**A Project Report**

on

# DISCERNING OF GESTURES USING MACHINE LEARNING

***Submitted in partial fulfillment of the***

***requirement for the award of the degree of***

Bachelor of Technology in Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The Supervision of**
**Dr. Arvind Kumar**
Associate Professor

**Department of Computer Science and Engineering**

**Submitted By**

19SCSE1010856-ARPIT JAISWAL

19SCSE1180039-RISHABH CHAUHAN

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA**

**DECEMBER - 2021**

# SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
# GALGOTIAS UNIVERSITY, GREATER NOIDA

## CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"DISCERNING OF GESTURES USING MACHINE LEARNING"** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of August, 2021 to December and 2021, under the supervision of Dr. Arvind Kumar Associate Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Arpit Jaiswal (19SCSE1010856)

Rishabh Chauhan (19SCSE1180039)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Arvind Kumar

Associate Professor

## CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Arpit Jaiswal(19SCSE1010856) and Rishabh Chauhan(19SCSE1180039) has been held on Discerning of Gestures using Machine Learning and his/her work is recommended for the award ofBachelor Of Engineering and Technology.

**Signature of Examiner(s)** **Signature of Supervisor**

**Signature of Project Coordinator** **Signature of Dean**

Date: December, 2021

Place: Greater Noida

## Abstract

There have been several advancements in technology and a lot of research has been done to help the people who are deaf and dumb. Aiding the cause, Deep learning, and computer vision can be used too to make an impact on this cause. This can be very helpful for the deaf and dumb people in communicating with others as knowing sign language is not something that is common to all, moreover, this can be extended to creating automatic editors, where the person can easily write by just their hand gestures. In this sign language recognition project, we will create a sign detector, which detects numbers from 1 to 10 that can very easily be extended to cover a vast multitude of other signs and hand gestures including the alphabets.

We are going to develop this project using OpenCV and Keras modules of python. It is fairly possible to get the dataset we need on the internet but in this project, we will be creating the dataset on our own. We will be having a live feed from the video cam and every frame that detects a hand in the ROI (region of interest) created will be saved in a directory (gesture directory) that contains two folders train and test.

The prerequisites software & libraries for the sign language project are:

- Python (3.7.4)
- IDE (Jupyter)
- Numpy (version 1.16.5)
- cv2 (openCV) (version 3.4.2)
- Keras (version 2.3.1)
- Tensorflow (as keras uses tensorflow in backend and for image preprocessing) (version 2.0.0)

Many breakthroughs have been made in the field of artificial intelligence, machine learning and computer vision. They have immensely contributed in how we perceive things around us and improve the way in which we apply their techniques in our everyday lives. Many researches have been conducted on sign gesture recognition using different techniques like ANN, LSTM and 3D CNN. However, most of them require extra computing power . On the other hand, our research paper requires low com- putting power and gives a remarkable accuracy of above 90%. In our research, we proposed to normalise and rescale our images to 64 pixels in order to extract features (binary pixels) and make the system more robust. We use CNN to classify the 10 alphabetical American sign gestures and successfully achieve an accuracy of 98% which is better than other related work stated in this paper.

# List of Tables

## List of Figures

## Acronyms

| | |
|---|---|
| B.Tech. | Bachelor of Technology |
| M.Tech. | Master of Technology |
| BCA | Bachelor of Computer Applications |
| MCA | Master of Computer Applications |
| B.Sc. (CS) | Bachelor of Science in Computer Science |
| M.Sc. (CS) | Master of Science in Computer Science |
| SCSE | School of Computing Science and Engineering |

# Table of Contents

# CHAPTER-1
# INTRODUCTION

Poses and gestures are one the basic means of communication between humans while they may also play a crucial role in human-computer interaction, as they are able to transfer some kind of meaning. The research area of pose and gesture recognition aims to recognizing such expressions, which typically involve some posture and/or motion of the hands, arms, head, or even skeletal joints of the whole body. In certain cases, meaning may differ, based on a facial expression. Several application areas may benefit from the recognition of a human's pose or the gestures she/he performs, such as sign language recognition, gaming, medical applications involving the assessment of a human's condition and even navigation in virtual reality environments. There exist various approaches and techniques which involve some kind of sensor, either "worn" by the subject (e.g., accelerometers, gyroscopes etc.), or monitor the subject's motion (e.g., cameras). In the latter case, the subject may also wear special "markers" which are used to assist in the identification of several body parts and/or skeletal joints. However, during the last few years several approaches rely solely on a typical RGB camera, enhanced by depth information. One such example is the well-known Kinect sensor. Therefore, the user only needs to stand in front of the camera without wearing any kind of sensor. Several parts of her/his body are detected and tracked in the 3D space. Typically, features are extracted and are used for training models to recognize poses and/or gestures.

