SIGN LANGUAGUE RECOGNITION USING DEEP LEARNING

Design & developed by

R. KIRAN 2011CS020436

GUIDED BY

Dr. R V S S NAGINI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML)

MALLA REDDY UNIVERSITY, HYDERABAD

2020-2024

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled **SIGN LANGUAGUE RECOGNITION USING DEEP LEARNING,** submitted by **Y. CHANDRA SHEKAR 2011CS020435** of B. Tech II year II semester, Department of CSE (AI&ML) during the year 2021-22. The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Head of the department**
Dr. R V S S Nagini	**Dr. Thayyaba Khatoon**
	(CSE-AIML)

External Examiner

ACKNOWLEDGEMENT

I would like to express our gratitude to all those who extended their support and suggestions to come up with this application. Special Thanks to my mentor Dr. R V S S Nagini whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

I sincerely thank my HOD Dr. Thayyaba Khatoon for her constant support and motivation all the time. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least my sincere appreciation goes to my family who has been tolerant understanding my moods, and extending timely support.

ABSTRACT

Computer vision is the study it deals with the computers and systems which understand or takes data through images, videos, or any other visual inputs. It acts as a human visual system and can carry on tasks we specify. Hand gesture recognition is very significant for human-computer interaction. Hand gesture recognition systems received great attention in the recent few years because of their applications and the ability to interact with machines efficiently through human-computer interaction. These are a form of nonverbal communication that can be used in several fields such as communication between deaf-mute people, robot control, human-computer interaction (HCI), home automation, and medical applications. Due to the effect of lighting and complex background, most visual hand gesture recognition systems work only in a restricted environment. An adaptive skin color model based on face detection is utilized to detect skin color regions like hands.

INDEX

1. INTRODUCTION:

1.1 PROJECT INTRODUCTION:

Hand gestures are a form of nonverbal communication that can be used in several fields such as communication between deaf-mute people, robot control, human-computer interaction (HCI), home automation, and medical application. This is an area with many different possible applications, giving users a simpler and more natural way to communicate with robots/systems interfaces, without the need for extra devices. The vision-based technology of hand gesture recognition is an important part of human-computer interaction (HCI). In the last decades, keyboard and mouse play a significant role in human-computer interaction. However, owing to the rapid development of hardware and software, new types of HCI methods have been required. In particular, technologies such as speech recognition and gesture recognition receive great attention in the field of HCI.

The gesture is a symbol of physical behavior or emotional expression. It includes body gestures and hand gestures. It falls into two categories: static gesture and dynamic gesture. For the former, the posture of the body or the gesture of the hand denotes a sign. For the latter, the movement of the body or the hand conveys some messages. Gestures can be used as a tool of communication between computers and humans. Some methods used additional hardware devices such as data glove devices and color markers to easily extract the comprehensive description of gesture features. Other methods are based on the appearance of the hand using the skin color to segment the hand and extract necessary features, these methods are considered easy, natural, and less cost compared with methods mentioned before.

In our project gestures are continuously monitored by the webcam. Each gesture corresponds to a specific command. The workflow of hand gesture recognition is described as follows. First, the hand region is detected from the original images from the input devices. Then, some kinds of features are extracted to describe hand gestures. Last, the recognition of hand gestures is accomplished by measuring the similarity of the feature data.

1.2. PROBLEM STATEMENT:

Our idea is to create a system that can identify specific human hand gestures and use them to address some of the basic handicapped people to interact with the systems also people to intrect with machines using convolutional neural networks and it can also act as a medium of communication between handicapped and the person who does not know the sign language.

1.3. MOTIVATION:

Biometric technologies make use of various physical and behavioral characteristics of human such as fingerprints, expression, face, hand gestures and movement. These features are then processed using sophisticated machines for detection and recognition and hence used for security purposes. Unlike common security measures such as passwords, security cards that can easily be lost, copied or stolen; these biometric features are unique to individuals and there is little possibility that these pictures can be replaced or altered.

Among the biometric sector hand gesture recognition are gaining more and more attention because of their demand regarding security for law enforcement agency as well as in private sectors such as surveillance systems.

In video conferencing system, there is a need to automatically control the camera in such a way that the current speaker always has the focus. One simple approach to this is to guide the camera based on sound or simple cues such as motion and skin color.
Hand gestures are important to intelligent human and computer interaction to build fully automated systems that analyze information contained in images, fast and efficient hand gesture recognition algorithms are required.

1.4. OBJECTIVE:

- The user can interact with the virtual environment using hand gestures.
- The hand gestures are predicted by the machine.
- The gestures are predicted and result will be displayed
- No Special Hardware is Necessary, except for the Camera.

1.5. EXISTING SYSTEM

Hand Gesture Recognition involves a variety of techniques from diverse areas. First let us do prior work related to Sign Language Recognition. The first step of Sign Language Recognition is to capture the symbols performed by the user. To do this we need to examine sensors that provide the best balance between frame rate, accuracy, and affordability. Then, we present a set of sensors that cover a raise of capture devices and their aspects. With sensors to capture the scene the next step is to track and recognize the user's body and different parts of it, such as the face, arms, and hands. Methods for recognizing bodies and body parts have been widely studied and a diverse set of applications created. Neural Networks (NN) are often used as recognition models for image processing. In this project, we present a basic NN model and some variations applied to different areas. We focus on a specific type of network, Convolutional Neural Network (CNN), a popular NN in image processing because of its success in recognizing local features. This way of interacting avoids the use of physical control devices, allowing natural
with the computer. The focus is on developing methods to provide an effective user experience when interacting with the computer.

1.6. PROPOSED SYSTEM :

In this proposed system, The user can interact with the virtual environment using hand gestures. Hand gesture recognition is very significant for human-computer interaction. Hand gesture recognition systems received great attention in the recent few years because of their applications and the ability to interact with machines efficiently through human-computer interaction. These are a form of nonverbal communication that can be used in several fields such as communication between deaf-mute people, robot control, human-computer interaction (HCI), home automation, and medical applications. Due to the effect of lighting and complex background, most visual hand gesture recognition systems work only in a restricted environment. An adaptive skin color model based on face detection is utilized to detect skin color regions like hands. The user can interact with the virtual environment using hand gestures. The hand gestures are predicted by the machine. The gestures are predicted and result will be displayed. No Special Hardware is Necessary, except for the Camera.

2. LITERATURE SURVEY

S.NO	Authors	Title	Publishing	Techniques & dataset	Pros
1.	Kevin Simon	How does AI recognize your hand gestures and movements	2019 MANTRA.AI	Static Gesture Recognition DataSet	1. We can look forward to gestural recognition applications moving into education, real estate, fashion design, and even law enforcement. 2. The consumer electronics market is set to lead the forecasted demand and growth for this technology, leading to an increase in the number of vendors creating ground-breaking applications in this space.
2.	Yacine BENAFFANE	[Deep Learning] Hand gesture recognition	Jan 14, 2020	Dataset hand gesture EgoGesture	1. Gesture recognition is a hot topic in computer vision and pattern recognition. 2. It's an interesting aspect of our society, the signs will be able to help, depending on the case, to establish and improve existing communication.
3	Ahmet Gunduz	Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks	29 Jan 2019	EgoGesture and NVIDIA Dynamic Hand Gesture Datasets	1. An acceptable classification accuracy 2. Fast reaction time 3. Resource efficiency 4. Single-time activation per each performed gesture.
4.	Viraj Shinde, Tushar	Hand Gesture Recognition	1, January - 2014	Object detection,	1.Reduce external Interface:

	Bacchav, Jitendra Pawar, Mangesh Sanap	System Using Camera		Object tracking, Object recognition	The Advantage of System is to Reduce External Interface like Mouse And Keyboard. 2.High Portability : The proposed System reduce the working of external interface like keyboard and mouse so it makes it high portable.
5.	Alexandre Campeau-Lecours	Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning	10 Jan 2018	MyoArmbandDataset	Artificial intelligence can be leveraged to increase the autonomy of people living with disabilities.
6.	Abdullah Mujahid , Mazhar Javed Awan , Awais Yasin	Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model	2 May 2021	DarkNet-53	In this paper, we have proposed a lightweight model based on the YOLOv3 and DarkNet-53 deep learning models for hand gesture recognition. For future work, we will be focusing on hybrid methods with smart mobile applications or robotics with different scenarios.
7.	Adithya V, Rajesh R	A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognition	4 June 2020	Convo-lutional neural network (CNN)	The proposed CNN architecture eliminates the need of detection and segmentation of hands from the captured images, thus reducing the computational burden faced during hand posture recognition with classical approaches.

