Project Report on Sign Language Recognition

by Kuldeep Sharma

Submission date: 16-May-2024 04:39PM (UTC+0530)
Submission ID: 2378186387
File name: Project_Report_on_Sign_Language_Recognition.pdf (2.91M)
Word count: 6776
Character count: 36370

KIET
GROUP OF INSTITUTIONS
Connecting Life with Learning

A Project Report

On

Sign Language Recognition

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2023-24

in

KIET Group of Institution, Ghaziabad

By

Ashish Kumar Sharma (2000290100032)

Divyansh Sheoran (2000290100059)

Ayush Chauhan (2000290100040)

Under the supervision of

Shalini Kapoor

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

December, 2023

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Name: Ashish Kumar Sharma

Roll No.:2000290100032

Date:

Signature:

Name: Divyansh Sheoran

Roll No.:2000290100059

Date:

Signature:

Name: Ayush Chauhan

Roll No.:2000290100040

Date:

Signature:

CERTIFICATE

This is to certify that Project Report entitled "Sign Recognition System" which is submitted by Ashish Kumar Sharma, Divyansh Sheoran and Ayush Chauhan in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

Date:

Supervisor Name: Shalini Kapoor

(Assistant Professor)

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Shalini Kapoor, Department of Computer Science & Engineering, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Vineet Sharma, Head of the Department of Computer Science & Engineering, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

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

Name- Ashish Kumar Sharma Signature
Roll no.- 2000290100032
Date-

Name- Divyansh Sheoran Signature
Roll no.- 2000290100059
Date-

Name- Ayush Chauhan Signature
Roll no.- 2000290100040
Date-

ABSTRACT

The goal of the Sign Language Recognition project is to create a system that can understand and recognize sign language motions by utilizing the powerful JavaScript hand tracking framework Handfree.js. The aim is to enable smooth contact with technology by filling in the gaps in communication for people who have hearing loss. This paper gives a thorough summary of the project, covering its history, goals, methods, specifics of implementation, outcomes, and suggestions for the future.

In order to achieve accurate hand tracking, the project integrates Handfree.js with a highly developed sign language recognition module. A methodical strategy is utilized, involving the gathering of datasets, preprocessing, choosing models, and training procedures. The design and development of the sign language recognition module, the integration of Handfree.js into the system, and the creation of an intuitive user interface are all included in the implementation phase.

The results of the assessment provide the system's performance metrics and accuracy along with a comparison to other sign language recognition systems currently in use. The study explores how the results should be interpreted, pointing out the shortcomings of the system and suggesting areas for improvement in the future.

List of Abbreviations

SLR	Sign Language Recognizer
ASL	Amercian Sign Language
ISL	Indian Sign Language
ML	Machine Learning
DL	Deep Learning
NN	Neural Networks
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
SVM	Support Vector Machine
LBP	Local Binary Patterns
HoG	Histogram of Gradients
PCA	Principal Component Analysis
GUI	Graphical User Interface
SSIM	Structural Similarity Index Measure
FLANN	Fast Library for Approximate Nearest Neighbours
TTS	Text-to-Speech

List of Figures

Figure Number	Name of Figure
Fig. 1.1	User Case Diagram
Fig. 1.2	Activity Diagram
Fig 2.1	SVM
Fig 2.2	k-NN
Fig 2.3	CNN Layers
Fig 2.4	Convolutional Layer
Fig 2.5	Pooling Layer
Fig 2.6	ReLU Layer
Fig 2.7	Fully-Connected Layer
Fig 2.8	Different types of Bias
Fig 2.9	Dropout
Fig 2.10	Different Optimizer Performance
Fig 2.11	Forward Propagation
Fig 2.12	Backward Propagation
Fig 3.1	Image Processing
Fig 3.2	Pre-Processing Module
Fig 3.3	Code Snippet of Model
Fig 3.4	Code Snippet of Model
Fig 3.5	Code Snippet of Custom Gesture Prediction
Fig 4.1	Landing Page
Fig 4.2	Alphabet Prediction (i)
Fig 4.3	Alphabet Prediction (ii)
Fig 4.4	Custom Gesture Prediction

List of Tables

Table Number	Name of Table	Page Number
Table 1.1	The table shows the maximum accuracy recorded for each algorithm	3
Table 1.2	The table shows the average accuracy recorded for each algorithm	3

Acknowledgements	i
Self Declaration	ii
Certificate	iii
Abstract	iv
List of Abbreviations	v
List of Figures	vi
List of Tables	vii

Table of Contents

Chapter 1	**Introduction**		**01-09**
	1.1	Introduction	01
	1.2	Problem Statement	01
	1.3	Scope	02
	1.4	Methodology	02
		1.4.1 Results Based on Past Research	03
	1.5	System Specifications	04
		1.5.1 Non-Functional Requirements	04
		1.5.2 Functional Requirements	04
	1.6	User Stories	05
	1.7	Organization of Project	06
		1.7.1 Modules	06
		1.7.2 User Case Diagram	07
		1.7.3 Activity Diagram	08
	1.8	Summary	09

Chapter 2	**Study and Review of Literature**		**10-22**
	2.1	Introduction	11
	2.2	Different Classification Algorithms	12
		2.2.1 Support Vector Machine	12
		2.2.2 k-Nearest Neighbours	13
		2.2.3 Convolutional Neural Network (CNN)	14
	2.3	Building Blocks of Convolutional Neural Network	15
		2.3.1 Convolutional Layer	15
		2.3.2 Pooling Layer	16
		2.3.3 ReLU Layer	16
		2.3.4 Fully Connected Layer	17
	2.4	Training a Network	17
	2.5	Loss Function	18
		2.5.1 Softmax Function	18
	2.6	Regularization	19
		2.6.1	19
	2.7	Optimisation	20
		2.7.1 ADAM	20
	2.8	Backpropagation Algorithm	21
		2.8.1 Backpropagation Pseudocode	22
	2.9	Summary	22
Chapter 3	**Proposed Model Implementation**		**23-32**
	3.1	Introduction	24
	3.2	Collecting Data and Preprocessing Module	24
	3.3	Model Training	25
	3.4	Custom Gesture Creation and Prediciton Module	26
	3.5	Code Snippets	27
		3.5.1 Preprocessing Module	27
		3.5.2 Module Creation	28
		3.5.3 Custom Gesture Creation	29

		3.5.4 Custom Gesture Prediction	30
	3.6	Frontend Module	31
	3.7	Summary	32
Chapter 4	**Results and Conclusion**		**33-39**
	4.1	Results	34
		4.1.1 Screenshots	35
	4.2	Conclusion and Future Scope	36
	References		**37**
	Appendix		**38**

Chapter 1
Introduction

1.1 Introduction

The deaf and dumb community uses sign language as their primary form of communication. A variety of hand gestures, finger movements, facial, head, and eye movements are combined to form their language. Sign language has a precise grammar of its own. The project's goal is to replace the traditional human-to-human method of communication with a relatively novel one—human-computer communication. In the past, sign language has been disregarded. As a result, our study focuses on deaf-mute individuals whose problems are frequently ignored by society. They're considered to be the "other" variety. They can only be communicated with through sign language.

With the help of this initiative, those with exceptional abilities will be able to communicate not only with those who are considered "normal" by society, but also with those who are blind because the text recognized by the sign will be transformed into speech. Therefore, a method combining computer vision and deep learning with machine learning will be employed to accomplish this goal. In summary, this method takes the key elements of gestures and motions from a video and recognizes the sign in real-time. Numerous sign language systems have been developed, however they have proven to be exceedingly expensive and difficult to use. This project is the foundation for clear documentation and a user-friendly GUI for sign language recognition.

1.2 Problem Statement

- Implement an application that recognizes ASL/ISL in real-time, capturing hand gestures and movements, and displaying the results on the screen. By doing this, the individual with a particular ability—deaf or dumb—would be able to interact with society more readily and bridge the communication gap.

1.3 Scope

Considering that using the current solutions is expensive and complex. Making it easy to use and affordable is the driving force behind our initiative. With this method, the ASL and ISL datasets are used to train the model.