In this, we present a gesture recognition approach that focuses on hand gestures. We propose a novel deep learning architecture that uses a Convolutional Neural Network (CNN). More specifically, we use the Kinect sensor and its Software Development Kit (SDK) in order to detect and track the subject's skeletal joints in the 3D space. We then select a subset of these joints, i.e., all that are involved at any of the gestures of our data set. Then, we create an artificial image based on these 3D coordinates. We apply the Discrete Fourier Transform on these images and use the resulting ones to train the CNN. We compare our approach with previous work, where a set of handcrafted statistical features on joint trajectories had been used. Finally, we demonstrate that it is possible to efficiently recognize hand gestures without the need of a feature extraction step. Evaluation takes place using a new dataset of 10 hand gestures.

# CHAPTER-2
# Literature Review

The implementation is divided into four main steps:
1. Image Enhancement and Segmentation
2. Orientation Detection
3. Feature Extraction
4 [1]. Classification.

This work was focussed on above four categories but main limitation was change of color was happening very rapidly by the change in the different lighting condition, which may cause error or even failures. For example, due to insufficient light condition, the existence of hand area is not detected but the non-skin regions are mistaken for the hand area because of same color [2].

Involves three main steps for hand gesture recognition system:
1. Segmentation
2. Feature Representation
3. Recognition Techniques.

The system is based on Hand gesture recognition by modeling of the hand in spatial domain. The system uses various 2D and 3D geometric and non-geometric models for modeling. It has used Fuzzy c- Means clustering algorithm which resulted in an accuracy of 85.83%. The main drawback of the system is it does not consider gesture recognition of temporal space, i.e. motion of gestures and it is unable to classify images with complex background i.e. where there are other objects in the scene with the hand objects [3]. This survey focuses on the hand gesture recognition using different steps like data acquisition, pre-processing, segmentation and so on. Suitable input device should be selected for the data acquisition. There are a number of input devices for data acquisition. Some of them are data gloves, marker, and hand images (from webcam/Kinect 3D Sensor). But the limitation with this work was change in the illumination, rotation and orientation, scaling problem and special hardware which is pretty costlier [4].

The system implementation is divided into three phases:
1. Hand gesture recognition using kinetic camera
2. Algorithms for hand detection recognition
3. Hand gesture recognition.

The limitation here is that the edge detection and segmentation algorithms used here are not very efficient when compared to neural networks. The dataset being considered here is very small and can be used to detect very few sign gestures.

The System architecture consists of:
1. Image acquisition
2. Segmentation of hand region.
3. Distance transforms method for gesture recognition [5].

The limitations of this system involve
1. The numbers of gestures that are recognized are less
2. The gestures recognized were not used to control any applications [6].

In this implementation there are three main algorithms that are used:
1. Viola and jones Algorithm.
2. Convex Hull Algorithm.
3. The AdaBoost based learning Algorithm.

The work was accomplished by training a set of feature set which is local contour sequence. The limitations of this system are that it requires two sets of images for classification. One is the positive set that contains the required images, the other is the negative set that contains contradicting images [7].
The system implementation consists of three components:
1. Hand detection
2. Gesture recognition
3. HumanComputer Interaction (HCI).

It has implemented the following methodology:
1. the input image is preprocessed and the hand detector tries to filter out the hand from the input image
2.A CNN classifier is employed to recognize gestures from the processed image, while a Kalman Filter is used to estimate the position of the mouse cursor.
3. The recognition and estimation results are submitted to a control Centre which decides the action to be taken.

One of the limitations of this system is that it recognizes only the static images [8]. This implementation focuses on detection of hand gestures using java and neural networks. It is divided into two phases: -

1. Detection module using java where in the hand is detected using background subtraction and conversion of video feed into HSB video feed thus detecting skin pixels.
2. The second module is the prediction module; a convolutional neural network is used. The input feed image is gained from Java.

The input image is fed into the neural network and is analyzed with respect to the dataset images. One of the limitations of this system is that it requires socket programming in order to connect java and python modules.

# CHAPTER-3
# Dataset Description

A substantial role is played in resolving difficult difficulties in a research dataset. For outstanding study to achieve more immeasurable certainty, a dataset is especially important. Deep learning emphasises the value of datasets. It is the most important factor that allows algorithm training to take place . In deep learning, a huge dataset is important for improving the classification rate. The Senz3d collection of hand movements was used to deal with depth pictures.