8.	Peng Liu, Xiangxiang Li	Hand Gesture Recognition Based on Single-Shot Multibox Detector Deep Learning	30 Dec 2019	CNN	We chose four character’s hand gestures under three different complex backgrounds as the investigated objects. The research results show that the SSD algorithm can be used in the hand gesture recognition system for the human-computer interaction application.
9.	Xiaoguang Yu, Yafei Yuan	Hand Gesture Recognition Based on Faster-RCNN Deep Learning	October 31, 2018	regional convolutional neural network (Faster-RCNN)	In summary, the faster region full convolution neural network (Faster-RCNN) depth learning algorithm is proposed to apply for gesture recognition in this work. We chose five characters hand gesture under three different complex backgrounds as the investigated objects.
10.	Manju Khari, Aditya Kumar Garg1	Gesture Recognition of RGB and RGB-D Static Images Using Convolutional Neural Networks	16 September 2019	CNN	CNN is currently a powerful artificial intelligence tool that can recognize patterns with high accuracy. The proposed model is tested on ASL dataset and the recognition rate attained is 94.8%

3. SYSTEM DESIGN:

3.1. HARDWARE DESIGN:

Processor : For Intel i5 Gen 9 or higher
For AMD Ryzen 3
Os: Windows 7 or higher versions
Ram: 4GB ram
Hard Disk: 500GB
Camera: Webcam

3.2. SOFTWARE REQUIREMENTS:

Python 3.9
Opencv
Numpy package
Keras package
Matplotlib package
Visual studio code
Jupyter notebook

4. METHODOLOGY

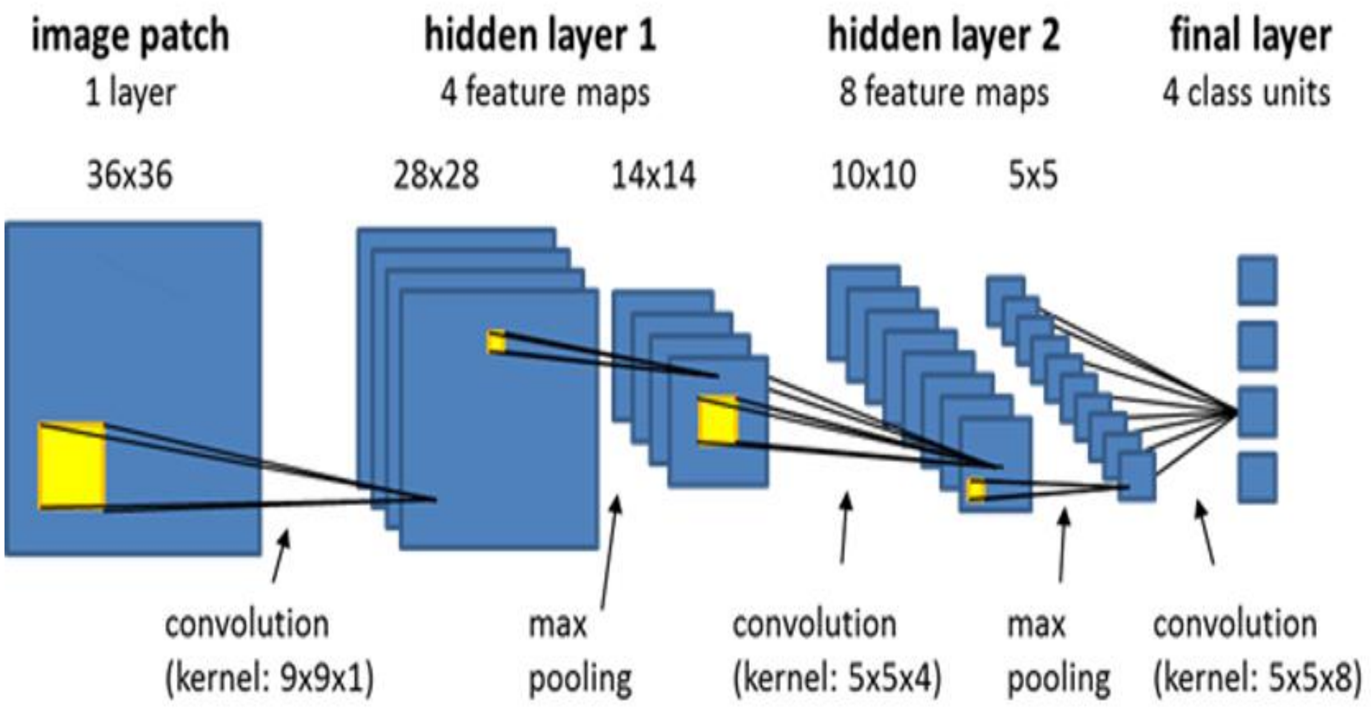
4.1 CNN model

Convolutional neural networks (CNN) – the concept behind recent breakthroughs and developments in deep learning.CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular. These convolutional neural network models are ubiquitous in the image data space. They work phenomenally well on computer vision tasks like image classification, object detection, image recognition, etc.
CNN is basically used for image analysis it detects patterns and makes sense of them, this pattern detection is useful for image analysis. The word convolutional refers to the filtering process that happens in the network. If there is a complex image the convolutional neural network simplifies it so it can be better processed and understood. The CNN or convolutional neural networks are the most commonly used algorithms for image classification problems. An image classifier takes a photograph or video as an input and classifies it into one of the possible categories that it was trained to identify. They have applications in various fields like driver less cars, defense, healthcare etc.
Using CNN is like watching an image through window which allows you to see specific features which you might not be able to see otherwise.

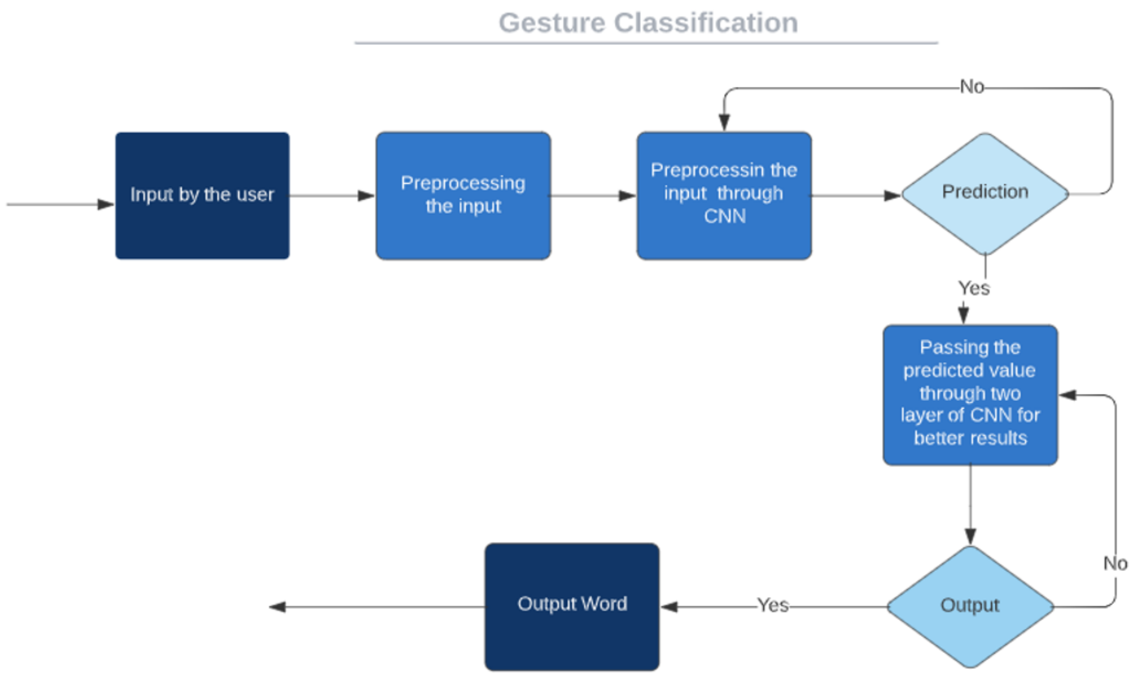
Layers in CNN Model

- Image Patch (1 Layer)
- Hidden Layer 1 (4 feature maps)
- Hidden Layer 2 (8 feature maps)
- Final Layer (Class units)

