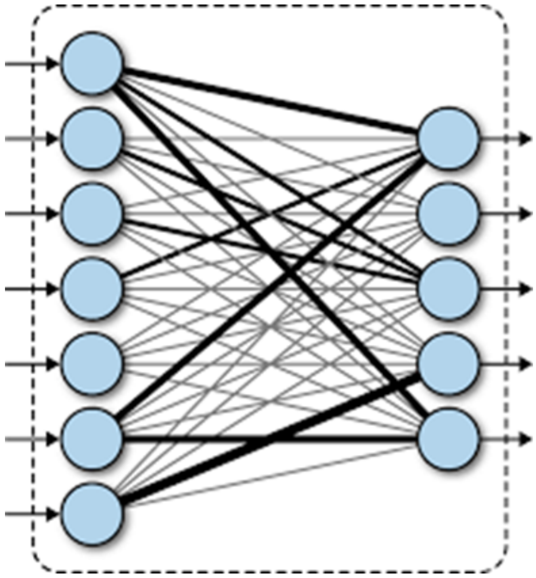


Figure 5

4. Final Output Layer:

After getting values from the fully connected layer, we will connect them to the final layer of neurons [having a count equal to a total number of classes], which will predict the probability of each image being in different classes.

3.1.3 TensorFlow:

TensorFlow is an end-to-end open-source platform for Machine Learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in Machine Learning and developers easily build and deploy Machine Learning powered applications.

TensorFlow offers multiple levels of abstraction to choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for rapid iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

3.1.4 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, objectives, activation functions, optimizers, and tools to make working with images and text data easier.

3.1.5 OpenCV:

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

It is mainly used for image processing, video capture, and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however, bindings are available for Python, Java, MATLAB/OCTAVE.