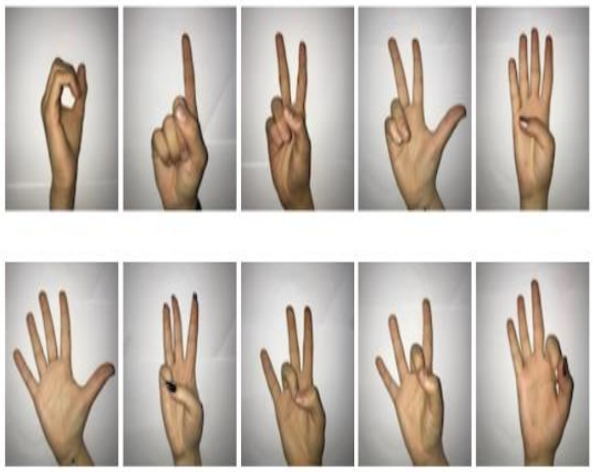


Fig 1: Sample Images from Dataset

The dataset contains a variety of static movements captured with the Creative Senz3D camera . There are 1320 photos in the dataset. Four separate individuals photos make up the dataset. Each person made 11 different gestures 30 times in a row. Color, depth, and confidence frames are accessible for each sample in the collection. Figure 1 shows a selection of photos from the dataset for hand gesture recognition.

# CHAPTER-4
# Proposed Methodology

In our research, the hand gesture reputation device has been proposed through hand segmentation and preprocessing operation (resizing all images) observed through CNN version architecture. However, photo category fashions have turn out to be presently distinguished for pc imaginative and prescient field. After obtaining the intensity photo, segmented the hand data. Following a few preprocessing activities, skilled CNN version for function extraction and gesture reputation. In Fig 2 the block diagram of proposed technique has been shown.

