VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRICAL ENGINEERING

CNN BASED SMART WHEELCHAIR

Done by

Harsh Trivedi-19BEE0014
Manya Sahu-19BEI0007
Divyansh Singh Raghav-19BEE0067

For the course
Course Code: EEE1008
Course Name: BIOMEDICAL INSTRUMENTATION

Semester: Winter 2021-22

Content

ABSTRACT

Nowadays, technology in medical field related to IoT has advanced and evolved in a tremendous amount.One of the applications is smart wheelchair, which has further many classifications.Smart wheelchair has achieved a lot of interest in the current era.User can interact with the wheelchair more efficiently.A hand gestured control bot is designed using CNN algorithm which has the capability to predict 6 hand gestures.The demonstration of the smart wheelchair will be done using a smart IoT bot.The hand gesture is recognized using raspberry pi-cam.Moreover,the movement of the bot is controlled with the help of raspberrypi which is a microprocessor.From the literature survey we also concluded that accuracy and efficiency of the CNN based wheelchairs is better than the accelerometer based wheelchairs.The proposed project will have a significant effect on society because of its user-friendly nature and cheaper price compared to other automated wheelchair designs.

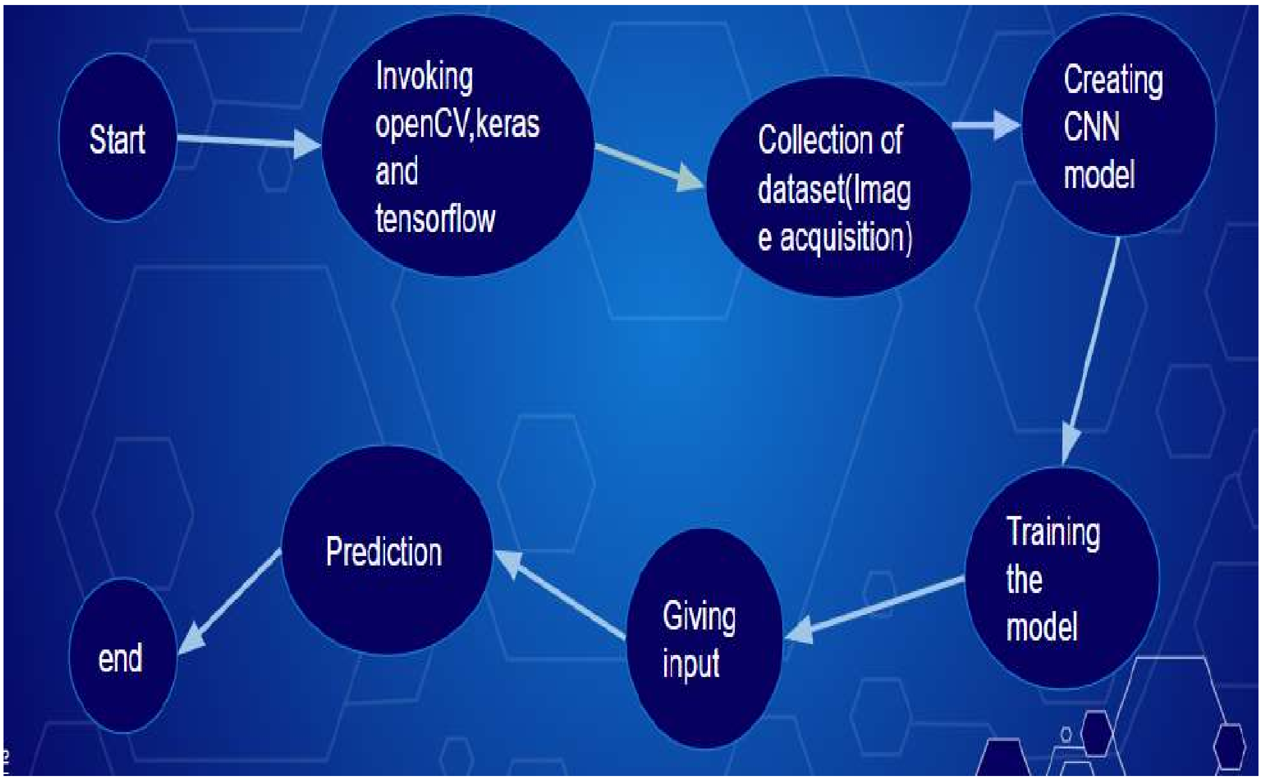
INTRODUCTION

Specially abled people face tremendous amounts of issues in day-to-day life and to make their life easier we have to come up with a smart device which will have capability to make the movement of the person easier and without any dependency on any other person.To break through some of these barriers, and a variety of technologies have been invented (and are constantly being developed) to assist and make the life of a specially abled person easier.Through the availability of linked technology, the Internet of Things can bring up novel options for persons with disabilities when used effectively. We can further – this project by embedding it with Smart Wheel Chair ie. it can be designed to have selfmobility with the help of the user command.We have to make the dataset on our own and train it using machine learning.It will be gesture controlled so only with the wrist/hand movements the device can be controlled easily ie.it is completely user friendly.

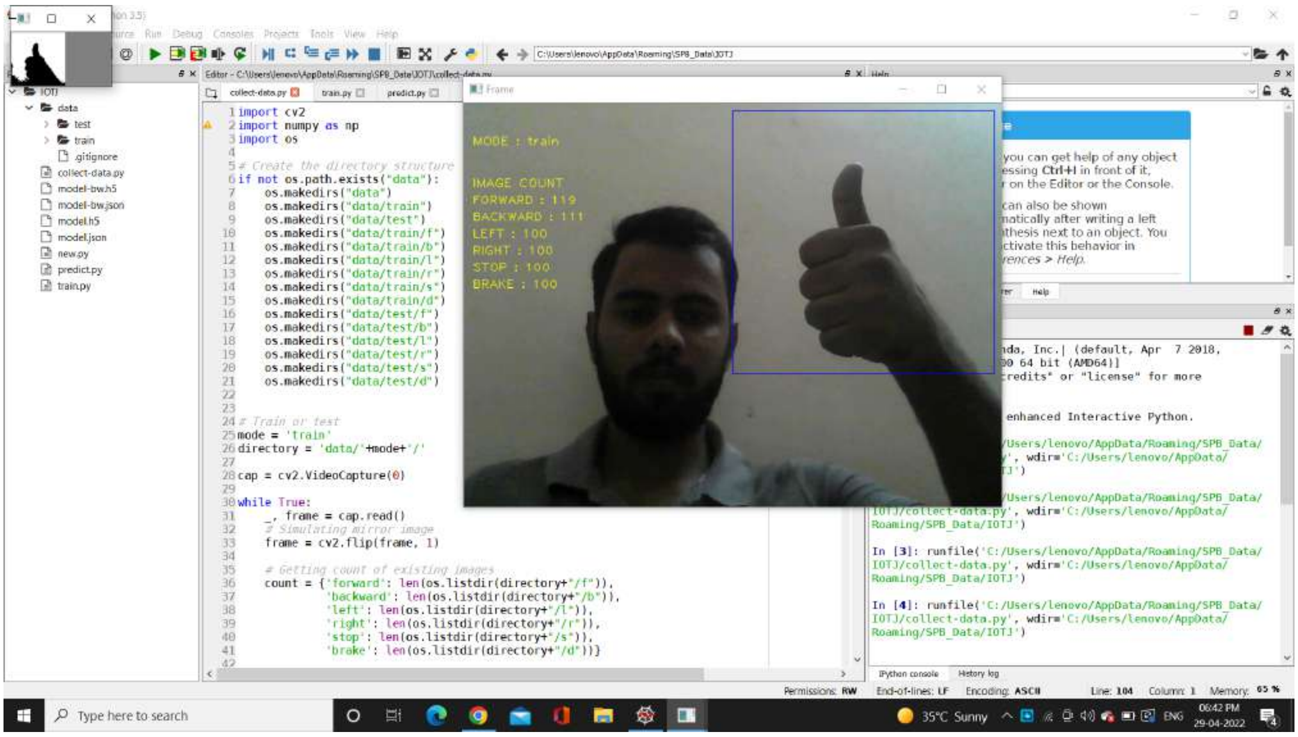
PROBLEM STATEMENT(OBJECTIVE)