CNN Layer Diagram



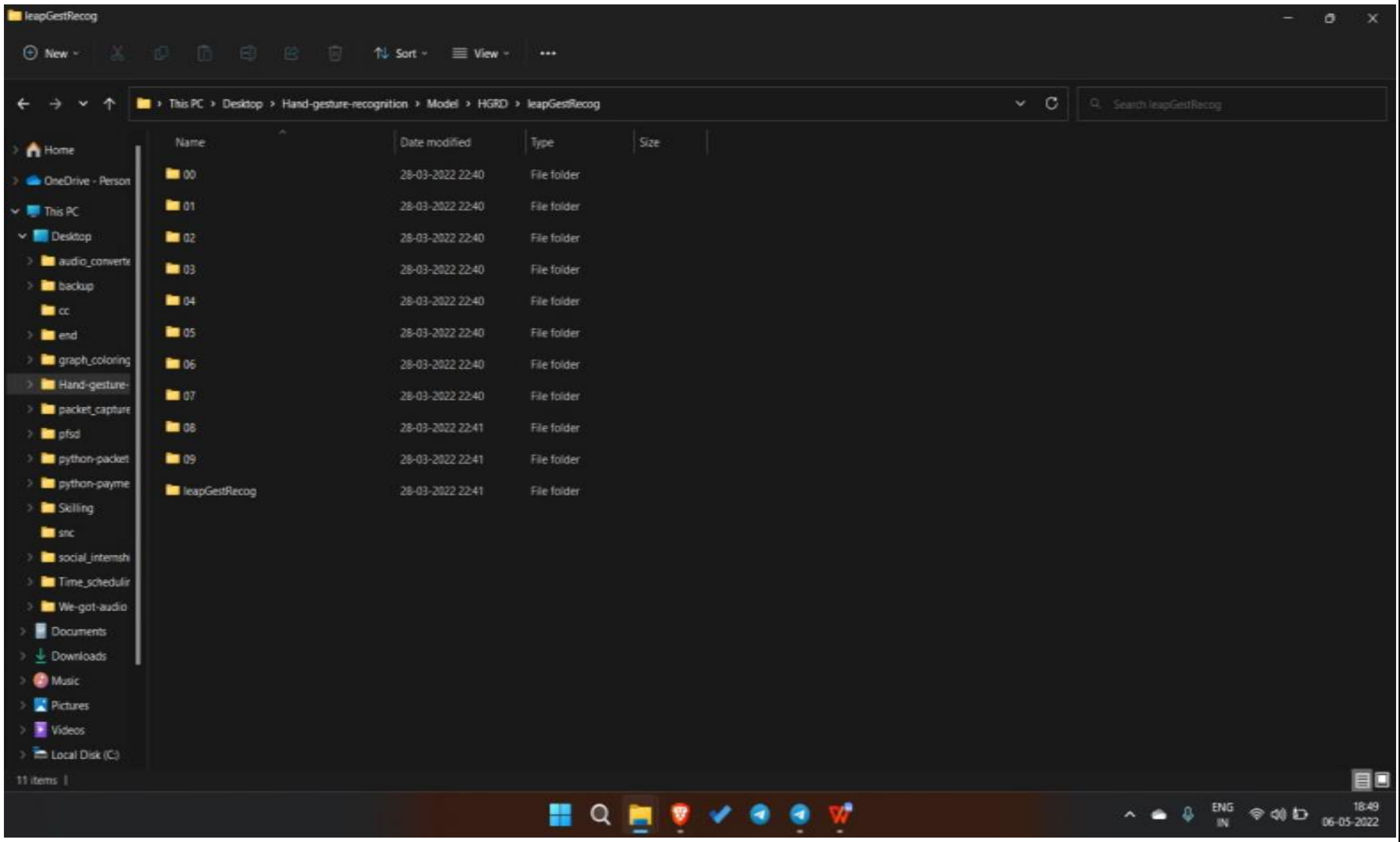
Flow chart for Hand gesture recognition


Gesture Classification
Input by the user
Preprocessing the input
Preprocessin the input through CNN
Prediction
No
Yes
Passing the predicted value through two layer of CNN for better results
Output
No
Yes
Output Word

5. DEVELOPMENT

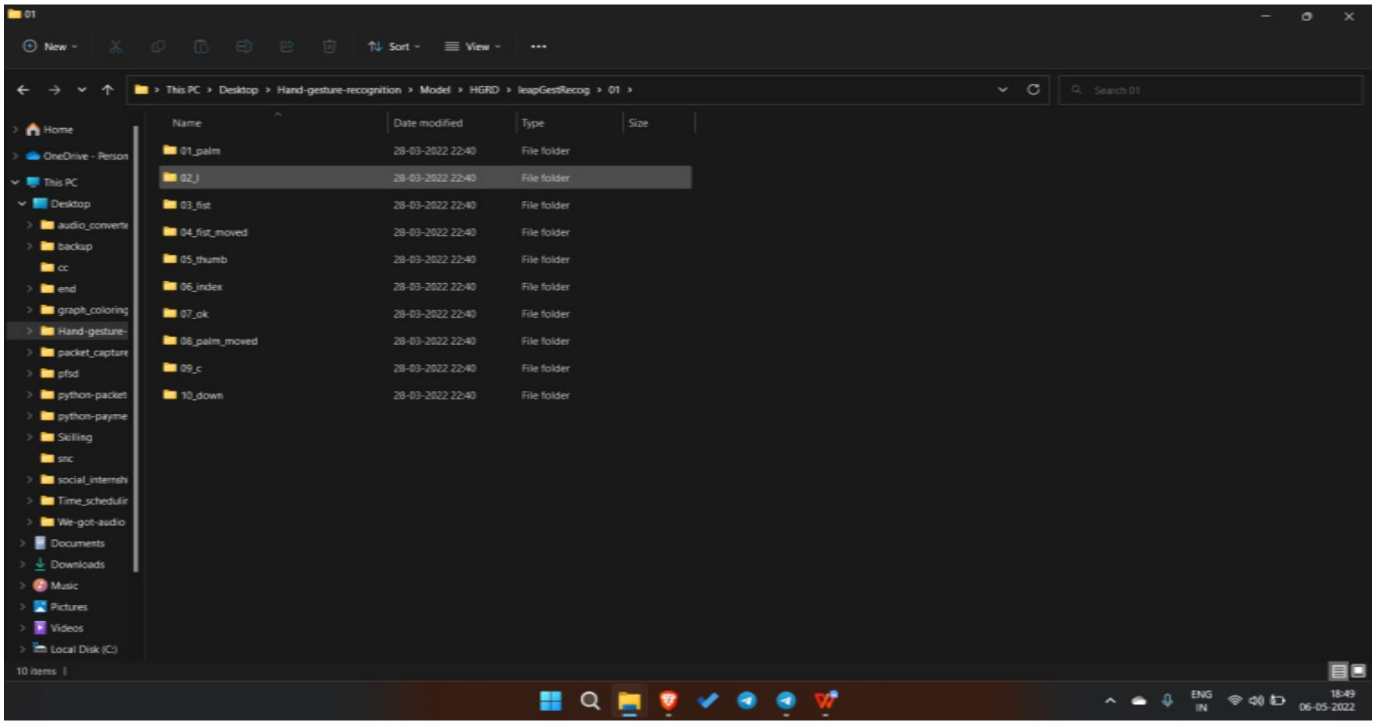
5.1. DATASET

In our project, we are using Hand Gesture Recognition Dataset with 20 Different Gestures with a total of 24000 images. For training purposes, there are 900 images in each directory and for testing purposes, there are 300 images in each directory. This dataset is primarily used for hand gesture recognition tasks.