3.2 Architecture Diagram

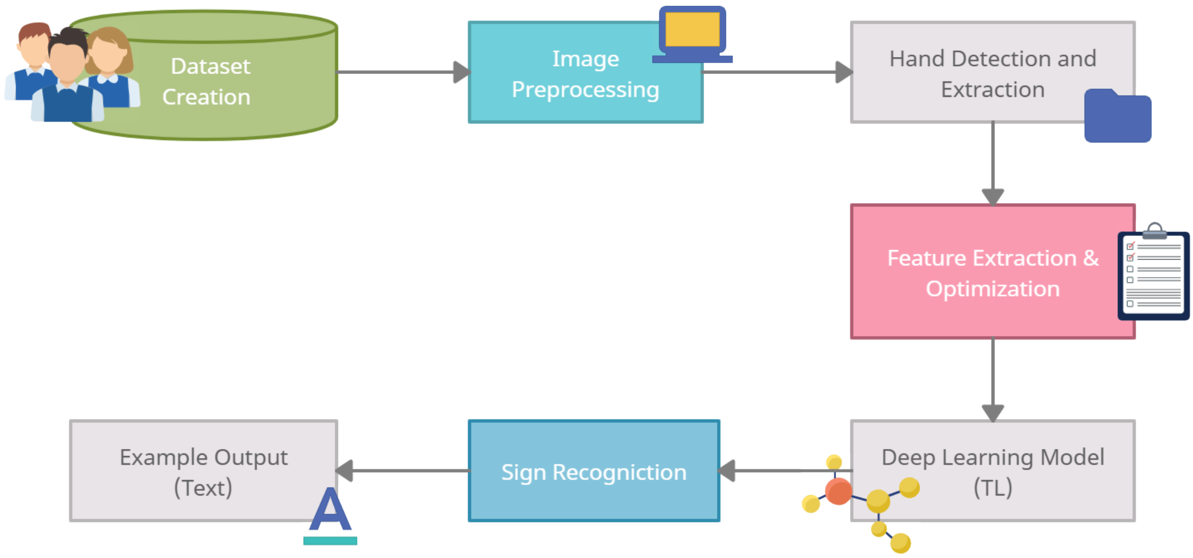


Figure 6

3.3 Activity Diagram

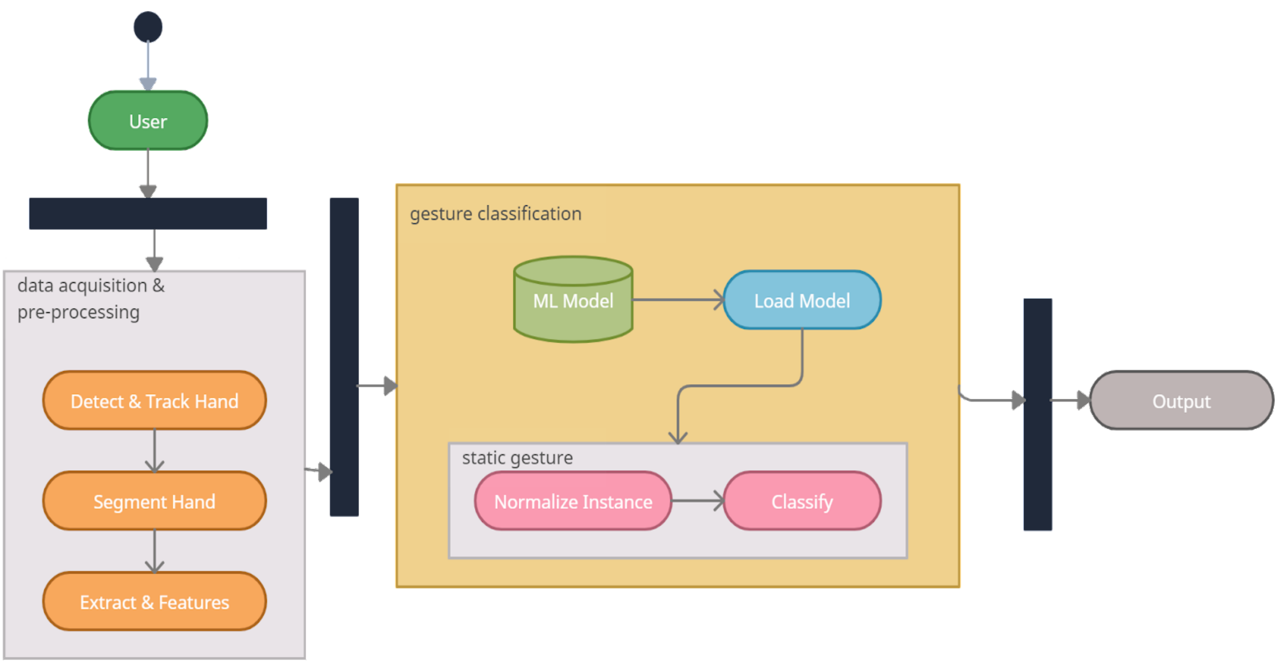


Figure 7

3.4 Methodology

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

3.4.1 Data Set Generation:

No proper dataset was available for our project which could be a proper fit, couldn't find a dataset in the form of raw images that matched our requirements. Hence, we decided to create our own data set. The steps we followed to create our data set are as follows.

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

Firstly, we captured around 800 images of each of the symbols in ASL (American Sign Language) for training purposes and around 200 images per symbol for testing purposes.

First, 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 green bounded square as shown in the image below:

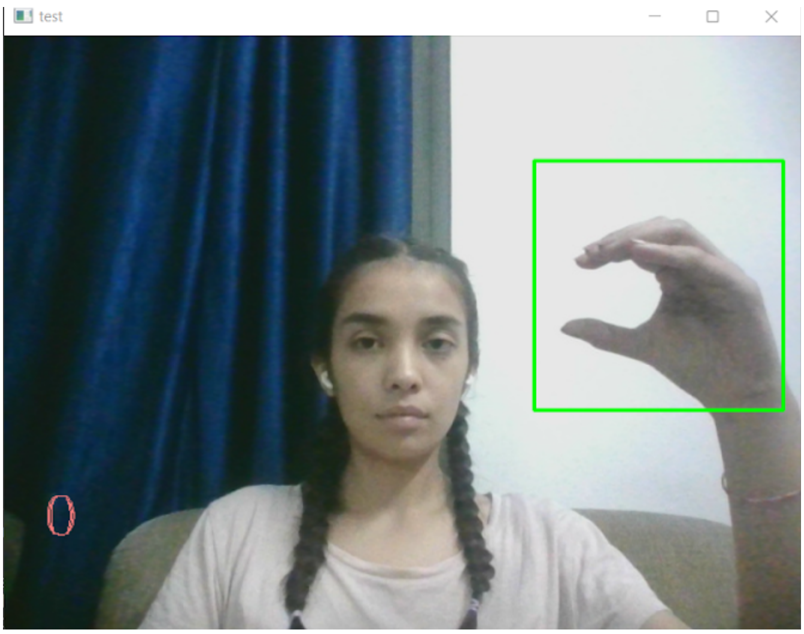


Figure 8

Then, we apply the Gaussian Blur Filter to our image which helps us extract various features of our image. The image, after applying Gaussian Blur, looks as follows:

Figure 9

3.4.2 Gesture Classification:

Our approach uses one layer of the algorithm interconnected (convolution + max-pooling layers) to predict the final symbol of the user.