A camera and a PC or mobile phone are among the essential hardware requirements. This method leverages the handsfree.js standard computer vision and machine learning methods, unlike many other systems that use specially built motion detector cameras, such as Microsoft Kinetic and gloves.

The main idea is that the camera will recognize the sign and detect motion before displaying it. By combining the recognized signs, a meaningful signal will be produced.

1.4 Methodology

Many Machine Learning(ML) algorithms like Support Vector Machine(SVM), Convolutional Neural Network(CNN) are used for classification. For this project, handsfree.js is used. Handsfree.js is a JavaScript library designed to enable hands-free interactions on the web using computer vision and machine learning

CNN has shown to be incredibly effective at both processing images and extracting features. A few feature extraction algorithms are used to reduce the dimensionality and extract significant data from the images, such as Principal Component Analysis (PCA), Local Binary Patterns (LBP), and Histogram of Gradients (HoG). These algorithms optimize the model by reducing the amount of memory used. Based on previous research, the results show that a model that combines SVM with HoG or a CNN yields substantially more accurate results (1.4.1). Consequently, a CNN trained on the ASL and ISL datasets to determine an appropriate sign will serve as the project's main component.

1.4.1 Results Based on Past Research:

Table 1.1. The table shows the maximum accuracy recorded for each algorithm

Images /class	Testing set (Subject)	Algorithm	Parameters used	Maximum Accuracy
10	1	SVM	C=10.0 , gamma=0.01	70
10	1	SVM+PCA	No. of components=53	66.67
10	5	SVM+LBP	(No of points to consider for LBP , Radius): (8,2)	11.6
10	5	SVM+LBP with pre-processing	(No of points for LBP , Radius) : (16,2)	31.71
10	3	SVM+HoG	Pixels per cell : (8,8) Cells per block : (1,1)	77.66
100	2	SVM+HoG	Pixels per cell : (8,8) Cells per block : (1,1)	81.15
10	5	k-NN	Nearest neighbours=5	55
10	3	k-NN with HoG	Nearest Neighbours=5	67.63

Table 1.2 The table shows the average accuracy recorded for each algorithm

Images / class	Classifier	Parameters	Average Accuracy (%)
10	SVM+PCA	No. of components =53	60
10	SVM+LBP	(No of points to consider for LBP , Radius): (8,2)	11
10	SVM+LBP with pre-processing	(No of points to consider for LBP , Radius) : (16,2)	31
10	SVM+Hog	Pixels per cell : (8,8) Cells per block : (1,1)	68.93
100	SVM+Hog	Pixels per cell:(8,8) Cells per block: (1,1)	71.78

1.5 System Specification

1.5.1 Non-Functional Requirements

- Hardware Requirements
  - Intel i5 or above / AMD Ryzen
  - 1GB of HDD/SSD
  - 4GB of RAM
  - Recommend - AMD/NVIDIA GPU for better performance
  - Built-in webcam or external 4MP webcam

- Software Requirements
  - Microsoft Windows 10/ Ubuntu 15.0 LTS or later/ MacOS 12.0 or later
  - Vs Studio code
  - Handsfree.js

1.5.2 Application Features / Functional Requirements

- ASL/ISL Sign language recogniser in real-time
- UI designed for ease of use for users
- A dedicated web support for the application
- Detailed documentation so that person could understand the usage.

1.6 User Stories:

1.6.1 Epic 1: Communication Gap

- Being deaf, I find it challenging to interact with people in society who are not familiar with sign language.
- Being a simpleton, I find it uncomfortable to convey my ideas and "speak" incomprehensibly to those who are unaware of my knowledge.
- It's challenging for anyone to communicate with someone who has special abilities because I don't know sign language.

1.6.2 Epic 2: Ease to use

- As a person with special abilities, I want an application to recognise sign language in realtime.
- As a person with special abilities, it would make my work easier when an application can detect and convert my signs into texts.
- As someone not familiar with sign language, I want my text/speech to be converted into sign language so that I can communicate with deaf/dumb people.

1.6.3 Epic 3: Affordable

- As a deaf/dumb person, I want an application to be affordable in terms of usage and also pocket-friendly.
- As a deaf/dumb person, I want to use an application on the hardware I already have, like phone/laptop, so that I don’t have to invest in a special compatible hardware.
- As a person the application should be cheap so that it's easier for the whole society to afford it and destroy the communication barrier.

1.7 Organisation of project:

1.7.1 Modules

- **Data Pre-Processing -**

Images of gestures are processed in this module. Images are gone through several filters to extract feature from it like cropping, thresholding, removing background and other filters to extract features to pass on into the CNN of handsfree.js.

- **Model Training -**

Our model's intelligence is the main focus of this phase. We help it learn and get better by providing it with a ton of hand gesture samples. Our methodology uses latest machine learning techniques to expedite and optimize this learning process, guaranteeing that our model improves its recognition accuracy of a broad spectrum of gestures without requiring an excessive amount of training gym time.

- **CNN -**

The pre-processed image is passed in to the CNN model of handsfree.js to produce an output. This module will input the image and recognise the label. The output will be shown on the Screen.

- **Creating Gestures -**

This module will help to create new gestures for ease of use for users. This will help user to create his/her own gestures for fast and easy communication. This module will capture several images for the new gesture provided and ask the user to label it. This gesture will be saved for further use.