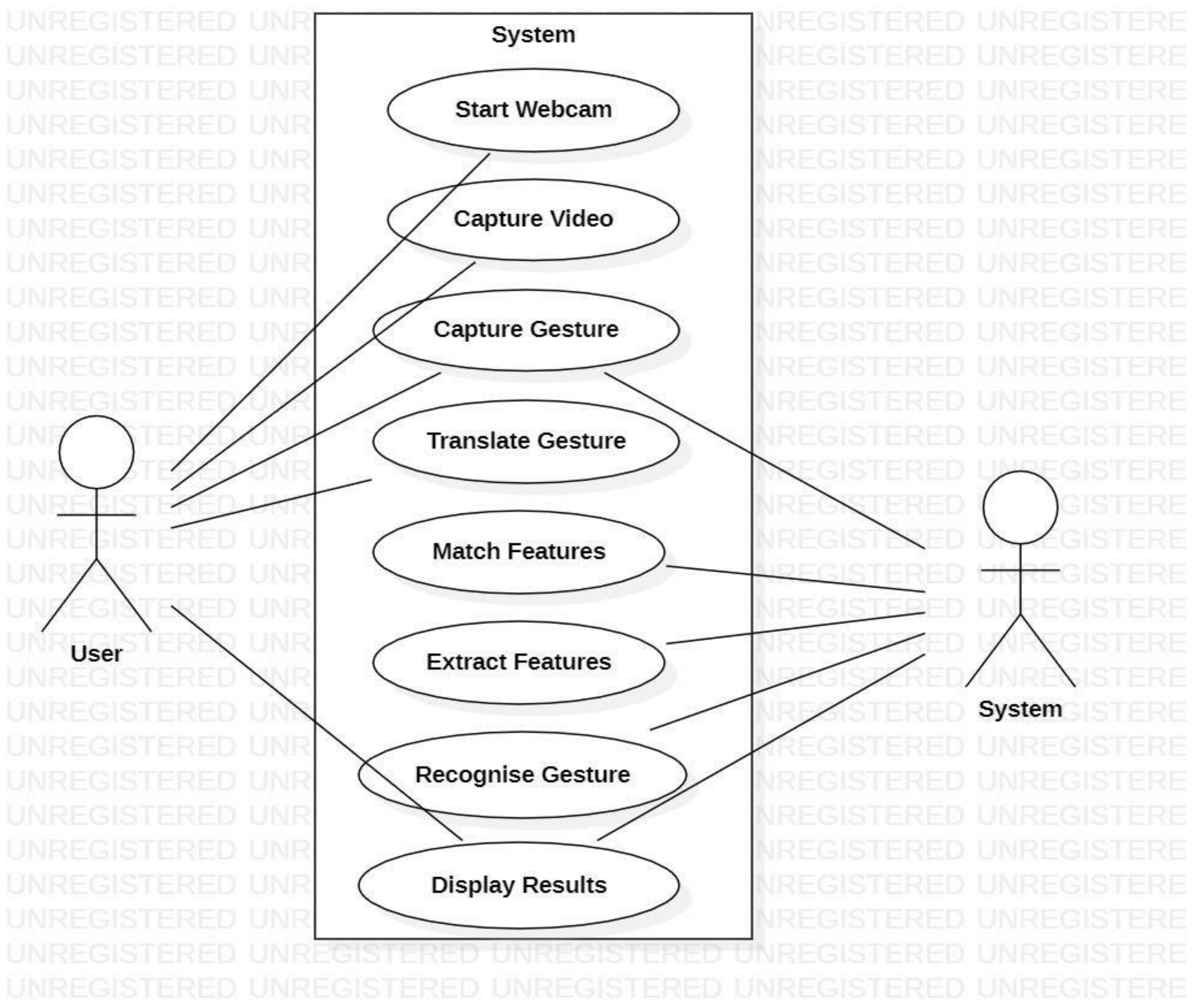


Fig 2 : Use Case Diagram

## 4.1 Hand Segmentation

To facilitate our research, we retrieved depth pictures from the dataset. In hand gesture recognition, the extraction of the hand region is the most crucial stage. As a result, separating the hand region from the depth map is the first stage in recognising motions. The segmentation technique starts with depth value thresholding, which filters out samples with a distance larger than a predefined threshold based on the application . The hand data was split using the YCbCr colour space after the depth photographs were taken. In a basic manner, it distinguishes the hand from the frame. The depth image's YCbCr value was changed. The YCbCr values of each pixel were compared to the reference values. Each parameter has a predetermined threshold value as follows: The following are the Y, Cb, and Cr ranges: $0 \leq Y \leq 255$ and $135 \leq C \leq 180$ and $85 \leq Cb \leq 135$ .

## 4.2 Preprocessing

When training a convolutional neural network, it can be difficult to know how to appropriately prepare visual data [5]. Because the photos in the training dataset were of different sizes, they had to be scaled before being fed into the model [5].This involves both resizing and cropping techniques during both the training and evaluation of the model [4]. After getting the segmented hand images, the images have been resized to 256×256.

# CHAPTER-5
# PROPOSED CNN MODEL

Artificial Intelligence is undergoing a rapid expansion. Volume 174 – No. 16, January 2021 30 of the International Journal of Computer Applications (0975 – 8887) in bridging the gap between human and machine [5]. Computer Vision and deep learning have evolved over time, largely because to one algorithm – the Convolutional Neural Network [5]. Over time, it has been conclusively demonstrated that neural networks surpass alternative algorithms in terms of accuracy and speed. With the progress of neural networks, image categorization has become a hot issue among researchers. In this field, the convolution neural network outperforms traditional machine learning methods. The whole work flow of CNN to process hand movements is depicted in Figure 3.

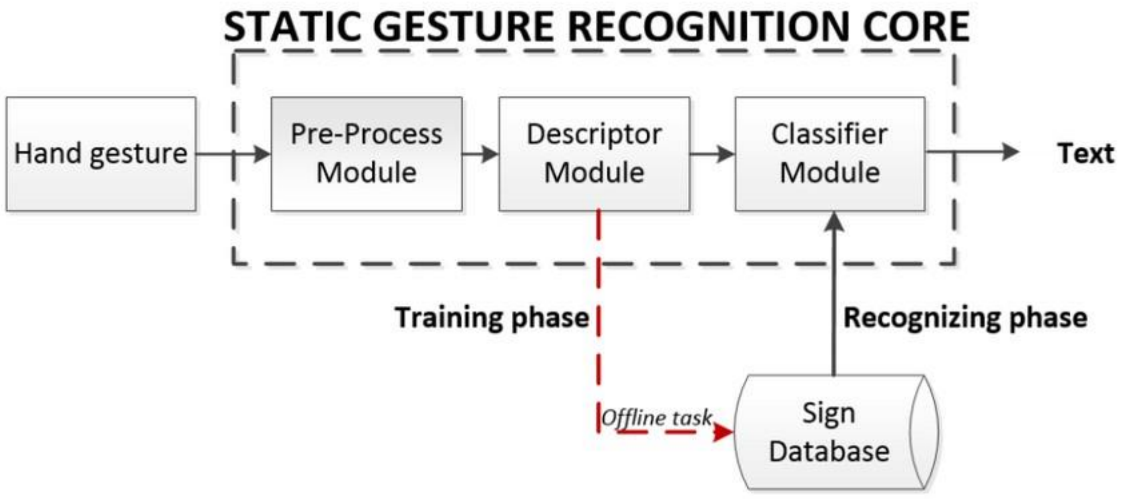


Fig 3: Data Flow Diagram

Each input image will be processed through a series of convolution layers, pooling, and fully connected layers, as well as the ReLU function, which will identify hand motions using probabilistic values. Three convolution layers, two maxpooling layers, and two fully connected layers make up our suggested CNN design. There are 11 nodes in the output layer. These 11 nodes have proposed that the dataset be used to recognise 11 gestures. The dataset was divided into 816 images for training and the rest for validation.