We have to come up with a smart device which will have capability to make the movement of the person easier and without any dependency on any other person.Smart wheelchair should be of low cost.The smart wheelchair which we plan to design will be controlled using hand gestures and the demonstration will be carried out which will be done with the help of a small bot.

BLOCK DIAGRAM


Start
Invoking openCV,keras and tensorflow
Collection of dataset(Image acquisition)
Creating CNN model
Training the model
Giving input
Prediction
end

HAND GESTURES(INPUT IMAGES)



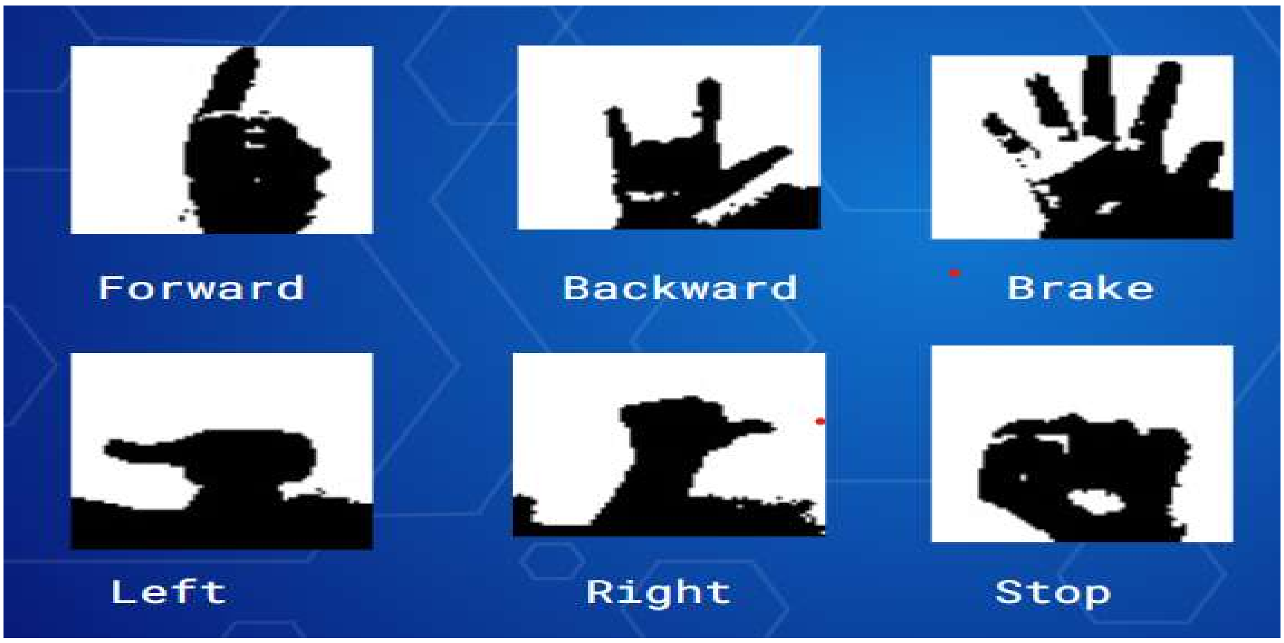
```
import cv2
import numpy as np
import os

# Create the directory structure
if not os.path.exists("data"):
    os.makedirs("data")
    os.makedirs("data/train")
    os.makedirs("data/test")
    os.makedirs("data/train/f")
    os.makedirs("data/train/b")
    os.makedirs("data/train/l")
    os.makedirs("data/train/r")
    os.makedirs("data/train/s")
    os.makedirs("data/train/d")
    os.makedirs("data/test/f")
    os.makedirs("data/test/b")
    os.makedirs("data/test/l")
    os.makedirs("data/test/r")
    os.makedirs("data/test/s")
    os.makedirs("data/test/d")


# Train or test
mode = 'train'
directory = 'data/'+mode+'/'

cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    # Simulating mirror image
    frame = cv2.flip(frame, 1)

    # Getting count of existing images
    count = {'forward': len(os.listdir(directory+"/f")),
             'backward': len(os.listdir(directory+"/b")),
             'left': len(os.listdir(directory+"/l")),
             'right': len(os.listdir(directory+"/r")),
             'stop': len(os.listdir(directory+"/s")),
             'brake': len(os.listdir(directory+"/d"))}
```

```
MODE : train

IMAGE COUNT
FORWARD : 119
BACKWARD : 111
LEFT : 100
RIGHT : 100
STOP : 100
BRAKE : 100
```

```
nda, Inc.| (default, Apr  7 2018,
00 64 bit (AMD64)]
credits" or "license" for more

 enhanced Interactive Python.

/Users/lenovo/AppData/Roaming/SPB_Data/
y', wdir='C:/Users/lenovo/AppData/
TJ')

/Users/lenovo/AppData/Roaming/SPB_Data/
IOTJ/collect-data.py', wdir='C:/Users/lenovo/AppData/
Roaming/SPB_Data/IOTJ')

In [3]: runfile('C:/Users/lenovo/AppData/Roaming/SPB_Data/
IOTJ/collect-data.py', wdir='C:/Users/lenovo/AppData/
Roaming/SPB_Data/IOTJ')

In [4]: runfile('C:/Users/lenovo/AppData/Roaming/SPB_Data/
IOTJ/collect-data.py', wdir='C:/Users/lenovo/AppData/
Roaming/SPB_Data/IOTJ')
```