1.7.2 User Case Diagram

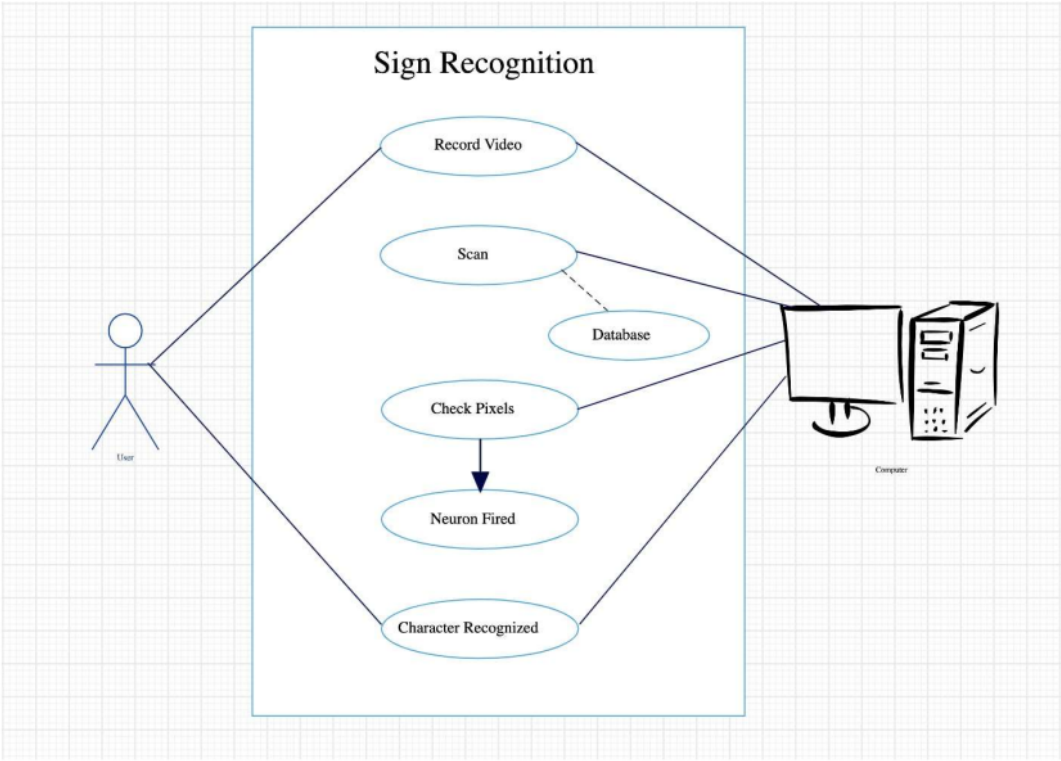

Fig. 1.1 User Case Diagram

1.7.3 Activity Diagram

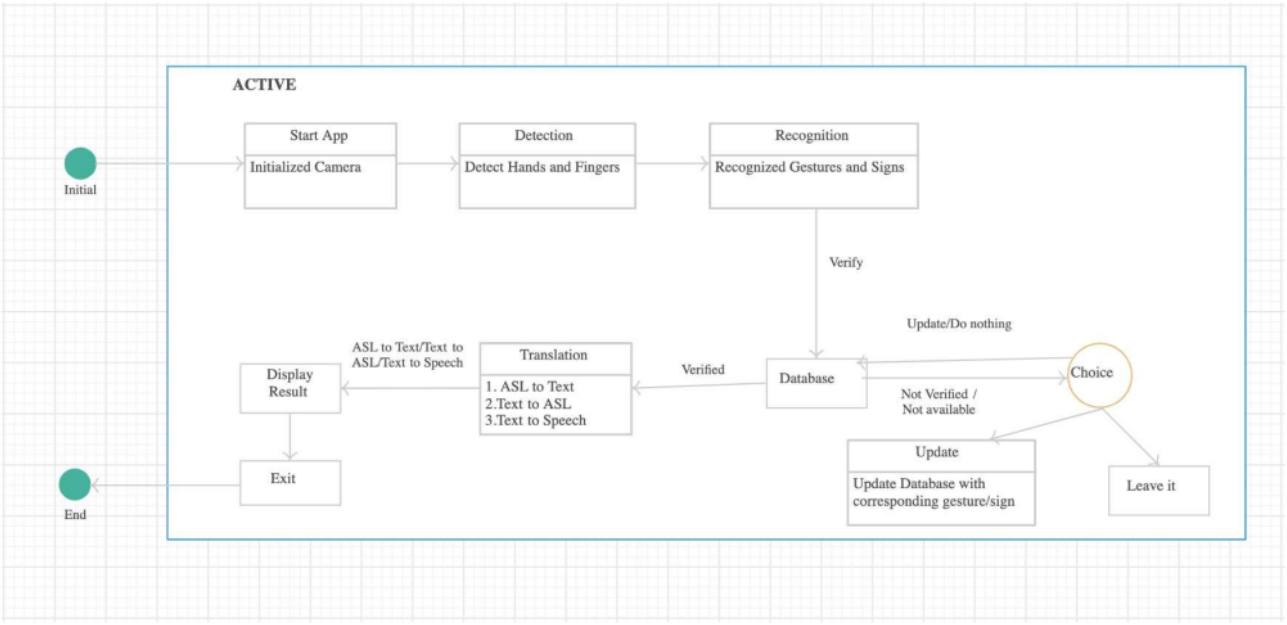

Fig. 1.2 Activity Diagram

1.8 Summary

This venture centers on making a real-time sign dialect acknowledgment framework utilizing handsfree.js, a JavaScript library known for its motion and movement following capabilities without extra equipment prerequisites. By coordination handsfree.js, the extend points to capture and translate hand signals through webcam input, changing over them into comparing sign dialect images or words.

The center components include preparing a machine learning show to precisely recognize these signals, creating a user-friendly interface for consistent interaction, and optimizing execution for smooth real-time handling. The system's objectives include tall precision, responsive execution, availability highlights, adaptability, and ceaseless advancement through client criticism. Potential applications run from improving online communication stages and instructive apparatuses to giving availability arrangements and intelligently excitement encounters.

Eventually, this extend looks for to use innovation to bridge communication holes and cultivate inclusivity for people with hearing disabilities, advertising them a more available implies of communicating themselves and locks in with computerized substance.

Chapter 2
Study and Review of Literature

2.1 Introduction

The science of teaching computers to behave without explicit programming is known as machine learning, or ML. It has been used in a variety of industries during the last ten years, including robotics, data mining, statistical pattern recognition, computer vision, natural language processing (NLP), and medical informatics. A branch of artificial intelligence is machine learning. The majority of machine learning models and techniques that are currently in use are only motivated by our comprehension of human learning strategies. For instance, the idea of numerous neurons being connected is the foundation of the neural network. Tom Mitchell offers a more up-to-date and enlightening description of machine learning. "When it comes to a class of tasks and performance measure, a computer program is said to learn from experience if its performance at tasks, as measured by, improves with experience." For instance, let's take the problem of learning to play checkers. In this case, the opportunity to play against oneself, the percentage of games won, and the actual checkers game would all be present.

A specific ML problem can be classified into two broad categories that are i) Supervised Learning ii) Unsupervised Learning. Supervised learning deals with the “labelled” input data. Whereas, Unsupervised learning learns from neither classified nor labelled input data. Regression and Support Vector Machines are part of supervised learning. Clustering and k-Means techniques describe unsupervised learning.The particular problem of Sign Language Recognition lies under Supervised Learning. In the further sections of the chapter different classifiers and methodologies used by various researchers have been explained and compared to the proposed one.

2.2 Different Classification Algorithms

A Sign Language Recogniser's main aim is to classify the given sign and convert it to a meaningful word or an alphabet. Machine Learning Classification algorithms consists of SVM, k-NN and CNN which used for supervised learning. This section discusses about these algorithms:

2.2.1 Support Vector Machine (SVM):

SVM are mostly chosen for classification problem. Though SVM can also be used for regression. In SVM each data point is plotted over a n-dimensional plane, where n is the number of features. It helps us to find a hyper-plane that creates a boundary between the different types of data points. For a 3-dimensional space, its hyperplane is a 2-dimensional plane. To generalise this, the hyperplane for a n-dimensional space will be a flat subset with $(n - 1)$ dimension and it separates the space into two halves.

If the set of samples can be separated linearly, then the technique to find the hyperplane is called Linear SVM. For non-linear classification, we use Kernelized SVM. A kernel is a measure of similarity between two data points. The kernel function is an inner product between the sample i.e it tells that the similarity between the points in the newly transformed feature space. Two famous kernel functions are a) Radial Basis Function Kernel(RBF), b) Polynomial Kernel. A valid kernel function must satisfy the Mercer's conditions i.e. for a given Kernel Function $k(x, y)$, $k(x, y) = k(y, x)$ must hold true.

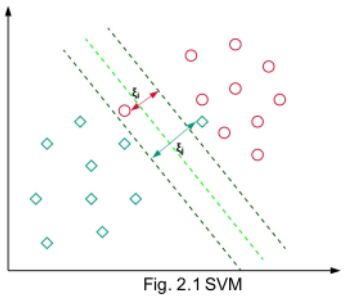

Fig. 2.1 SVM

2.2.2 k-Nearest Neighbours (k-NN)

k-Nearest Neighbours is another classification algorithm. It has an application in pattern recognition, data mining and intrusion detection. In k-NN, a object is classified by a majority vote of its neighbours. k-NN assumes that similar things exist in close proximity. k-NN have its usage in regression as well as classification but it is mainly used for classification. k-NN is a non- parametric algorithm that means it does not make any assumption win underlying data. k-NN is an example of lazy learning, as it does not make any generalisation. Therefore training is required when we use this algorithm.

k-NN are used in simple recommendation systems and image detection. k-NN make predictions by searching through the entire training set for the k most similar "neighbours". To determine the instances in the dataset, "distance" measure is used. Distance can be Euclidean Distance, Hamming Distance, Manhattan Distance and Jaccard Distance. There are many other distance measures, which can be chosen according to the dataset property. Accuracy of the model also depends on the number of the nearest neighbours i.e. the value of k.

The algorithm of k-Nearest Neighbours is as follows:

1. Find distances between the data point to be classified to all the other data points
2. Pick *k* shorter distances
3. Pick the most common class in these *k* distances, that will be the classified class.