Name	Date modified	Type	Size
00	28-03-2022 22:40	File folder	
01	28-03-2022 22:40	File folder	
02	28-03-2022 22:40	File folder	
03	28-03-2022 22:40	File folder	
04	28-03-2022 22:40	File folder	
05	28-03-2022 22:40	File folder	
06	28-03-2022 22:40	File folder	
07	28-03-2022 22:40	File folder	
08	28-03-2022 22:41	File folder	
09	28-03-2022 22:41	File folder	
leapGestRecog	28-03-2022 22:41	File folder	

Fig 1&2 - Hand gesture dataset organized folders

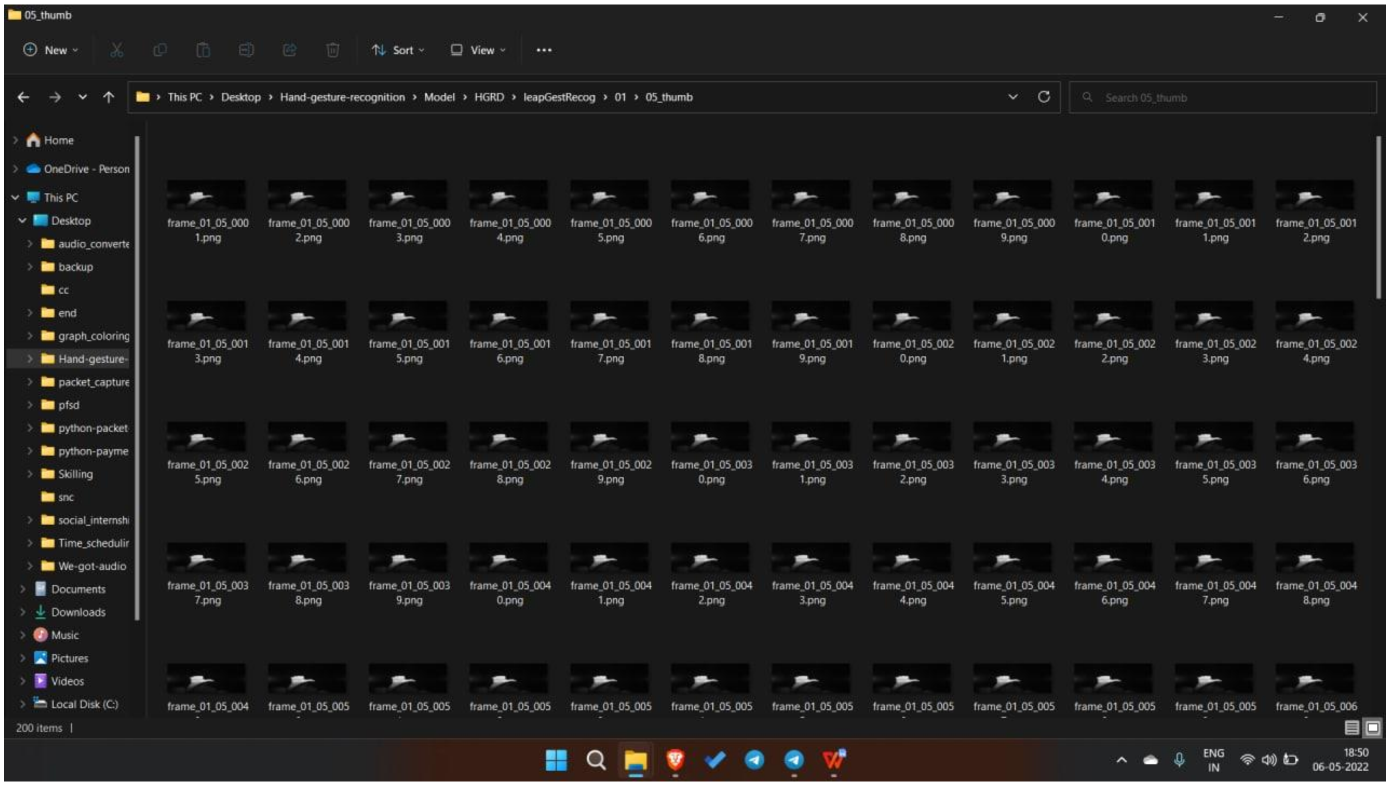


This PC > Desktop > Hand-gesture-recognition > Model > HGRD > leapGestRecog > 01

Name	Date modified	Type	Size
01_palm	28-03-2022 22:40	File folder	
02_l	28-03-2022 22:40	File folder	
03_fist	28-03-2022 22:40	File folder	
04_fist_moved	28-03-2022 22:40	File folder	
05_thumb	28-03-2022 22:40	File folder	
06_index	28-03-2022 22:40	File folder	
07_ok	28-03-2022 22:40	File folder	
08_palm_moved	28-03-2022 22:40	File folder	
09_c	28-03-2022 22:40	File folder	
10_down	28-03-2022 22:40	File folder	

10 items

Fig 2



This PC > Desktop > Hand-gesture-recognition > Model > HGRD > leapGestRecog > 01 > 05_thumb

frame_01_05_0001.png frame_01_05_0002.png frame_01_05_0003.png frame_01_05_0004.png frame_01_05_0005.png frame_01_05_0006.png frame_01_05_0007.png frame_01_05_0008.png frame_01_05_0009.png frame_01_05_0010.png frame_01_05_0011.png frame_01_05_0012.png

frame_01_05_0013.png frame_01_05_0014.png frame_01_05_0015.png frame_01_05_0016.png frame_01_05_0017.png frame_01_05_0018.png frame_01_05_0019.png frame_01_05_0020.png frame_01_05_0021.png frame_01_05_0022.png frame_01_05_0023.png frame_01_05_0024.png

frame_01_05_0025.png frame_01_05_0026.png frame_01_05_0027.png frame_01_05_0028.png frame_01_05_0029.png frame_01_05_0030.png frame_01_05_0031.png frame_01_05_0032.png frame_01_05_0033.png frame_01_05_0034.png frame_01_05_0035.png frame_01_05_0036.png

frame_01_05_0037.png frame_01_05_0038.png frame_01_05_0039.png frame_01_05_0040.png frame_01_05_0041.png frame_01_05_0042.png frame_01_05_0043.png frame_01_05_0044.png frame_01_05_0045.png frame_01_05_0046.png frame_01_05_0047.png frame_01_05_0048.png

frame_01_05_004 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_005 frame_01_05_006

200 items

Fig 3: Gesture images of thumb sorted into one folder

4.2 Adam optimizer algorithm

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.Adam is different to classical stochastic gradient descent.

Stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training.

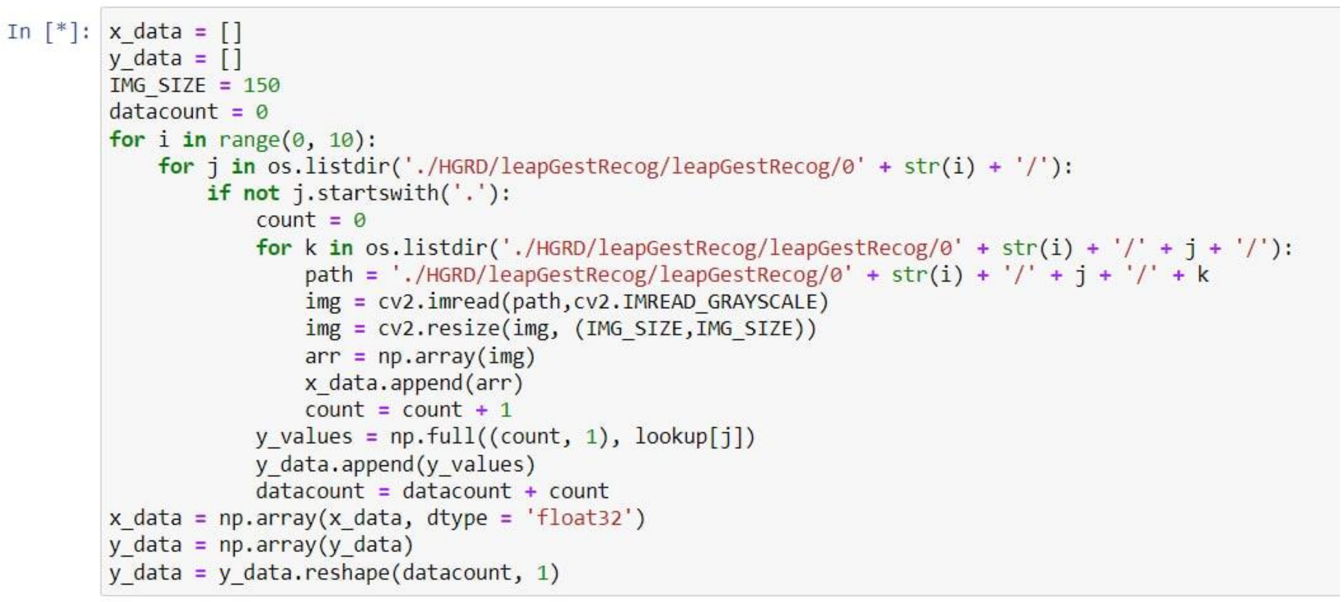
A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