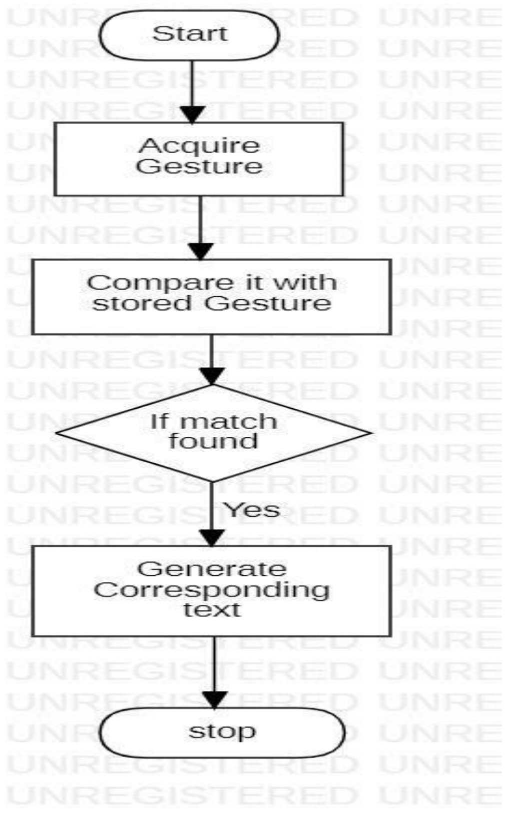


Fig 4: Functionality Flowchart Diagram

### 5.1 Convolution layer

The convolution layer received the normalised pictures. The input photos were scaled to 256 by 256 pixels. Four convolution layers with a tractable feature map have been applied to the input image. A kernel is an array of numbers that are commonly referred to as weights. The kernel size in the convolution layers has been set to 33%. A mathematical kernel has performed an operation on the image. Edge detection, blur, and sharpness can all be achieved by convolutioning an image with multiple filters. In our proposed methodology, three convolution layers were used. The first convolution layer has 64 filters, while the second and third layers had 128 filters. Filters in convolutional layers identify features that improve categorization.

### 5.2 Pooling Layer

Similar to the first layer, the pooling layer lowers the number of parameters and the spatial dimension of features. Pooling feature maps minimises their dimensionality while preserving critical features. Pooling is applied to each feature map separately to produce the same amount of pooled feature maps. In our proposed architecture, we used two pooling layers.. Choosing a pooling formula is part of the pooling process. Max-pooling has been chosen among the various pooling types. For computer vision studies such as image classification, max pooling has been found to perform better than other pooling operations. Unlike average pooling, the results are pooled feature maps that highlight the patch's most prominent feature. Maximum pooling is a pooling procedure that determines the highest value in each feature map area. The kernel's size is always smaller than the feature map's size. By reducing the dimensionality of images, the pooling layer minimizes the processing power required to process them. Furthermore, it is advantageous for extracting powerful rotational and positional invariant characteristics in order to make the training process more effective.

## 5.3 Fully Connected Layer

The convolution neural network's final layer is the fully connected layer. These layers act in the same way as a conventional deep neural network. Every neuron in one layer connects with every neuron in another layer via the completely linked layer. Convolution, like the pooling layer, extracts capabilities; however, completely connected layers classify the data according on the capabilities extracted by the previous layer. The FC layer holds composite records from all of the convolution and pooling layers. To categorise the images, the flattened matrix is handed thru an FC layer. A absolutely linked layer plays the function vector representation (FC). This function vector consists of all the records approximately the enter this is required. This feature vector is then used for classification after the network has been trained. This layer uses non-linear functions to classify photos based on their attributes. A ReLU activation function is included in this layer, which offers a probability for each of the categorization labels.

## 5.4 ReLU Activation Function

In a neural network, the activation function is in charge of modifying the total of weighted input [8]. The rectified linear activation function (ReLU), on the other hand, produces direct output. This linear function outputs the input directly if the input is positive; else, it outputs zero [8]. It has become the default activation function for many types of neural networks since it is faster to train and generally generates better performance[8].