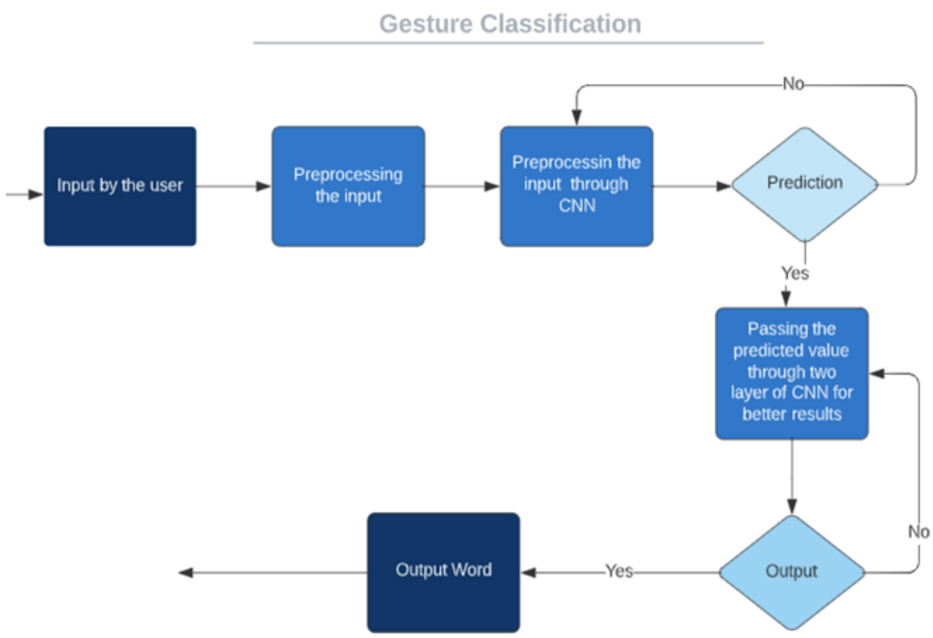


Figure 10

3.4.3 Algorithm Layer:

1. Apply color space conversions, 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.
3. If a letter is detected for more than 30 frames then the letter is printed.

3.4.4 CNN Model:

1. 1st Convolution Layer: The input picture has a resolution of 128x128 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126x126 pixel image, one for each Filter-weights.
2. 1st Pooling Layer: The pictures are downsampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of an 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 serve 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 a max pool of 2x2 and are 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 this layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.

6. 2nd Densely Connected Layer: Now the output from the 1st Densely Connected Layer is used as an input to a fully connected layer with 96 neurons.
7. Final layer: The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

Activation Function:

We have used ReLU (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons).ReLU calculates max(x,0) for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

Pooling Layer:

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

Dropout Layers:

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

Optimizer:

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

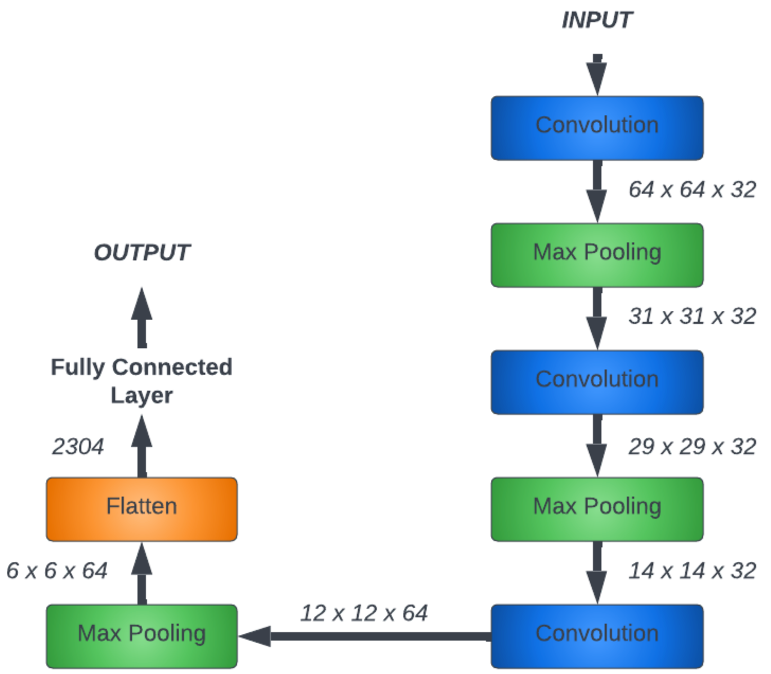


Figure 11

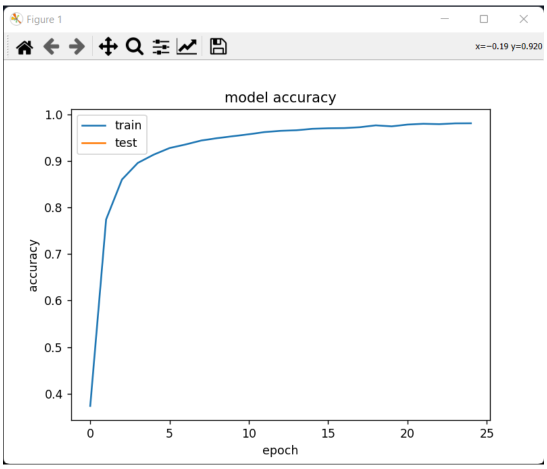
4.4.5 Training and Testing:

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

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

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

- At first, the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labeled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function that is positive at values that are not the same as the 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.



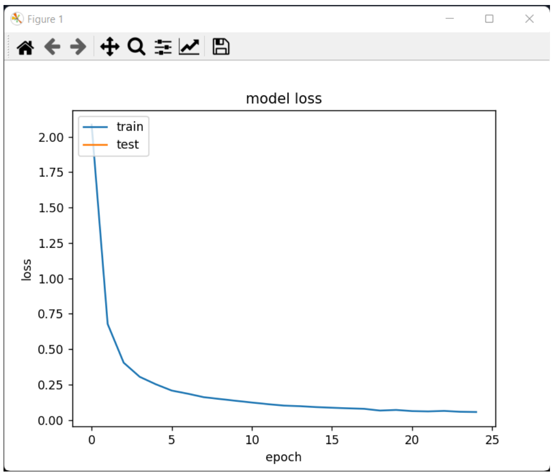


Figure 12

CHAPTER 4

MODULES

Our project was divided into 4 main modules which covered the formulation of data to deploy the CNN model using GUI.

1. Collecting Images for Dataset
2. Formulating CNN Model
3. Gesture Prediction
4. Sign Language to text deployment of CNN model using Tkinter (GUI)

Figure 13

CHAPTER 5

CONCLUSION

The proposed system successfully predicts the signs of signs and some common words under different lighting conditions and different speeds. Accurate masking of the images is being done by giving a range of values that could detect human hands dynamically. The proposed system uses CNN for the training and classification of images. For classification and training, more informative features from the images are finely extracted and used. A total of 1199 static images for each sign are used for training to get the accurate output. Finally, the output of the recognized sign is shown in the form of text as well as converted into speech. The system is capable of recognizing all 26 alphabets out of which 10 are predicted with 98% accuracy. Thus this is a user-friendly system that can be easily accessed by all deaf and dumb people.

CHAPTER 6

FUTURE ENHANCEMENTS

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

We are also thinking of improving the Pre Processing to predict gestures in low-light conditions with higher accuracy.

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

CHAPTER 7

REFERENCES

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

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

[3] Smoothing Images — OpenCV 2.4.13.7 documentation

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

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

[6] Research - Sign Language Recognition (rwth-aachen.de)

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

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

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

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

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

[12] Home - OpenCV

[13] TensorFlow - Wikipedia

[14] Convolutional neural nework - Wikipedia

APPENDIX

1. OpenCV:

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

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

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

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

2. Convolutional Neural Network:

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

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

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

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

3. TensorFlow:

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

It is used for both research and production at Google.

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

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

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

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

PAPER PUBLICATION STATUS

Mention either one of the following:

Publication process not yet started (Research Paper Draft formulated)