The advantages of two other extensions of stochastic gradient descent. Specifically:

- **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).
- **Root Mean Square Propagation** (RMSProp) also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

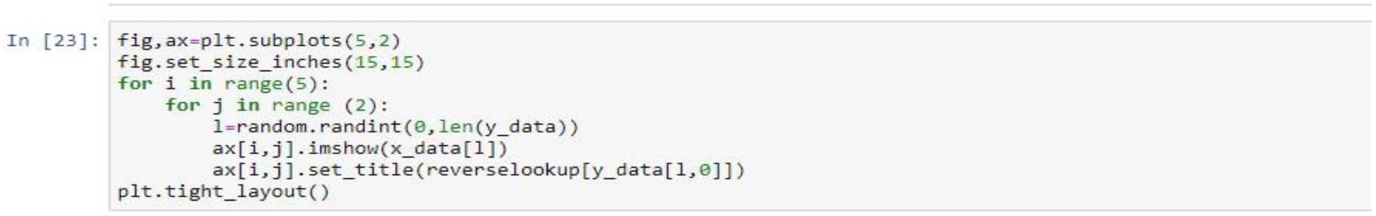
Adam is a popular algorithm in the field of deep learning because it achieves good results fast.

5.2.2. Resizing the Data

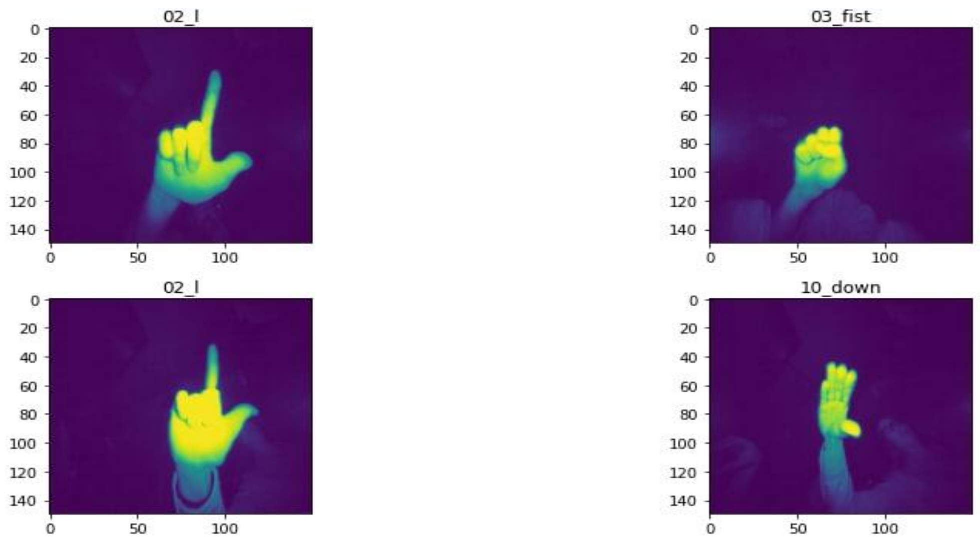


```
In [*]: x_data = []
        y_data = []
        IMG_SIZE = 150
        datacount = 0
        for i in range(0, 10):
            for j in os.listdir('./HGRD/leapGestRecog/leapGestRecog/0' + str(i) + '/'):
                if not j.startswith('.'):
                    count = 0
                    for k in os.listdir('./HGRD/leapGestRecog/leapGestRecog/0' + str(i) + '/' + j + '/'):
                        path = './HGRD/leapGestRecog/leapGestRecog/0' + str(i) + '/' + j + '/' + k
                        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
                        arr = np.array(img)
                        x_data.append(arr)
                        count = count + 1
                    y_values = np.full((count, 1), lookup[j])
                    y_data.append(y_values)
                    datacount = datacount + count
        x_data = np.array(x_data, dtype = 'float32')
        y_data = np.array(y_data)
        y_data = y_data.reshape(datacount, 1)
```

5.2.3. Data cleaning and data processing



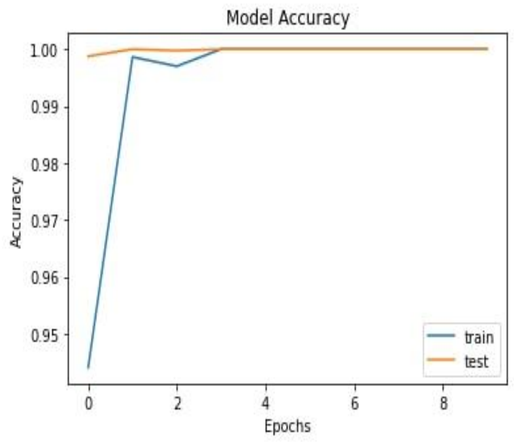
```
In [23]: fig,ax=plt.subplots(5,2)
         fig.set_size_inches(15,15)
         for i in range(5):
             for j in range (2):
                 l=random.randint(0,len(y_data))
                 ax[i,j].imshow(x_data[l])
                 ax[i,j].set_title(reverselookup[y_data[l,0]])
         plt.tight_layout()
```



5.2.6. Checking model accuracy

Check Model Accuracy

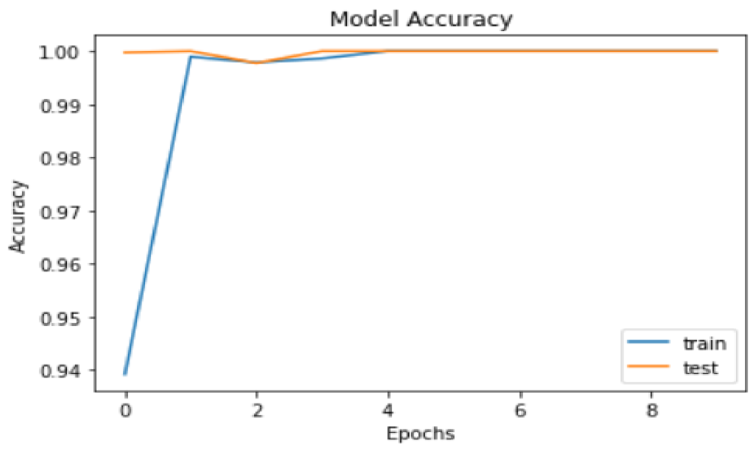
```
In [43]: plt.plot(History.history['accuracy'])
         plt.plot(History.history['val_accuracy'])
         plt.title('Model Accuracy')
         plt.ylabel('Accuracy')
         plt.xlabel('Epochs')
         plt.legend(['train', 'test'])
         plt.show()
```