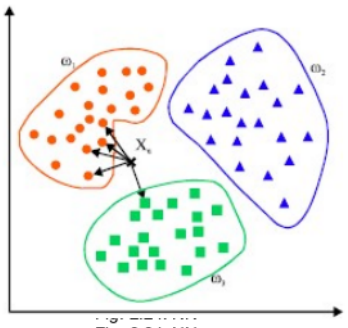

Fig. 2.2 k-NN

2.2.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks, is a deep learning algorithm which are commonly used for analysing visual imagery.These models has proven very successful at image recognition. CNN are inspired by the visual cortex. Neural Networks are based on the idea of interconnecting neurons. CNN are usually used to process the data that have grid like topology, eg. images that can be represented as a 2D array of pixels. A CNN sequence consists of four main operations: a) Convolution , b) Non-Linearity(ReLU) , c) Pooling and Fully-Connected layer.

For an image classifier, convolution layer extracts the features from the input image. It learns image feature using small squares of input data. The following layer is ReLU which replaces the negative pixels by zero. It produces a non-linearity in convolution network. The next layer, which is the pooling layer downsamples each feature map but retains important data. Now, the fully-connected layer uses a softmax function to use features from previous layer to classify the input image based on training.

CNN models can also be pre-trained on the data different from the original. This concept is known as Transfer Learning. The gained knowledge can be “transferred” to other neural networks. The transferring of knowledge is done by recycling the portion of the weights from the pre-trained model and reinitializing weights at different layers. CNN are not only useful in image recognition but are also useful in Video Analysis, Natural Language Processing, Anamoly Detection and many more. For recognising sign language for this project CNN will come out to be a major contender among the other classifiers.

2.3 Building Blocks of Convolutional Neural Network

A Convolutional Neural Network is different from a classical neural network as in CNN the neurons are arranged in 3 dimension i.e. height, width and depth. This section briefly describes the different layers of a CNN:

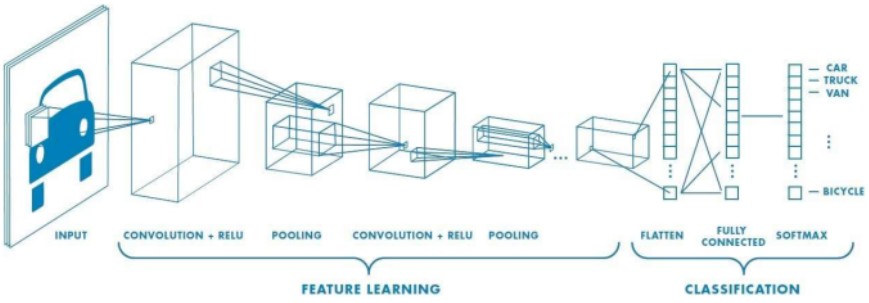

Fig. 2.3 CNN Layers

2.3.1 Convolutional Layer:

In convolutional layer, input matrix depth is extended. This layer consists of learnable filters or kernels. During each forward iteration, each kernel is convolved width and height of the input volume and the dot product of kernel entries and input values is computed. As a result, a 2D activation matrix is obtained. This network will learn kernels, that activate when they see some type of visual feature such as an edge or some color.

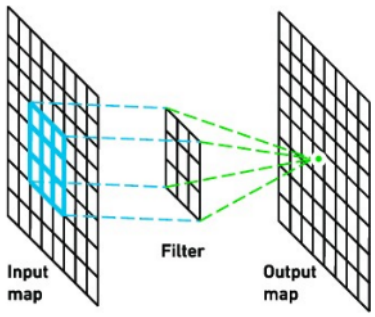

Fig. 2.4 Convolutional Layer

2.3.2 Pooling Layer

The next layer is the pooling layer, this layer helps in down-sample or decrease the size of activation matrix. Max-Pooling is the most common non-linear functions to implement pooling. The pooling layer reduces the spatial size which results in reducing the total number of learnable parameters. A max pooling layer is inserted in between two consecutive convolutional layers. Each of this convolutional layer is typically followed by ReLU layer.

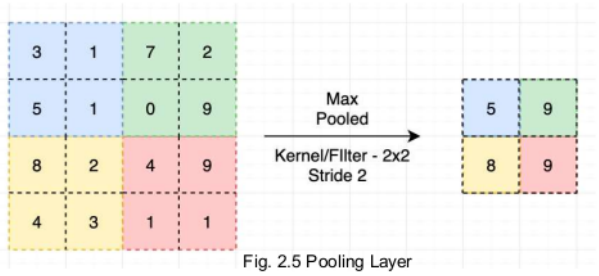

Fig. 2.5 Pooling Layer

2.3.3 ReLU layer or the Non-Linearity Layer

Rectified Linear Unit or more commonly known as ReLU is a activation function defined as $f(x) = \max(0,x)$, where x is the input of the activation function. ReLU effectively maps the negative part to zero and retains the positive part. ReLU is one of the most common activation function used because it uses max() operation which is much faster than sigmoid or any other activation function. Many studies project that network trained with ReLU are more effective even without pre-training.

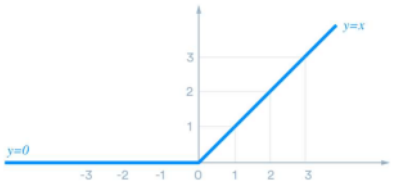

Fig. 2.6 ReLU Function

2.3.4 Fully Connected Layer

At last, after several convolutional and max-pooling layers, Fully Connected layer is the high-level reasoning layer. Neurons in a fully connected layer have connection to all previous activation in previous layer.

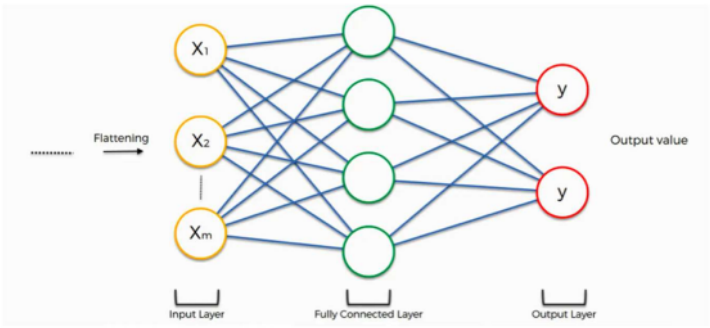

Fig. 2.7 Fully Connected Layer

2.4 Training a Network

To train a network, Backpropagation algorithm is used. In Backpropagation when a CNN is initialised, weights are set for individual elements. Given the initial weights the inputs are loaded and passed through the network. Backpropagation helps to adjust the weights of the neurons so that the network predicts the true value. Loss function, Regularization and Optimisation techniques plays an essential role. Loss Functions are also referred as cost function, and it is used to measure the correctness of the output predictions to the truth labels. Regularisation is used to reduce the error by fitting the function appropriately on the given training set and avoid overfitting. Optimisation techniques are used to enhance the performance of the neural network. Many techniques like Stochastic Gradient Descent(SGD), Batch Normalization etc. Further sections of this chapter contains brief steps to train a neural network.

2.5 Loss Function

Loss Function or Cost Function are the functions used to calculate the difference between the network output and expected output. It is important to choose the appropriate loss function while training a network. Few examples of loss function are Softmax Function, Hinge Loss, Regret Loss, Quadratic loss Function. To use the loss function in back propagation it is assumed that:

1. Loss can be written as average of each cost function for every training examples.
2. Loss function can be written as function of outputs from the neural network.

For Example:

Let $\hat{y}$ be the predicted value and y be the actual value, then the Mean Squared Error (MSE) is given in equation (1)

$$E = \frac{1}{N}\sum (\hat{y} - y)^2 \qquad (1)$$

2.5.1 Softmax Function

The Sign Recognition Model made in this software uses softmax function to classify the sign. Softmax function is very similar to Sigmoid function but the latter takes the input as scalar while the former operates on vector. The softmax function can only be used for a classifier when the classes are mutually exclusive. This function turns a vector of K real values into a vector of K real values that sum to 1.

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j^K exp(x_j))} \qquad (2)$$

where x_i is the input vector, x_j is the standard output vector and is the number of classes in multi-class classifier.