## 5.5 Implementation

### 5.5.1 Create Gesture Dataset

```
import cv2
import numpy as np

background = None
accumulated_weight = 0.5

ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350


def cal_accum_avg(frame, accumulated_weight):

    global background

    if background is None:
        background = frame.copy().astype("float")
        return None

    cv2.accumulateWeighted(frame, background, accumulated_weight)


def segment_hand(frame, threshold=25):
    global background

    diff = cv2.absdiff(background.astype("uint8"), frame)

    _ , thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)

    # Grab the external contours for the image
    image, contours, hierarchy = cv2.findContours(thresholded.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) == 0:
        return None
    else:

        hand_segment_max_cont = max(contours, key=cv2.contourArea)

        return (thresholded, hand_segment_max_cont)
```

```
cam = cv2.VideoCapture(0)

num_frames = 0
element = 10
num_imgs_taken = 0

while True:
    ret, frame = cam.read()

    # filpping the frame to prevent inverted image of captured frame...
    frame = cv2.flip(frame, 1)

    frame_copy = frame.copy()

    roi = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]

    gray_frame = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.GaussianBlur(gray_frame, (9, 9), 0)

    if num_frames < 60:
        cal_accum_avg(gray_frame, accumulated_weight)
        if num_frames <= 59:

            cv2.putText(frame_copy, "FETCHING BACKGROUND...PLEASE WAIT", (80, 400),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)
            #cv2.imshow("Sign Detection",frame_copy)

    #Time to configure the hand specifically into the ROI...
    elif num_frames <= 300:

        hand = segment_hand(gray_frame)

        cv2.putText(frame_copy, "Adjust hand...Gesture for" + str(element), (200,
400), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

        # Checking if hand is actually detected by counting number of contours
detected...
        if hand is not None:

            thresholded, hand_segment = hand

            # Draw contours around hand segment
            cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1,
(255, 0, 0),1)
```

```
            cv2.putText(frame_copy, str(num_frames)+"For" + str(element), (70, 45),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

            # Also display the thresholded image
            cv2.imshow("Thresholded Hand Image", thresholded)

    else:

        # Segmenting the hand region...
        hand = segment_hand(gray_frame)

        # Checking if we are able to detect the hand...
        if hand is not None:

            # unpack the thresholded img and the max_contour...
            thresholded, hand_segment = hand

            # Drawing contours around hand segment
            cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1,
(255, 0, 0),1)

            cv2.putText(frame_copy, str(num_frames), (70, 45),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
            #cv2.putText(frame_copy, str(num_frames)+"For" + str(element), (70,
45), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
            cv2.putText(frame_copy, str(num_imgs_taken) + 'images' +"For" +
str(element), (200, 400), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

            # Displaying the thresholded image
            cv2.imshow("Thresholded Hand Image", thresholded)
            if num_imgs_taken <= 300:
                #cv2.imwrite(r"D:\\gesture\\train\\"+str(element)+"\\" +
str(num_imgs_taken+300) + '.jpg', thresholded)
                cv2.imwrite(r"D:\\gesture\\x"+"\\" + str(num_imgs_taken) + '.jpg',
thresholded)
            else:
                break
            num_imgs_taken +=1
        else:
            cv2.putText(frame_copy, 'No hand detected...', (200, 400),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

    # Drawing ROI on frame copy
    cv2.rectangle(frame_copy, (ROI_left, ROI_top), (ROI_right, ROI_bottom),
(255,128,0), 3)
```

```
    cv2.putText(frame_copy, "DataFlair hand sign recognition_ _ _", (10, 20), 
cv2.FONT_ITALIC, 0.5, (51,255,51), 1)

    # increment the number of frames for tracking
    num_frames += 1

    # Display the frame with segmented hand
    cv2.imshow("Sign Detection", frame_copy)

    # Closing windows with Esc key...(any other key with ord can be used too.)
    k = cv2.waitKey(1) & 0xFF

    if k == 27:
        break

# Releasing camera & destroying all the windows...

cv2.destroyAllWindows()
cam.release()
```

### 5.5.2 Train CNN

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, 
MaxPool2D, Dropout
from keras.optimizers import Adam, SGD
from keras.metrics import categorical_crossentropy
from keras.preprocessing.image import ImageDataGenerator
import itertools
import random
import warnings
import numpy as np
import cv2
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import ModelCheckpoint, EarlyStopping
warnings.simplefilter(action='ignore', category=FutureWarning)


train_path = r'D:\gesture\train'
test_path = r'D:\gesture\test'
```

```
train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_in
put).flow_from_directory(directory=train_path, target_size=(64,64),
class_mode='categorical', batch_size=10,shuffle=True)
test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_in
put).flow_from_directory(directory=test_path, target_size=(64,64),
class_mode='categorical', batch_size=10, shuffle=True)

imgs, labels = next(train_batches)