5.2.7. Checking model Loss

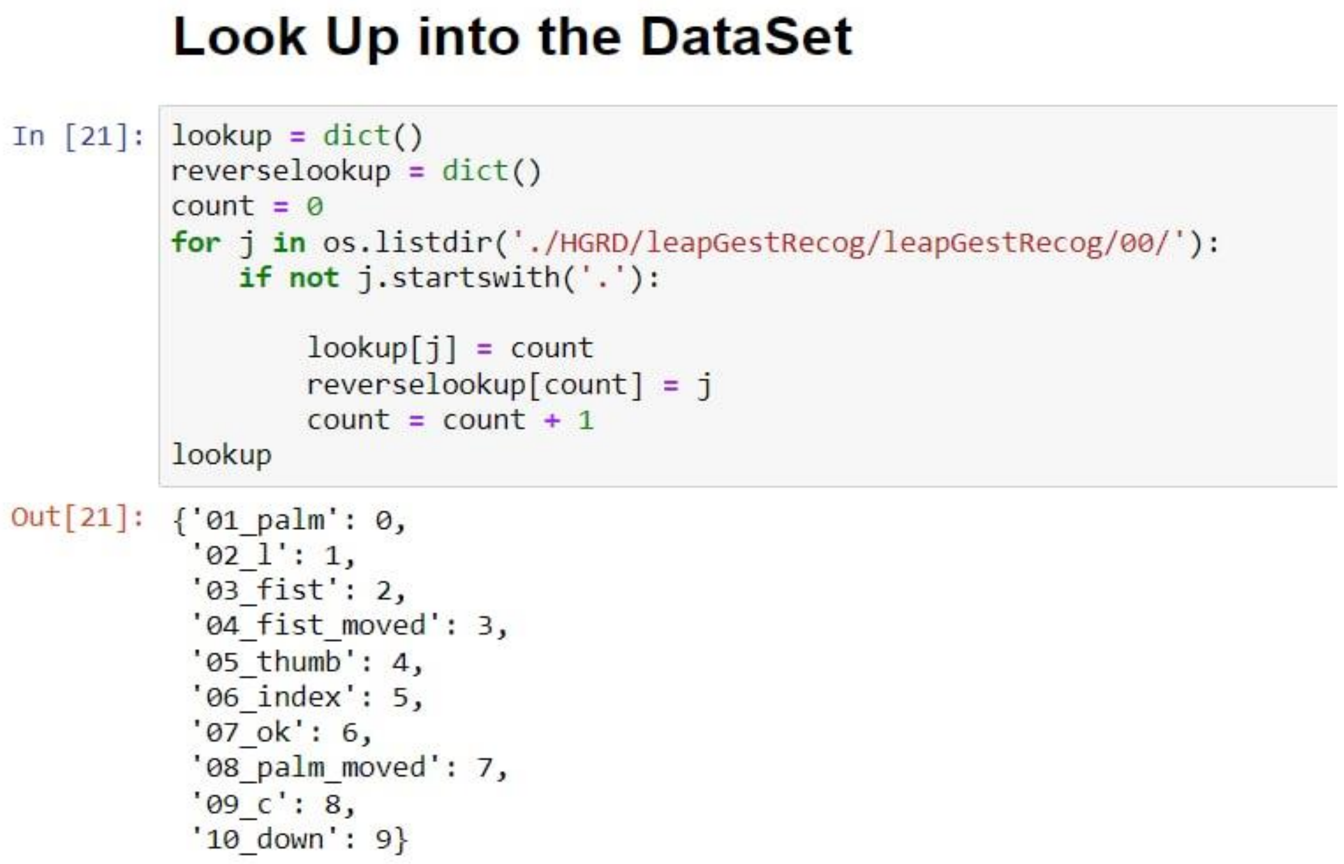
Check Model Accuracy

```
In [28]: plt.plot(History.history['accuracy'])
         plt.plot(History.history['val_accuracy'])
         plt.title('Model Accuracy')
         plt.ylabel('Accuracy')
         plt.xlabel('Epochs')
         plt.legend(['train', 'test'])
         plt.show()
```



5.2. IMPLEMENTATION

5.2.1. DataSet lookup

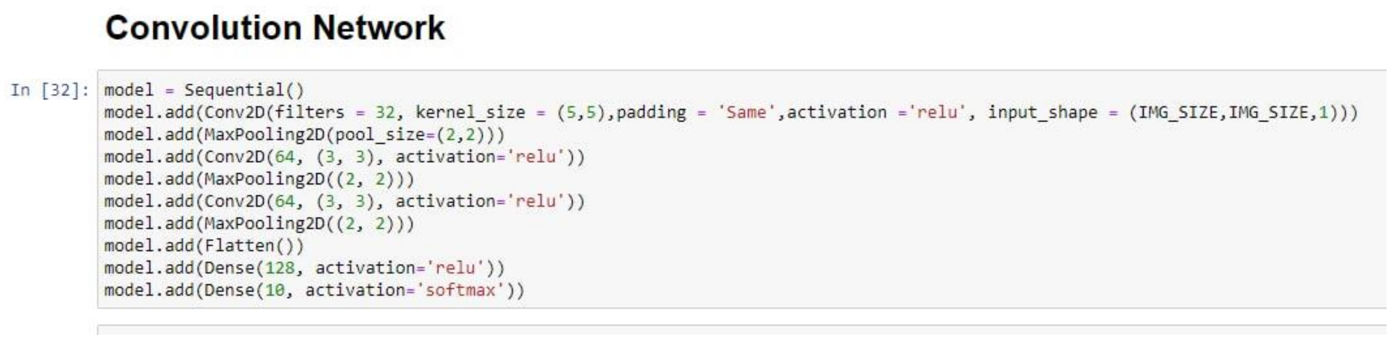


Look Up into the DataSet

```
In [21]: lookup = dict()
         reverselookup = dict()
         count = 0
         for j in os.listdir('./HGRD/leapGestRecog/leapGestRecog/00/'):
             if not j.startswith('.'):

                 lookup[j] = count
                 reverselookup[count] = j
                 count = count + 1
         lookup
```

```
Out[21]: {'01_palm': 0,
          '02_l': 1,
          '03_fist': 2,
          '04_fist_moved': 3,
          '05_thumb': 4,
          '06_index': 5,
          '07_ok': 6,
          '08_palm_moved': 7,
          '09_c': 8,
          '10_down': 9}
```