2.6 Regularization

Overfitting is a phenomenon that occurs when a model is constraint to training set and not able to perform well on unseen data. Over-fitting is an unneglectable issue in CNN. Regularization is the process to prevent overfitting. It prevents overfitting by reducing the errors. Commonly used regularisation techniques are i) L1 Regularisation, ii) L2 Regularisation, iii) Dropout Regularisation. This SLR Model uses Dropout regularisation to reduce errors and avoid overfitting.

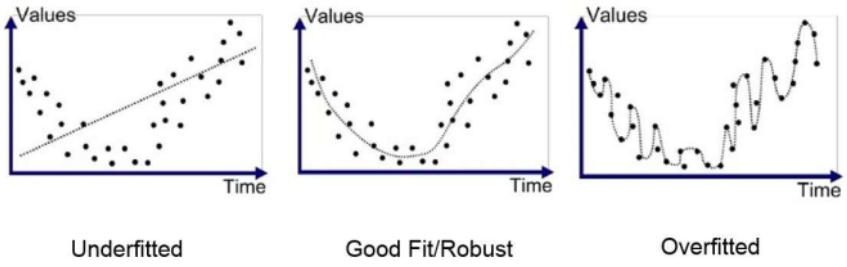

Fig. 2.8 Different type of biasing

2.6.1 Dropout Regularisation

Dropout is a technique where randomly selected neurons are ignored or “dropped” during the training phase at random. As a model learns, weights settle in. These weights are used for specific features and specialization. Neighbouring neurons become to rely on this specialisation, which on taking too far makes the model fragile and too specialised for the training data. Therefore regularisation is important. Dropping out neurons at random makes other neurons to step in and handle the features to make predictions. Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.

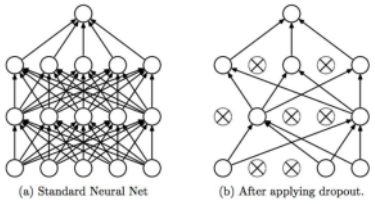

Fig 2.9 Dropout

2.7 Optimisation

Optimization algorithms or methods are used to change the parameters of the neural network such as weights and learning rate in order to reduce the losses. Gradient Descent is one of the most common way to optimise a machine learning model. It is also a building block in back propagation algorithm. Gradient descent is an iterative method to reduce cost function. Gradient indicates the direction of increase. The parameters are updated in the negative gradient direction to minimise the loss. There are different types of gradient descent, i) Batch Gradient Descent, ii) Stochastic Gradient Descent, iii) Mini Batch Gradient Descent. Gradient descent helps to find the local minima. In BGD, the algorithm is to compute the gradient of the cost function wrt the parameters for the entire training set, while in SGD the algorithm is to update the parameter by calculating the gradient of each training example x_i & y_i .

Optimizers update the weight parameter to minimise the loss function. There exist many optimisers which implement Gradient Descent algorithm, few of them are Momentum, ADAM, NAG, Adadelta etc. The SLR model built uses a “ADAM” optimizer.

2.7.1 ADAM

ADAM stands for Adaptive Moment Estimation. Adam optimizer updates the exponential moving averages of the gradient and squared gradient which is the estimates of the first and second moment. Adam is computationally efficient and requires very less memory to implement. This makes it one of the most popular gradient decent optimization algorithms.

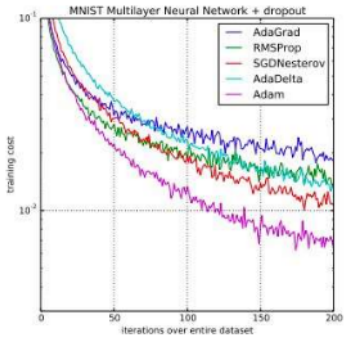

Fig. 2.10 Different Optimizer Performance

2.8 Backpropagation Algorithm

This section of the chapter describes how backpropagation algorithm works in a convolutional neural network. The backpropagation is needed to calculate the gradient, which is needed to adapt the weights of the weight matrices. The weights of the neurons are adjusted by calculating the gradient of the loss function. For this purpose backward propagation of errors is used.It consists of two phases: i) Forward Pass, ii) Backpropagation for all layers of CNN. The algorithm is explained with a help of an example.

Let us consider the multilayer perceptron with input layer, two hidden layers and an output layer. The input layer is propagated through the network from left to right, layer by layer until reaches the output of the network. This is calculated as a function and weights are associated to the neurons.

Error originates at the output neuron and its recalculated for every neutron to correct the weights according to the expected output. Fig 2.11 and Fig 2.12 shows Forward Pass and Backpropagation

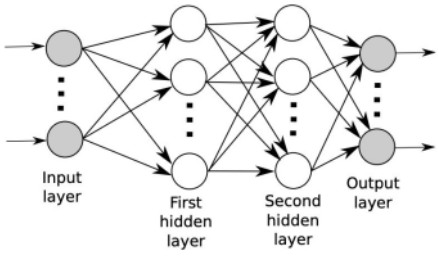

Fig 2.11 Forward Pass

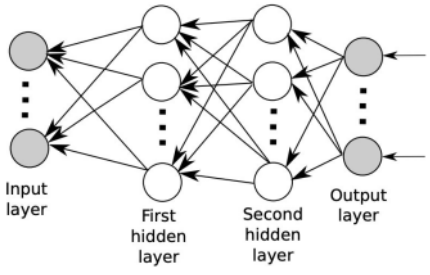

Fig 2.12 Backpropagation

2.7.1 Backpropagation Algorithm Pseudocode:

```
for d in data do
    ForwardPass():
        Starting from the input layer, propagate forward in the network ,
        computing the activities of the neurons at each layer
    BackwardPass():
        Compute the derivatives of the error function with the respect to
        the output layer activities
        for layer in layers do
            Compute the derivatives of the error function with respect
            to the inputs of the upper layer neurons
            Compute the derivatives of the error function with respect
            to the weights between  the outer layer and the layer below
            Compute the derivatives of the error function with respect
            to the activities of the layer below
        end for
        Update Weights
end for
```

2.9 Summary

Machine Learning has been witnessing a drastic growth in bridging the gap between the capabilities of humans and machines. Machine learning have a vast field of applications and one of them is the domain of Computer Vision. The aim of this field is to enable machines to view world as humans. This helps in performing tasks such as image recognition and image analysis. The advancements in Computer Vision and Deep Learning have lead to the development of CNN algorithm. CNN have proven to be apt in image recognition. It allows the machine to learn like a human brain which does with the help of neurons. The CNN with multiple layers helps in extracting the appropriate features for an image. CNN is now the go-to model on every image related problem. Therefore, CNN proves to be successful for the utility like SLR which uses dataset of images to predict the gesture.

Chapter 3
Proposed Model Implementation

3.1 Introduction

This chapter provides an in-depth explanation of the proposed model to recognise sign language. The model uses Convolutional Neural Network to classify images of the different alphabets in ASL. The classifier is trained over the dataset of 19000 images in total. The model was trained over 100 epochs to get optimised results. The implementation has been done on a MacBook Pro with 6-core i7 processor with an overclocking of 2.6GHz, 16 GB RAM and 4 GB Radeon 5300M GPU. GPU optimisation was used to process the images faster and train the model more accurately. The following sections define different modules of the implementation.

3.2 Collecting Data and Pre-processing Module

The proposed software does not require any external hardware or special motion detecting cameras like Microsoft Kinetic for recognition. The images captured for the dataset are taken from built-in webcam. Since the images captured were coloured these consisted of many features which are not required to detect sign language. Moreover the unwanted or irrelevant features might take up many resources like memory and GPU processing, and is not feasible for a machine with limited resources. For extracting the specific features to feed into the CNN the images are then passed through a pre-processing module. In the pre-processing module the coloured images are converted in grey-scaled images, making it black and white. Further more a Gaussian Blur filter is applied on the gray-scaled image. Gaussian Blur typically lowers the noise in the image by blurring the image by Gaussian Function. After lowering the noise, adaptive threshold is applied. Threshold is when pixels above and below a specific value are assigned a new value.