#Plotting the images...
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(30,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()


plotImages(imgs)
print(imgs.shape)
print(labels)

model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(64,64,3)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding =
'same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding =
'valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64,activation ="relu"))
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.2))
```

```
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.3))
model.add(Dense(10,activation ="softmax"))


# In[23]:


model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1,
min_lr=0.0001)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0,
mode='auto')



model.compile(optimizer=SGD(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1,
min_lr=0.0005)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0,
mode='auto')


history2 = model.fit(train_batches, epochs=10, callbacks=[reduce_lr,
early_stop],  validation_data = test_batches)#, checkpoint])
imgs, labels = next(train_batches) # For getting next batch of imgs...

imgs, labels = next(test_batches) # For getting next batch of imgs...
scores = model.evaluate(imgs, labels, verbose=0)
print(f'{model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of
{scores[1]*100}%')


#model.save('best_model_dataflair.h5')
model.save('best_model_dataflair3.h5')

print(history2.history)

imgs, labels = next(test_batches)

model = keras.models.load_model(r"best_model_dataflair3.h5")

scores = model.evaluate(imgs, labels, verbose=0)
print(f'{model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of
{scores[1]*100}%')
```

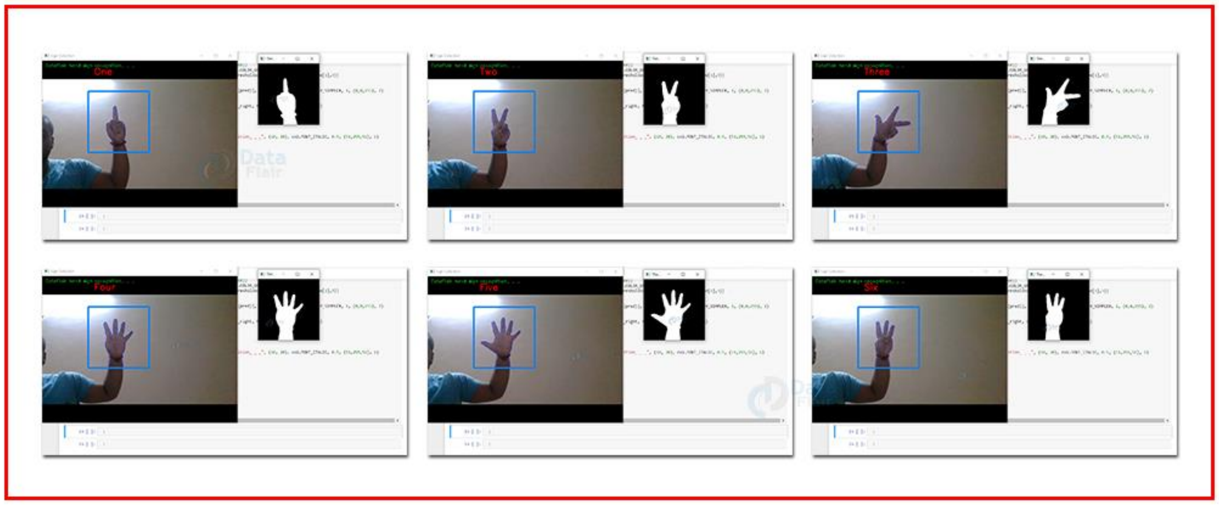
```
model.summary()

scores #[loss, accuracy] on test data...
model.metrics_names


word_dict =
{0:'One',1:'Ten',2:'Two',3:'Three',4:'Four',5:'Five',6:'Six',7:'Seven',8:'Eight',9:
'Nine'}

predictions = model.predict(imgs, verbose=0)
print("predictions on a small set of test data--")
print("")
for ind, i in enumerate(predictions):
    print(word_dict[np.argmax(i)], end='   ')

plotImages(imgs)
print('Actual labels')
for i in labels:
    print(word_dict[np.argmax(i)], end='   ')

print(imgs.shape)

history2.history
```

### 5.5.3 Model for Gesture

```
import numpy as np
import cv2
import keras
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

model = keras.models.load_model(r"C:\Users\abhij\best_model_dataflair3.h5")

background = None
accumulated_weight = 0.5

ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350
```

```
def cal_accum_avg(frame, accumulated_weight):

    global background

    if background is None:
        background = frame.copy().astype("float")
        return None

    cv2.accumulateWeighted(frame, background, accumulated_weight)



def segment_hand(frame, threshold=25):
    global background

    diff = cv2.absdiff(background.astype("uint8"), frame)


    _ , thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)