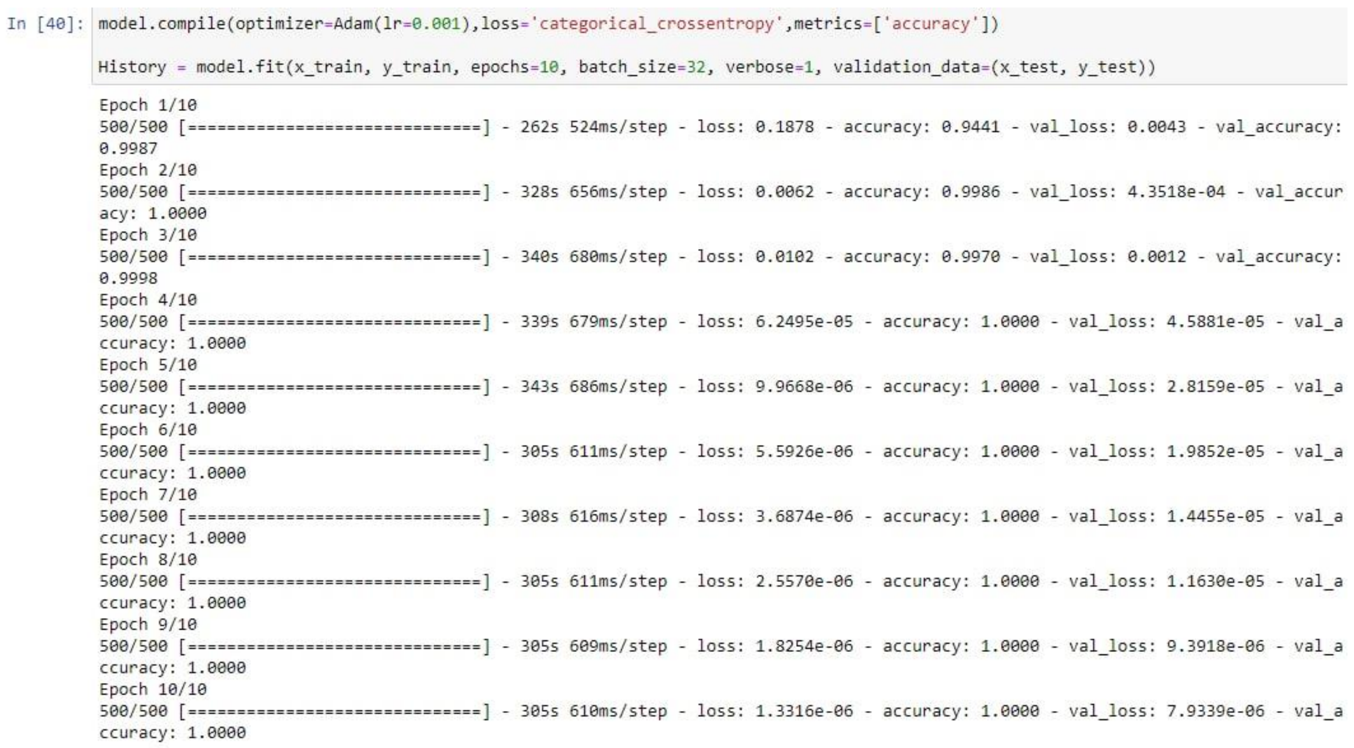
5.2.4. Neural Network



Convolution Network

```
In [32]: model = Sequential()
         model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (IMG_SIZE,IMG_SIZE,1)))
         model.add(MaxPooling2D(pool_size=(2,2)))
         model.add(Conv2D(64, (3, 3), activation='relu'))
         model.add(MaxPooling2D((2, 2)))
         model.add(Conv2D(64, (3, 3), activation='relu'))
         model.add(MaxPooling2D((2, 2)))
         model.add(Flatten())
         model.add(Dense(128, activation='relu'))
         model.add(Dense(10, activation='softmax'))
```

5.2.5. Data Training



```
In [40]: model.compile(optimizer=Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])

         History = model.fit(x_train, y_train, epochs=10, batch_size=32, verbose=1, validation_data=(x_test, y_test))

         Epoch 1/10
         500/500 [==============================] - 262s 524ms/step - loss: 0.1878 - accuracy: 0.9441 - val_loss: 0.0043 - val_accuracy:
         0.9987
         Epoch 2/10
         500/500 [==============================] - 328s 656ms/step - loss: 0.0062 - accuracy: 0.9986 - val_loss: 4.3518e-04 - val_accur
         acy: 1.0000
         Epoch 3/10
         500/500 [==============================] - 340s 680ms/step - loss: 0.0102 - accuracy: 0.9970 - val_loss: 0.0012 - val_accuracy:
         0.9998
         Epoch 4/10
         500/500 [==============================] - 339s 679ms/step - loss: 6.2495e-05 - accuracy: 1.0000 - val_loss: 4.5881e-05 - val_a
         ccuracy: 1.0000
         Epoch 5/10
         500/500 [==============================] - 343s 686ms/step - loss: 9.9668e-06 - accuracy: 1.0000 - val_loss: 2.8159e-05 - val_a
         ccuracy: 1.0000
         Epoch 6/10
         500/500 [==============================] - 305s 611ms/step - loss: 5.5926e-06 - accuracy: 1.0000 - val_loss: 1.9852e-05 - val_a
         ccuracy: 1.0000
         Epoch 7/10
         500/500 [==============================] - 308s 616ms/step - loss: 3.6874e-06 - accuracy: 1.0000 - val_loss: 1.4455e-05 - val_a
         ccuracy: 1.0000
         Epoch 8/10
         500/500 [==============================] - 305s 611ms/step - loss: 2.5570e-06 - accuracy: 1.0000 - val_loss: 1.1630e-05 - val_a
         ccuracy: 1.0000
         Epoch 9/10
         500/500 [==============================] - 305s 609ms/step - loss: 1.8254e-06 - accuracy: 1.0000 - val_loss: 9.3918e-06 - val_a
         ccuracy: 1.0000
         Epoch 10/10
         500/500 [==============================] - 305s 610ms/step - loss: 1.3316e-06 - accuracy: 1.0000 - val_loss: 7.9339e-06 - val_a
         ccuracy: 1.0000
```

5.3. SOURCE CODE

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import pandas as pd
from tensorflow import keras
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import Dense, Flatten, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from tqdm import tqdm
import random
from random import shuffle
from zipfile import ZipFile
from PIL import Image
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam

lookup = dict()
reverselookup = dict()
count = 0
for j in os.listdir('./HGRD/leapGestRecog/leapGestRecog/00/'):
    if not j.startswith('.'):

        lookup[j] = count
        reverselookup[count] = j
        count = count + 1
```

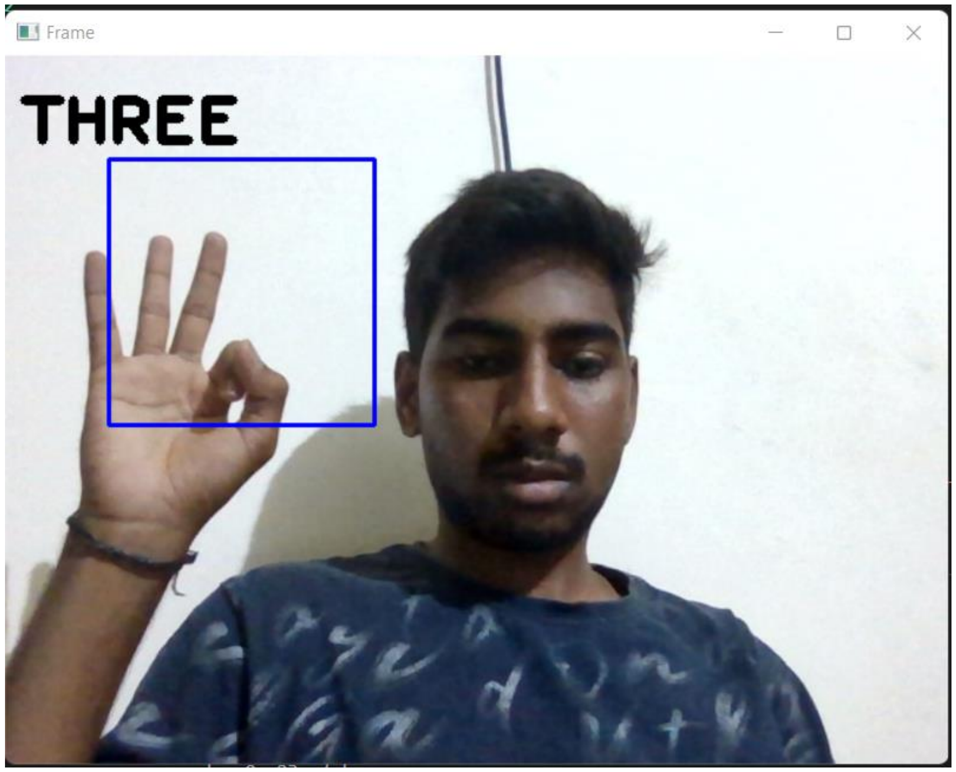
```
lookup
x_data = []
y_data = []
IMG_SIZE = 150
datacount = 0
for i in range(0, 10):
    for j in os.listdir('./HGRD/leapGestRecog/leapGestRecog/0' + str(i) + '/'):
        if not j.startswith('.'):
            count = 0
            for k in os.listdir('./HGRD/leapGestRecog/leapGestRecog/0' + str(i) + '/' + j + '/'):
                path = './HGRD/leapGestRecog/leapGestRecog/0' + str(i) + '/' + j + '/' + k
                img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                if img is None:
                    continue
                img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
                arr = np.array(img)
                x_data.append(arr)
                count = count + 1
            y_values = np.full((count, 1), lookup[j])
            y_data.append(y_values)
            datacount = datacount + count
x_data = np.array(x_data, dtype = 'float32')
y_data = np.array(y_data)
y_data = y_data.reshape(datacount, 1)