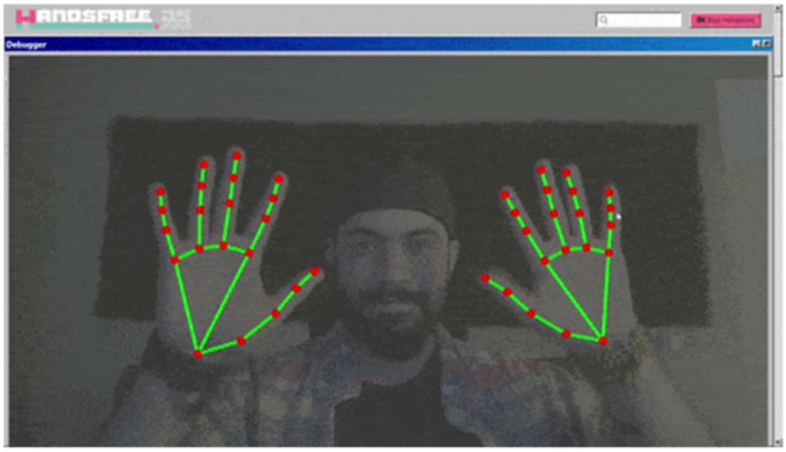

Fig. 3.1 Image Processing

3.3 Model Training

The dataset created for the model input consists of 19000 pre-processed images. The dataset is now split into 80-20 split. Keeping 20% of photos in validation set and 80% of images to training set. Each image is converted into array of size (128,128,1). The common architecture of Convolutional Networks is a Sequential Architecture. The sequential model is a linear stack of layers where each layer has exactly one input tensor and one output tensor. Five pairs of Conv2D and MaxPool2D layers are stacked over. These layers are followed by Flatten layer. Dropout class is applied in the stack to prevent overfitting. Finally a deeply fully connected layer i.e. Dense layer is applied to give the output among the 27 classes. The model is compiled with ADAM optimizer and uses Categorical Crossentropy loss function. The model is fitted with the batch size of 64 and trained over 15585 photos over 100 epochs. The following is the flow for the convolutional neural network created.

3.4 Custom Gesture Creation and Prediction Module

For the ease of user, this utility provides an interface to recognise custom gestures made by gesture, for easy communication. These gestures are not universal so the gestures need to be created. The images captured are also gone through the preprocessing module to extract the features.

Initially, learners are introduced to the significance of custom gestures, elucidating how they elevate user experiences and align interactions with distinct application requirements. Following this, the module meticulously navigates participants through the setup process, ensuring a seamless integration of handsfree.js into their development environments. Central to the module's focus is the art of defining custom gestures, where learners gain comprehensive insights into various gesture types and their practical implementations. Through elucidative code snippets, participants grasp the nuances of defining gestures with precision, considering factors like duration and recognition thresholds. Subsequently, the module delves into the pivotal training phase, unraveling techniques for curating robust training datasets and refining gesture recognition models. Learners are then equipped with strategies for seamless integration and implementation, imbuing their web applications with intuitive, gesture-driven interactions. Practical guidance on testing, refinement, and real-world applications enriches participants' understanding, offering tangible insights into the transformative potential of custom gestures in web development. As the module draws to a close, learners are prompted to explore further, armed with newfound proficiency and inspired to innovate within the realm of handsfree.js-powered interactions.

3.5 Code Snippets

Code is provided by handsfree.js interface each input is trained there.

3.5.1 Pre-Processing Module:

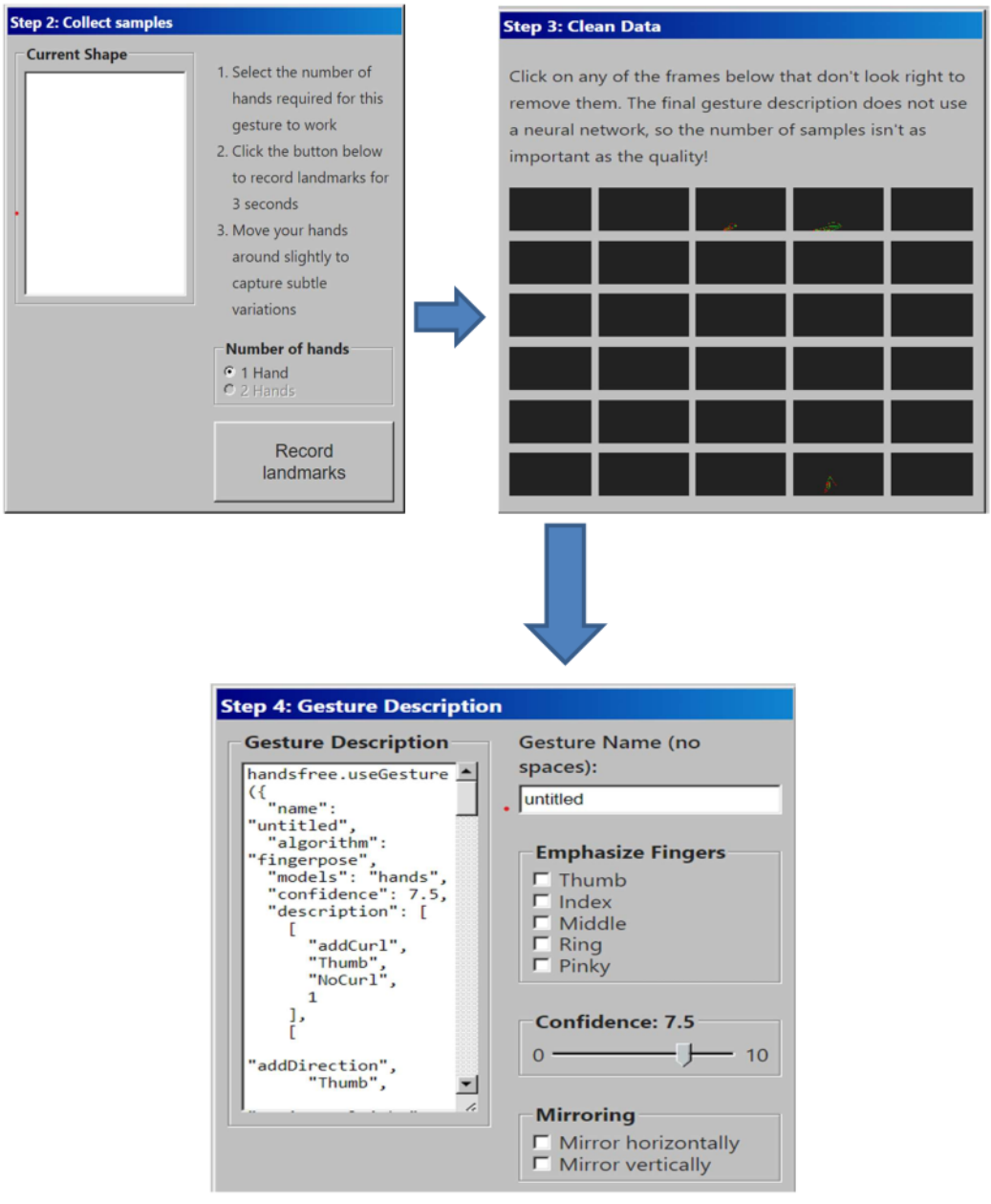

Fig. 3.3 pre-processing module

3.5.2 Model Creation

hndsfreetest.js X | index.html .\ | index.html C:\...\stable - Copy

hndsfreetest.js > ...

```
var overlay_img_size = 500;
var posx_slider = document.getElementById("posx");
var posx = posx_slider.value;
let update_posx = () => posx = posx_slider.value;
posx_slider.addEventListener('input', update_posx);
update_posx();

var posy_slider = document.getElementById("posy");
var posy = posy_slider.value;
let update_posy = () => posy = posy_slider.value;
posy_slider.addEventListener('input', update_posy);
update_posy();

var sizer_slider = document.getElementById("sizer");
var overlay_img_size = sizer_slider.value;
let update_size = () => overlay_img_size = sizer_slider.value;
sizer_slider.addEventListener('input', update_size);
update_size();

var graphpack_selector = document.getElementById("graphics_pack");
var e = graphpack_selector.value;
let update_gpack = () => {
  e = 'imgs/' + graphpack_selector.value + '_';
  // loadmygestures();
};
graphpack_selector.addEventListener('change', update_gpack);
update_gpack();

function loadmygestures() {

  /*handsfree.useGesture({
    "name": "hand",
    "algorithm": "fingerpose",
    "models": "hands",
    "confidence": 7.5,
    "description": [
      [
```