    #Fetching contours in the frame (These contours can be of hand or any other
object in foreground) ...
    image, contours, hierarchy = cv2.findContours(thresholded.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # If length of contours list = 0, means we didn't get any contours...
    if len(contours) == 0:
        return None
    else:
        # The largest external contour should be the hand
        hand_segment_max_cont = max(contours, key=cv2.contourArea)

        # Returning the hand segment(max contour) and the thresholded image of
hand...
        return (thresholded, hand_segment_max_cont)

cam = cv2.VideoCapture(0)
num_frames =0
while True:
    ret, frame = cam.read()

    # filpping the frame to prevent inverted image of captured frame...
    frame = cv2.flip(frame, 1)

    frame_copy = frame.copy()

    # ROI from the frame
    roi = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]

    gray_frame = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.GaussianBlur(gray_frame, (9, 9), 0)


    if num_frames < 70:

        cal_accum_avg(gray_frame, accumulated_weight)

        cv2.putText(frame_copy, "FETCHING BACKGROUND...PLEASE WAIT", (80, 400),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)

    else:
        # segmenting the hand region
        hand = segment_hand(gray_frame)


        # Checking if we are able to detect the hand...
        if hand is not None:

            thresholded, hand_segment = hand

            # Drawing contours around hand segment
            cv2.drawContours(frame_copy, [hand_segment + (ROI_right, ROI_top)], -1,
(255, 0, 0),1)

            cv2.imshow("Thesholded Hand Image", thresholded)

            thresholded = cv2.resize(thresholded, (64, 64))
            thresholded = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2RGB)
            thresholded = np.reshape(thresholded,
(1,thresholded.shape[0],thresholded.shape[1],3))

            pred = model.predict(thresholded)
            cv2.putText(frame_copy, word_dict[np.argmax(pred)], (170, 45),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

    # Draw ROI on frame_copy
    cv2.rectangle(frame_copy, (ROI_left, ROI_top), (ROI_right, ROI_bottom),
(255,128,0), 3)

    # incrementing the number of frames for tracking
    num_frames += 1

    # Display the frame with segmented hand
    cv2.putText(frame_copy, "DataFlair hand sign recognition_ _ _", (10, 20),
cv2.FONT_ITALIC, 0.5, (51,255,51), 1)
    cv2.imshow("Sign Detection", frame_copy)


    # Close windows with Esc
    k = cv2.waitKey(1) & 0xFF

    if k == 27:
        break

# Release the camera and destroy all the windows
cam.release()
cv2.destroyAllWindows()
```

# CHAPTER-6
## Result



We have successfully developed sign language detection project. This is an interesting machine learning python project to gain expertise. This can be further extended for detecting the English alphabets.

# CHAPTER-7
# CONCLUSION

The activation function in a neural network is in charge of changing the total weighted input [8]. On the other hand, the rectified linear activation function (ReLU) generates direct output. If the input is positive, this linear function sends the input directly; otherwise, it outputs zero [8]. Because a model that employs it is faster to train and delivers superior results in general, it has become the default activation function for many types of neural networks [8].

## REFERENCES

[1]. M. Panwar and P. Singh Mehra, “Hand gesture recognition for human computer interaction,” 2011 International Conference on Image Information Processing, Shimla, pp. 1-7, 2011.

[2]. Rafiqul Zaman Khan and Noor Adnan Ibraheem. “Comparitive Study of Hand Gesture Recognition System.” International Conference of Advanced Computer Science & Information Technology, 2012.

[3]. Arpita Ray Sarkar, G. Sanyal, S. Majumder. “Hand Gesture Recognition Systems: A Survey.” International Journal of Computer Applications, vol. 71, no.15, pp. 25-37, May 2013.

[4]. Manjunath AE, Vijaya Kumar B P, Rajesh H. “Comparative Study of Hand Gesture Recognition Algorithms.” International Journal of Research in Computer and Communication Technology, vol 3, no. 4, April 2014.

[5]. Dnyanada R Jadhav, L. M. R. J Lobo, Navigation of PowerPoint Using Hand Gestures, International Journal of Science and Research (IJSR) 2015. [6]. Ruchi Manish Gurav, Premanand K. Kadbe, Real time finger tracking and contour detection for gesture recognition using OpenCV, IEEE Conference May 2015, Pune India.

[7]. Pei Xu, Department of Electrical and Computer Engineering, University of Minnesota, A Real-time Hand Gesture Recognition and Human-Computer Interaction System, Research Paper April 2017.

[8]. P. Suganya, R. Sathya, K. Vijayalakshmi. “Detection and Recognition of Gestures to Control the System Applications by Neural Networks.” International Journal of Pure and Applied Mathematics, vol. 118, no. 10, pp. 399-405, January 2018.