fig,ax=plt.subplots(5,2)
fig.set_size_inches(15,15)
for i in range(5):
    for j in range (2):
        l=random.randint(0,len(y_data))
        ax[i,j].imshow(x_data[l])
        ax[i,j].set_title(reverselookup[y_data[l,0]])
```

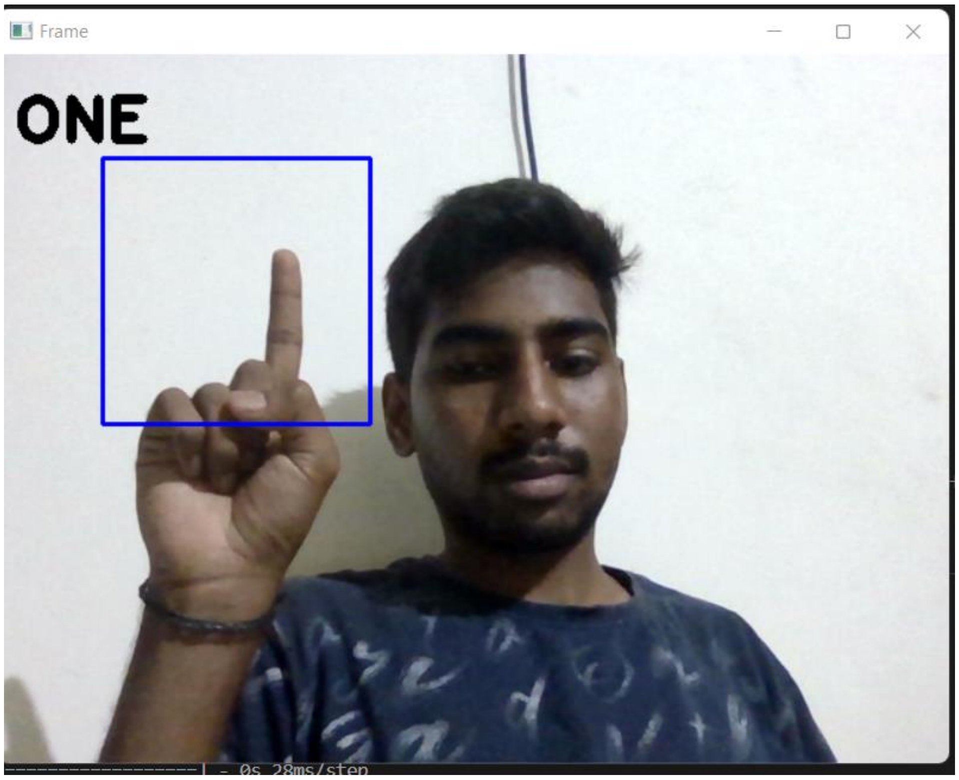
```
plt.tight_layout()
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (IMG_SIZE,IMG_SIZE,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy',metrics=['accuracy')
History = model.fit(x_train, y_train, epochs=10, batch_size=32, verbose=1, validation_data=(x_test, y_test))
model.save('handgesture_model.h5')

plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()

plt.plot(History.history['accuracy'])
plt.plot(History.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```

#Source code for prediction

```
import numpy as np
from keras.models import model_from_json
import operator
import cv2
import sys, os



# Loading the model
json_file = open("model_test_two.json", "r")
model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(model_json)
# load weights into new model
loaded_model.load_weights("model_test_two.h5")
print("Loaded model from disk")


# Category dictionary
categories = { 1: 'ONE', 2: 'TWO', 3: 'THREE', 4: 'FOUR', 5: 'FIVE',6:'SIX',7:'SEVEN',8:'EIGHT',
            9:'NINE'}


vid = cv2.VideoCapture(0)
#

while True:
    sucess, frame = vid.read()
    x1 = int(0.5 * frame.shape[1])
    y1 = 10
    x2 = frame.shape[1] - 10
    y2 = int(0.5 * frame.shape[1])
    # Drawing the ROI
    # The increment/decrement by 1 is to compensate for the bounding box
    cv2.rectangle(frame, (70,70), (250,250), (255, 0, 0), 2)
    # Extracting the ROI
    roi = frame[70:250, 70:250]
    #    roi = cv2.resize(roi, (64, 64))
    #roi = cv2.resize(roi, (64, 64))
    cv2.imshow("Frame", frame)
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

    blur = cv2.GaussianBlur(gray, (5, 5), 2)
    # #blur = cv2.bilateralFilter(roi,9,75,75)

    th3 = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY_INV, 11, 2)
    ret, test_image = cv2.threshold(th3, 70, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
    # time.sleep(5)
    # cv2.imwrite("/home/rc/Downloads/soe/im1.jpg", roi)
    # test_image = func("/home/rc/Downloads/soe/im1.jpg")

    test_image = cv2.resize(test_image, (150, 150))



    cv2.imshow("test", test_image)
    # Batch of 1
    result = loaded_model.predict(test_image.reshape(1, 150, 150, 1))
    prediction = {
        'ONE': result[0][0],
        'TWO': result[0][1],
        'THREE': result[0][2],
        'FOUR': result[0][3],
        'FIVE': result[0][4],
        'SIX': result[0][5],
        'SEVEN': result[0][6],
        'EIGHT': result[0][7],
        'NINE': result[0][8]}
    # Sorting based on top prediction
    prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)

    # Displaying the predictions
    cv2.putText(frame, prediction[0][0], (10, 60), cv2.FONT_HERSHEY_PLAIN, 3, (0, 0, 0), 5)
    cv2.imshow("Frame", frame)

    interrupt = cv2.waitKey(10)
    if interrupt & 0xFF == 27:  # esc key
        break

vid.release()
cv2.destroyAllWindows()
```

6. OUTPUT :





```
- 0s 28ms/step
```

7. CONCLUSION :

We have developed an output while predicting the gestures of persons .We have used the Hand gesture recognition detection dataset from Kaggle and did data cleaning from that dataset .Based on the results, a computer vision application could detect and recognize simple hand gestures for robot navigation using simple heuristic rules. While the use of moment invariants was not considered suitable because the same gestures could be used pointing in opposite directions, other learning algorithms like AdaBoost could be explored to make the program more robust and less affected by extraneous objects and noise.

In conclusion, Gesture recognition involves techniques from computer vision. The importance of gesture recognition lies in building efficient human-machine interaction.
CNN is a multi-layer neural network which is one of the deep learning techniques used efficiently in the field of gesture recognition. The user can interact with the machine using hand gestures more easily and quickly. No Special Hardware is Necessary, except for the Camera. Gesture recognition promises wide-ranging applications in the field. In the future, this project can be done by keeping track of all the various inputs such as hand gestures and giving cumulative results.

The system could also be made smart to be trained for only one or two gestures rather than all and then made ready for testing. This will require only a few changes in the current interface code, which were not performed due to the shortage of time.
One time training constraint for real time system can be removed if the algorithm is made efficient to work with all skin types and light conditions which seems impossible by now altogether. Framing with COG (Centre of gravity) to control orientation factor could make this system more perfect for real application.

BIBILOGRAPHY

Mujahid, Abdullah, et al. "Real-time hand gesture recognition based on deep learning YOLOv3 model." Applied Sciences 11.9 (2021): 4164.

Mujahid, A., Awan, M. J., Yasin, A., Mohammed, M. A., Damaševičius, R., Maskeliūnas, R., & Abdulkareem, K. H. (2021). Real-time hand gesture recognition based on deep learning YOLOv3 model. Applied Sciences, 11(9), 4164.

Mujahid, Abdullah, Mazhar Javed Awan, Awais Yasin, Mazin Abed Mohammed, Robertas Damaševičius, Rytis Maskeliūnas, and Karrar Hameed Abdulkareem. "Real-time hand gesture recognition based on deep learning YOLOv3 model." Applied Sciences 11, no. 9 (2021): 4164.

Mujahid, A., Awan, M.J., Yasin, A., Mohammed, M.A., Damaševičius, R., Maskeliūnas, R. and Abdulkareem, K.H., 2021. Real-time hand gesture recognition based on deep learning YOLOv3 model. Applied Sciences, 11(9), p.4164.

Mujahid A, Awan MJ, Yasin A, Mohammed MA, Damaševičius R, Maskeliūnas R, Abdulkareem KH. Real-time hand gesture recognition based on deep learning YOLOv3 model. Applied Sciences. 2021 Jan;11(9):4164.