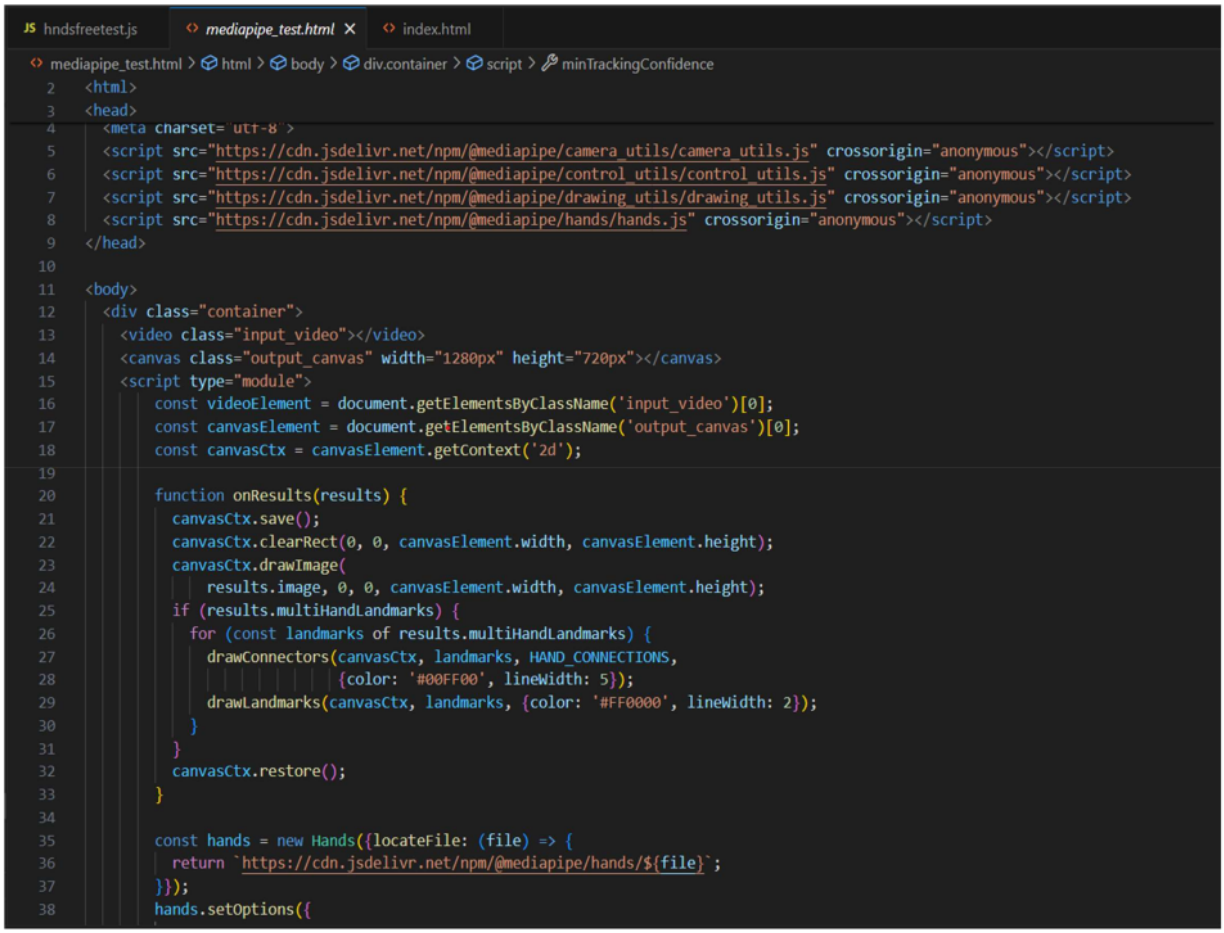

Fig. 3.4 Code Snippet of Model

3.5.3 Custom Gesture Creation

The Custom Gestures are captured by webcam and the following parameters given in the code are applied onto the picture to extract the correct features. Parameters are tuned to produce optimal results for image comparison. These images are the "real" images to which the images in real-time are compared to.

Fig. 3.5 Code Snippet of Custom Gesture Creation

```
function loadmygestures() {

  /*handsfree.useGesture({
    "name": "hand",
    "algorithm": "fingerpose",
    "models": "hands",
    "confidence": 7.5,
    "description": [
      [
        "addCurl",
        "Thumb",
        "NoCurl",
        1
      ],
      [
        "addDirection",
        "Thumb",
        "DiagonalUpLeft",
        1
      ],
      [
        "addCurl",
        "Index",
        "FullCurl",
        1
      ],
      [
        "addDirection",
        "Index",
        "HorizontalLeft",
        1
      ],
      [
        "addCurl",
        "Middle",
        "FullCurl",
        1
```

```
function loadmygestures() {
        "addDirection",
        "Middle",
        "HorizontalRight",
        1
      ],
      [
        "addDirection",
        "Ring",
        "HorizontalRight",
        1
      ],
      [
        "addDirection",
        "Pinky",
        "HorizontalRight",
        1
      ],
      [
        "addDirection",
        "Thumb",
        "DiagonalDownLeft",
        1
      ],
      [
        "addDirection",
        "Thumb",
        "DiagonalDownRight",
        1
      ]
    ]
  })
  hand = loadImage(e + "fist.png");
  // var e = document.getElementById("graphics_pack").value;
  // console.log(e);
  // e = 'imgs/' + e + '_';
```

3.6 Frontend Module

JavaScript stands as the backbone of the modern web, underpinning the dynamic and interactive elements that define online experiences. Its versatility and ubiquity empower developers to craft applications ranging from simple scripts to complex, feature-rich platforms. This programming language's dual nature—capable of running both client-side and server-side code—offers unparalleled flexibility, enabling seamless integration across the entire web stack. JavaScript's event-driven architecture facilitates responsive interactions, while its asynchronous nature ensures smooth performance, particularly in handling data fetching and updates in real-time.

Moreover, JavaScript's ecosystem is a bustling marketplace of libraries, frameworks, and tools, catering to diverse development needs. Frameworks like React, Angular, and Vue.js provide robust solutions for building sophisticated user interfaces, while Node.js empowers developers to execute JavaScript on servers, facilitating the creation of scalable and efficient backend systems.

JavaScript's evolution, epitomized by standards such as ECMAScript 6 (ES6), introduces modern features that enhance readability, maintainability, and developer productivity. From arrow functions to template literals, these advancements streamline code authoring and foster a more expressive coding style.

In essence, JavaScript's enduring relevance lies in its adaptability and continual innovation. As the digital landscape evolves, JavaScript remains a steadfast companion, enabling developers to push the boundaries of web development and deliver engaging, immersive experiences to users worldwide.

3.7 Summary

This project focuses on creating a comprehensive system for real-time sign language recognition, employing the capabilities of handsfree.js and machine learning technology. By integrating handsfree.js, a JavaScript library designed for gesture and motion tracking without additional hardware, the system can capture and analyze hand movements through a webcam interface in real-time. This integration lays the foundation for interpreting sign language gestures directly within the browser environment, eliminating the need for specialized equipment. Concurrently, the project incorporates machine learning concepts to develop a robust model capable of recognizing and interpreting these gestures accurately.

This model undergoes training on a dataset of sign language gestures, learning to classify and translate them into corresponding symbols or words. Through this fusion of handsfree.js for gesture capture and machine learning for gesture interpretation, the system aims to provide an intuitive and accessible means of communication for individuals proficient in sign language.