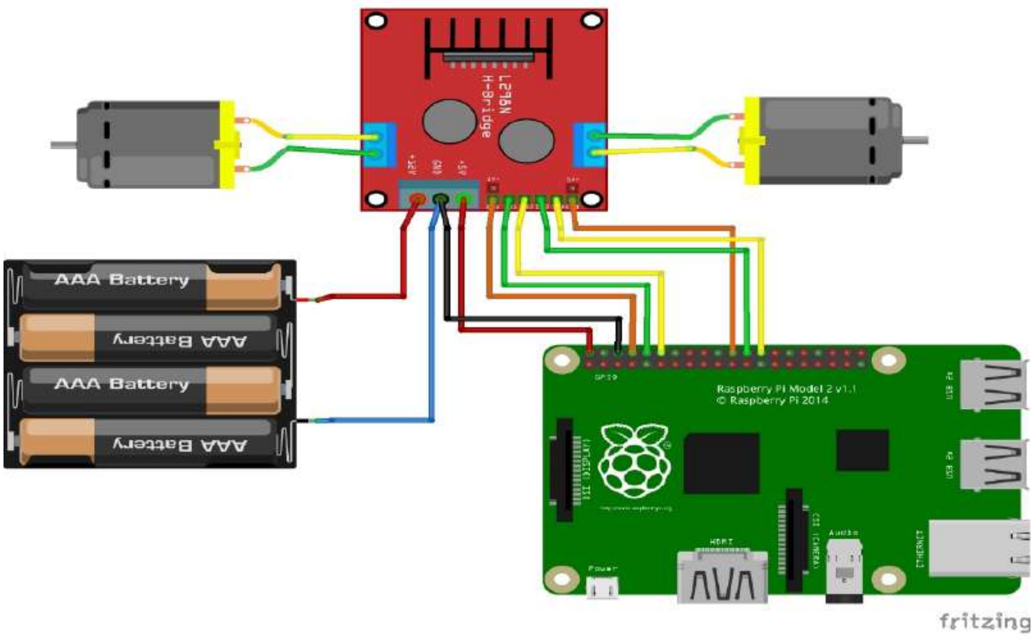
We collected the dataset for 6 hand gestures for controlling the wheelchair
Procedure:

- **We imported the library required for image capturing such as opencv.**
- **We created the ROI and captured the image using interrupt from keyboard and stored it in the directory.**
- **We captured 100 images per gesture.**



These are the six hand gestures we collected for training and testing of our model.

CIRCUIT DESIGN AND EXPLANATION

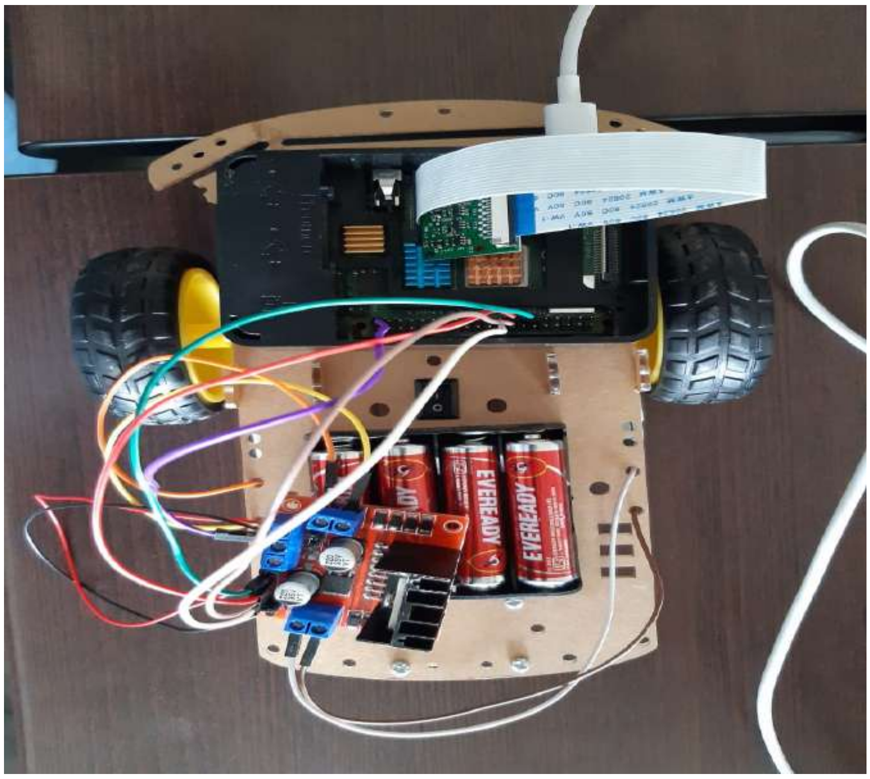


L298 is known as a dual bidirectional motor driver which is based on dual H-Bridge Motor driver IC. This circuit allows you to control two DC motors independently in either direction.

