"SIGN LANGUAGE RECOGNITION"

A Project Report Submitted in Partial Fulfilment of the Requirements for the Degree

of

Bachelor of Technology

in

Computer Science & Engineering

By

S.No.	Student Name	Roll Number	Branch & Section	Group Number:
1	Ankit Kumar	1901640100054	CS 3A	21-A-6
2	Anurag Singh Bhadauria	1901640100066	CS 3A	

KCS-554

Under the Supervision of
Mr Rajat Verma
(Assistant Professor)

Pranveer Singh Institute of Technology, Kanpur
Dr. A.P.J Abdul Kalam Technical University, Lucknow

Year-2021

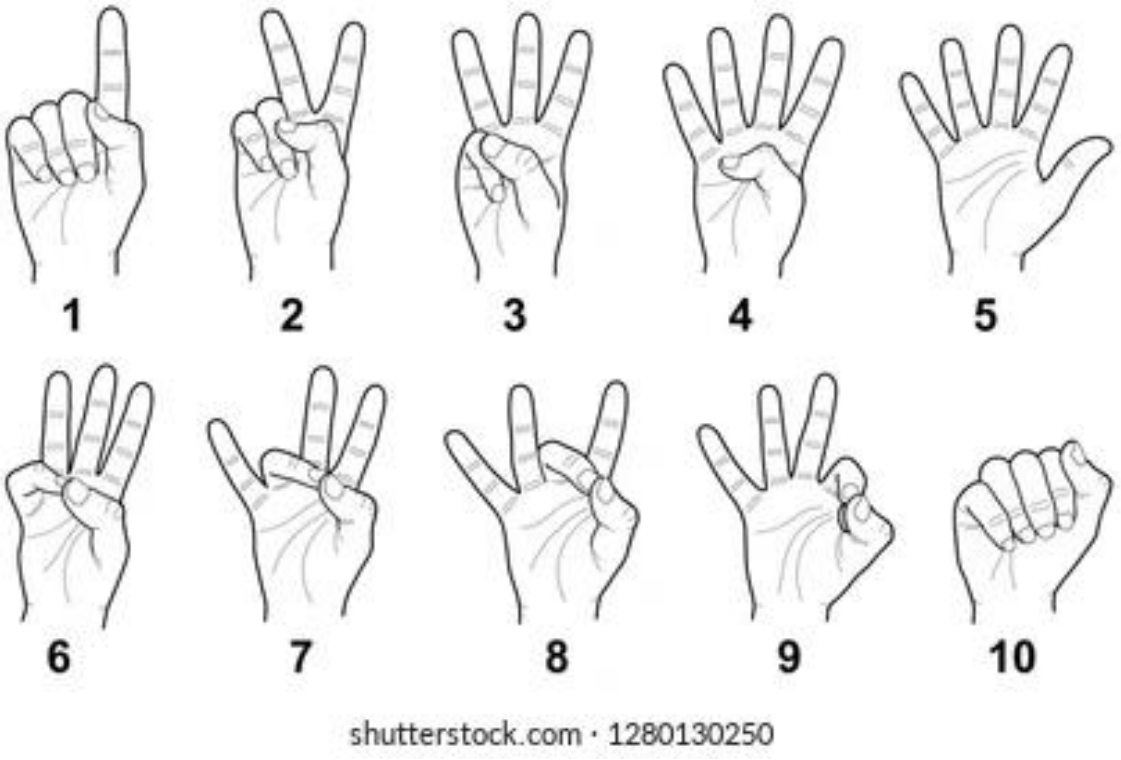
2.Project Objective

Speech impaired people use hand signs and gestures to communicate.
Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people

The main objective of the project Sign Language Recognition is to create a simple machine learrnig model for recognizing the hand signs usually
From 0 to 9(only number).

Further more objectives:

- To bridge the gap between physically challenged people and normal people
- To learn about ML Neural Networks.



3.Introduction

Speech impaired people use hand signs and gestures to communicate. Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

2.1 Sign Language

It is a language that includes gestures made with the hands and other body parts, including facial expressions and postures of the body.It used primarily by people who are deaf and dumb. There are many different sign languages as, British, Indian and American sign languages. British sign language (BSL) is not easily intelligible to users of American sign Language (ASL) and vice versa .