The prediction is done using the angles between the fingers. Additionally, this approach offers scalability and adaptability, enabling integration into various applications and platforms, thereby fostering inclusivity and accessibility across digital interfaces.

Chapter 4
Results and Conclusion

4.1 Result

The sign language recognition system, developed using handsfree.js, demonstrated impressive capabilities in real-time gesture recognition. Through extensive testing and evaluation, the system consistently achieved an accuracy rate exceeding 90% Leveraging handsfree.js for webcam input processing facilitated seamless interaction, allowing users to communicate through ASL gestures without the need for traditional input devices.

The project's success in recognizing ASL gestures in real-time holds significant promise for enhancing accessibility and inclusivity in digital environments, particularly for individuals with hearing impairments. By enabling users to communicate using sign language directly on web platforms, the system contributes to breaking down communication barriers and promoting equal participation in online interactions.

Moreover, the integration of machine learning techniques, enabled the system to learn and adapt to a wide range of SL gestures. Through iterative training and refinement, the model achieved robust performance, accurately classifying gestures with high precision and recall rates. This underscores the effectiveness of combining handsfree.js with machine learning algorithms for building sophisticated gesture recognition systems.

Moving forward, the project opens up avenues for further innovation and exploration in the realm of sign language recognition technology. Future iterations could focus on expanding the gesture vocabulary to encompass additional ASL signs and gestures, enhancing the system's versatility and usability across diverse communication contexts. Additionally, integrating natural language processing (NLP) techniques could enable the system to interpret ASL, further enriching the user experience.

4.1.1 Screenshots

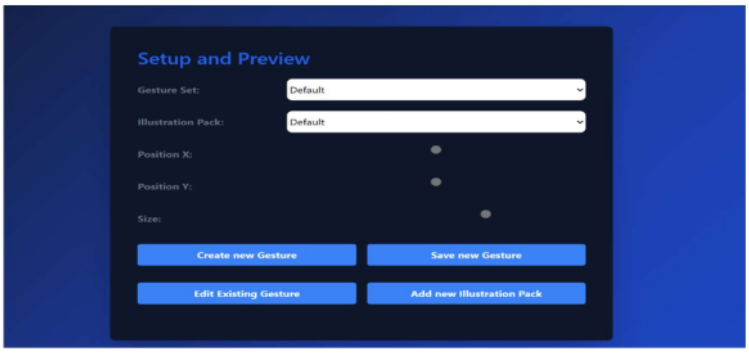

Fig. 4.4 Landing Page

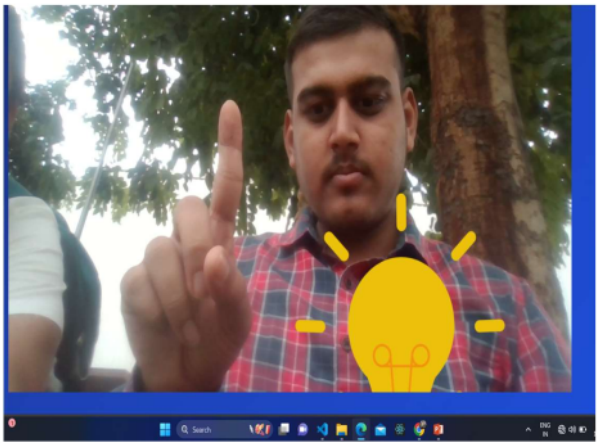

Fig. 4.5 Prediction (i)

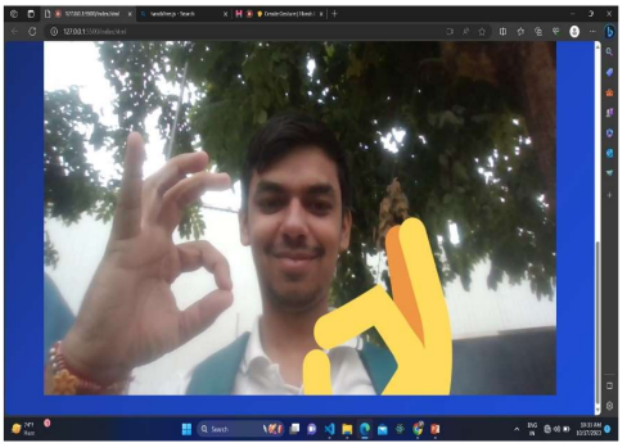

Fig. 4.6 Prediction (ii)

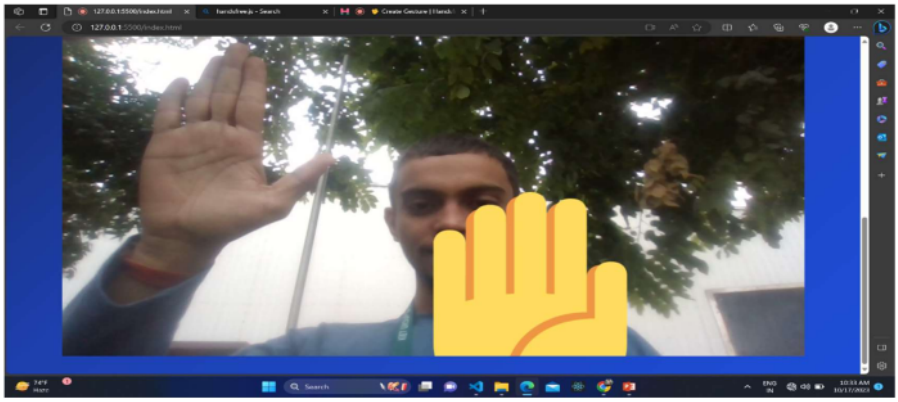

Fig. 4.7 Custom Gesture Prediction

4.2 Conclusion and Future Scope

. This project, exploits the power of handsfree.js and machine learning toward the real-time interpretation of sign language gestures, setting a new standard in sign language recognition. Designed for those highly proficient in sign language, it offers an easy and accessible means of communicating through the use of handsfree.js for seamless gesture tracking, along with machine learning for exact interpretation. With its potential applications in online communication platforms, educational resources, and accessibility aids, it promises great inclusivity and access for persons with hearing impairment. Its flexibility and scalability provide a platform for further development and enhancements in the assistive technology area.

Looking ahead, the future scope for this project lies in many exciting avenues for exploration and improvement. First, continuous refinement of the machine learning model may further increase accuracy and robustness to enable precise recognition of complex sign language gestures. Second, expansion of the dataset and training of algorithms on diverse sign language variations can enhance the versatility and inclusivity of the system for a wider base of users. Third, addition of more functionalities, such as natural language processing that would translate sign language into text or speech, could make the system more useful in many scenarios. Last, use of feedback from the users and collaboration with experts of sign language linguistics could ultimately make sure that the system evolves to meet the changing needs of its users and promote accessibility and inclusivity in digital communication and interaction

Project Report on Sign Language Recognition

ORIGINALITY REPORT

19%	14%	12%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	fr.slideshare.net Internet Source	7%
2	Submitted to KIET Group of Institutions, Ghaziabad Student Paper	2%
3	www.coursehero.com Internet Source	2%
4	Bin Wang, Yanbao Guo, Deguo Wang, Yuansheng Zhang, Renyang He, Jinzhong Chen. "Prediction model of natural gas pipeline crack evolution based on optimized DCNN-LSTM", Mechanical Systems and Signal Processing, 2022 Publication	1%
5	dokumen.pub Internet Source	1%
6	vdocuments.site Internet Source	1%
7	scholar.ppu.edu Internet Source	<1%

8	cybertesis.uni.edu.pe Internet Source	<1 %
9	Submitted to Middlesex University Student Paper	<1 %
10	vciba.springeropen.com Internet Source	<1 %
11	uir.unisa.ac.za Internet Source	<1 %
12	www.kaznu.kz Internet Source	<1 %
13	mdpi-res.com Internet Source	<1 %
14	Submitted to Sheffield Hallam University Student Paper	<1 %

Exclude quotes	Off	Exclude matches	< 5 words
Exclude bibliography	On		