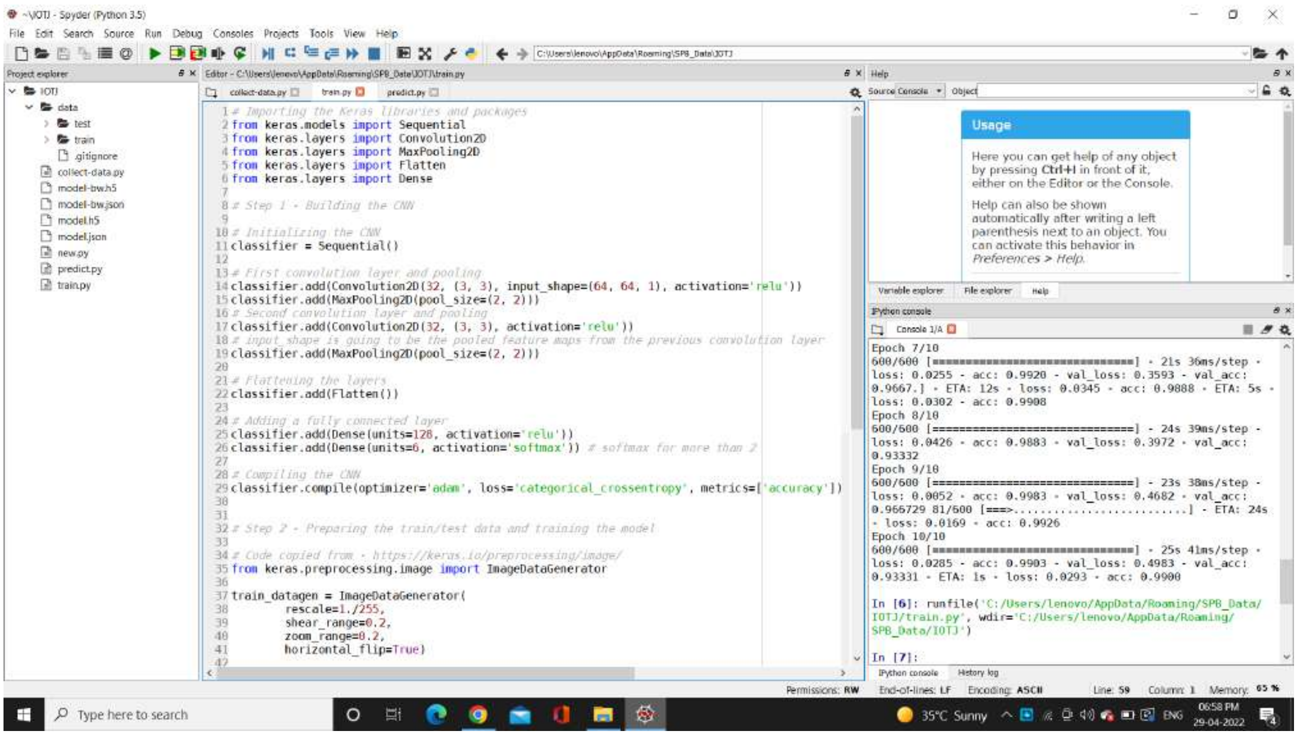
It is a commonly used component for prototypes and hobbyist projects, as it is easy to use and interface the L298 with a Raspberry Pi or an Arduino. Other than its minimal design, it also provides an onboard 5V regulator that you can use to power your 5V circuits very conveniently.

The code starts from importing required libraries for the L298 which include the system, GPIO, and time. It's good to notice that the GPIO is to provide high and low logic to the L298 and the time library is used to add delays between different actions.

Hardware Design and Software Design



- **We imported the keras libraries and packages**
- **Then we builded the CNN model and trained it using the data we collected and tested it.**
- **We created 3 layered CNN network using maxpooling and relu as an activation**



```python
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Step 1 - Building the CNN

# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=6, activation='softmax')) # softmax for more than 2

# Compiling the CNN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


# Step 2 - Preparing the train/test data and training the model

# Code copied from - https://keras.io/preprocessing/image/
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
```