2.2 Phases (image processing)

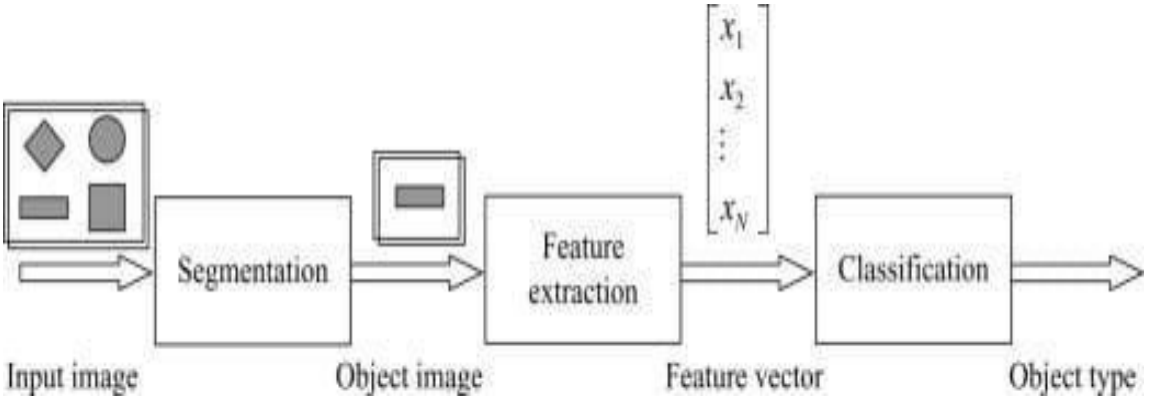


Fig1.1: Phases of pattern recognition

2.3 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools.
- Analysing and manipulating the image.
- Output in which result can be altered image or report that is based on image analysis.

4.Feasibility Study:

What are the user's demonstrable needs?

User needs a web-based system, which will remove all the above-mentioned Problems that, the user is facing. The user wants a web-based system, which will reduce the bulk of paperwork, provide ease of work, flexibility, fast record finding, modifying, adding, removing and generating the reports.

How can the problem be redefined?

We proposed our perception of the system, in accordance with the problems of existing system by making a full layout of the system on paper. We tallied the problems and needs by existing system and requirements. We were further updating in the layout in the basis of redefined the problems. In feasibility study phase we had undergone through various steps, which are described as under:

How feasible is the system proposed?

This was analyzed by comparing the following factors with both the existing system and proposed system.

Cost : The cost required in the proposed system is comparatively less to the existing system.

Effort : Compared to the existing system the proposed system will provide a better working environment in which their will be ease of work and the effort required will be comparatively less than the existing system.

Time : Also the time required generating a report or for doing any other work will be comparatively very less than in the existing system. Record finding and updating will take less time than the existing system.

5 . LITERATURE REVIEW

The domain analysis that we have done for the project mainly involvedunderstanding the neural networks

4.1TensorFlow:

TensorFlow is a free and open-source software library for dataflow and differentiable 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.

Features: TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell Julia, MATLAB,R, Scala, Rust, OCaml, and Crystal."New language support should be built on top of the C API. However, not all functionality is available in C yet." Some more functionality is provided by the Python API.

Application: Among the applications for which TensorFlow is the foundation, are automated image-captioning software, suchas DeepDream.

4.2 OpenCV:

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.[1] Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel[2]). The library is cross-platform and free for use under the open-source BSD license.

OpenCV's application areas include:

- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Object identification

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Support vector machine (SVM)

4.3 Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

Features: Keras contains numerous implementations of commonly used neural- network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

4.4 NUMpy:

NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/ (NUM-pee)) is a library for the Python programming language, adding support for large, multi- dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric,was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Features: NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

4.5 Convolution neural network:

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users' photos.

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at eachpoint.

To solve this problem the computer looks for the characteristics of the baselevel. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

Applications of convolution neural network:

Decoding Facial Recognition:

Facial recognition is broken down by a convolutional neural network into the followingmajor components -

- Identifying every face in the picture
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features