Usage

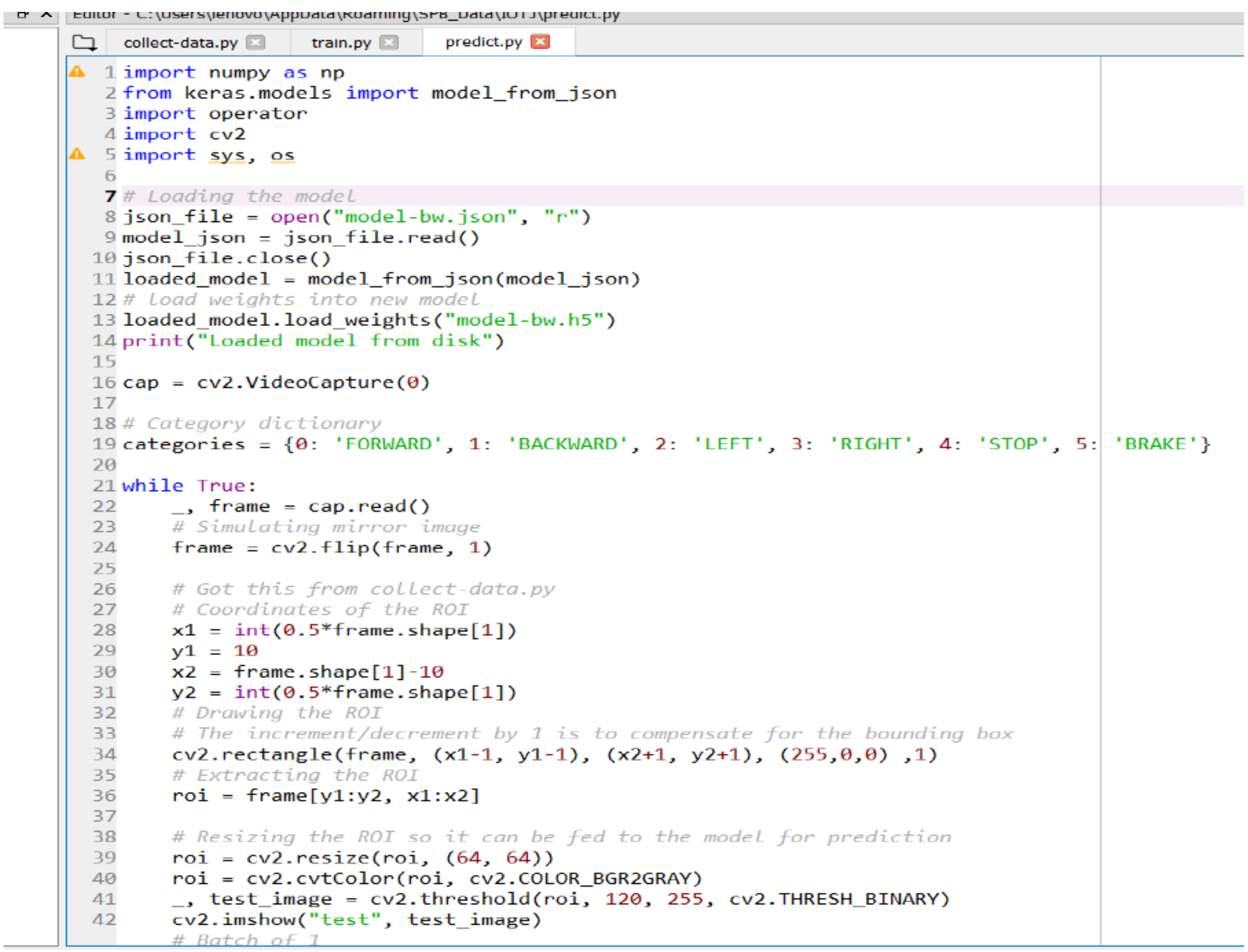
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

```
Epoch 7/10
600/600 [==============================] - 21s 36ms/step -
loss: 0.0255 - acc: 0.9920 - val_loss: 0.3593 - val_acc:
0.9667.] - ETA: 12s - loss: 0.0345 - acc: 0.9888 - ETA: 5s -
loss: 0.0302 - acc: 0.9908
Epoch 8/10
600/600 [==============================] - 24s 39ms/step -
loss: 0.0426 - acc: 0.9883 - val_loss: 0.3972 - val_acc:
0.93332
Epoch 9/10
600/600 [==============================] - 23s 38ms/step -
loss: 0.0052 - acc: 0.9983 - val_loss: 0.4682 - val_acc:
0.966729 81/600 [===>..........................] - ETA: 24s
- loss: 0.0169 - acc: 0.9926
Epoch 10/10
600/600 [==============================] - 25s 41ms/step -
loss: 0.0285 - acc: 0.9903 - val_loss: 0.4983 - val_acc:
0.93331 - ETA: 1s - loss: 0.0293 - acc: 0.9900

In [6]: runfile('C:/Users/lenovo/AppData/Roaming/SPB_Data/
IOTJ/train.py', wdir='C:/Users/lenovo/AppData/Roaming/
SPB_Data/IOTJ')

In [7]:
```