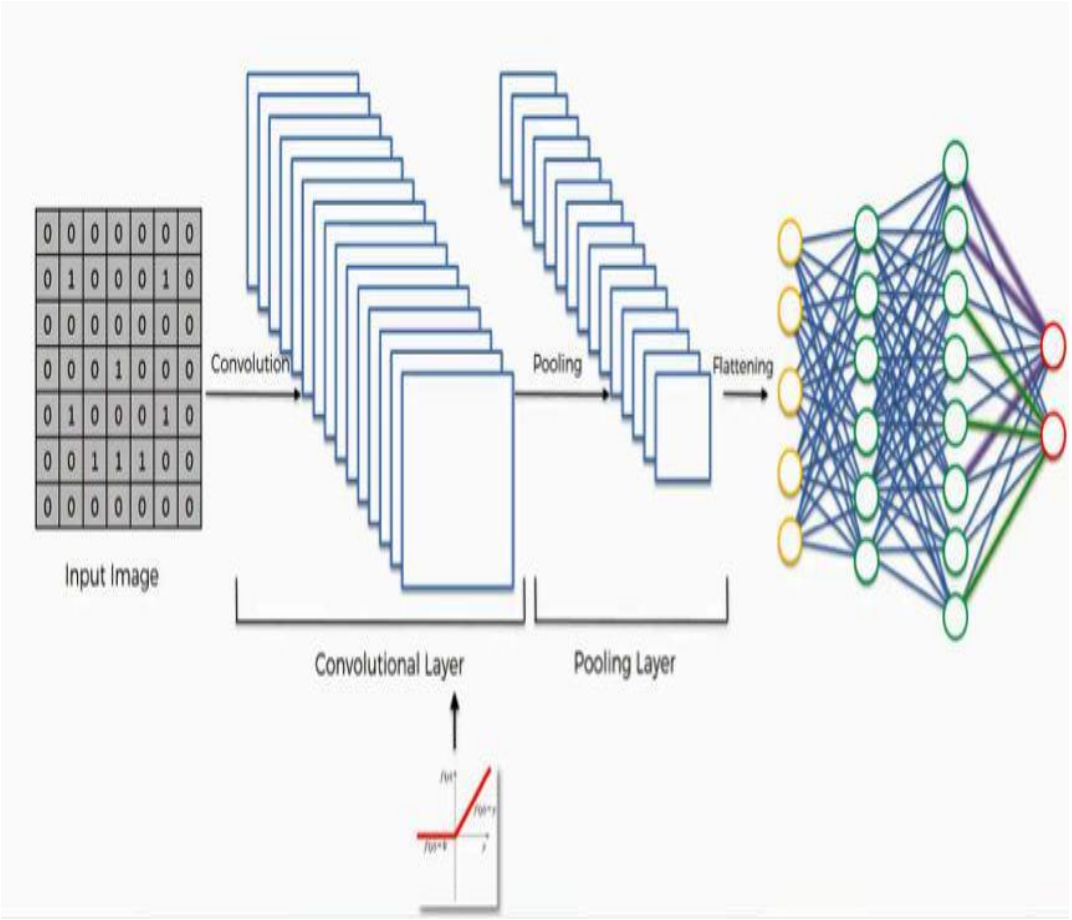


Fig 4.1: Layers involved in CNN

6 . TECHNOLOGY USED

Language: : P Y T H O N (3 . 7 . 4)
Development Environment : PYCHARM)

Libraries and Modules Used:

1. NUMPY
2. CV2 (O P E N C V)
3. KERAS
4. TENSORFLOW

7 . HARDWARE / SOFTWARE REQUIRED:

6.1 SOFTWARE REQUIREMENTS:

1.OPERATING SYSTEM: Windows

2.SDK: OpenCV, Tensorflow, Keras,Numpy

3.IDE: Pycharm

4.Coding Language: Python (3.7 or above)

6.2 HARDWARE REQUIREMENTS:

1. External device: Webcam/Camera

2 .RAM: Minimum 4GB

3. Processor: Intel Pentium 4 or higher

4. HDD: 10 GB or higher

5. Monitor: coloured Monitor

6.Mouse/Keyboard

8. CODING

3Phases:

A) creating Data set

B) Training ML model

C) Predicting gestures

1. CREATING DATA SET

```
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 = 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)



    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)


        if hand is not None:

            thresholded, hand_segment = hand


            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)


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

    else:


        hand = segment_hand(gray_frame)


        if hand is not None:
```

```
            thresholded, hand_segment = hand


            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_imgs_taken) + 'images' +"For" +
str(element), (200, 400), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)


            cv2.imshow("Thresholded Hand Image", thresholded)
            if num_imgs_taken <= 300:

                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


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

2. TRAINING ML MODEL

```
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.preproces
s_input).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.preproces
s_input).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
```

3 .PREDICTING GESTURES

```
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()
```

9.CONCLUSION

Nowadays, applications need several kinds of images as sources of information for elucidation and analysis. Several features are to be extracted so as to perform various applications. When an image is transformed from one form to another such as digitizing, scanning, and communicating, storing, etc. degradation occurs. Therefore ,the output image has to undertake a process called image enhancement, which contains of a group of methods that seek to develop the visual presence of an image. Image enhancement is fundamentally enlightening the interpretability or awareness of information in images for human listeners and providing better input for other automatic image processing systems. Image then undergoes feature extraction using various methods to make the image more readable by the computer.Sign language recognition system is a powerful tool to preparae an expert knowledge, edge detect and the combination of inaccurate information from different sources.

the intend of convolution neural network is to get the appropriate classification

10.Future work

The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. Instead of displaying letter labels it will be more appropriate to display sentences as more appropriate translation of language.This also increases readability.The scope of different sign languages can be increased. More training data can be added to detect the letter with more accuracy. This project can further be extended to convert the signs to speech.

REFERENCES

1. https://www.sciencedirect.com/science/article/pii/S1877050917320720
2. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural network-cnn-deep-learning-99760835f148
3. https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To Understanding-Convolutional-Neural-Networks/