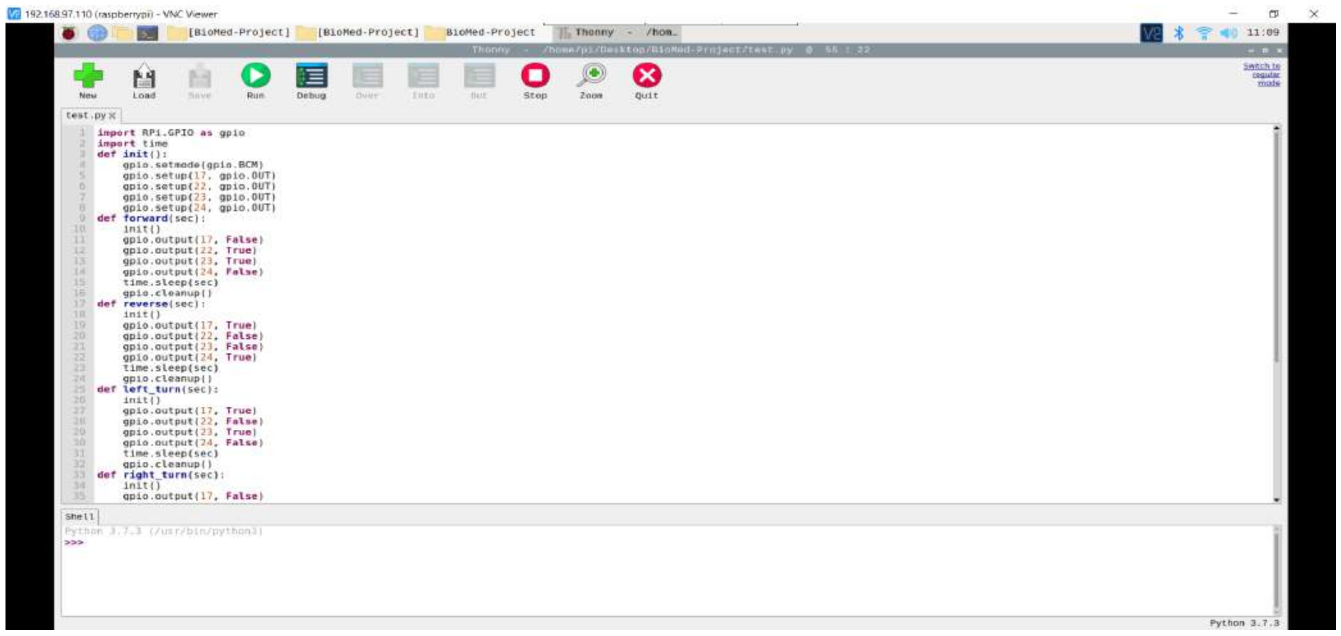
Training of a CNN model with the improved accuracy of 93%

SCREEN SHOTS



```python
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-bw.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-bw.h5")
print("Loaded model from disk")

cap = cv2.VideoCapture(0)

# Category dictionary
categories = {0: 'FORWARD', 1: 'BACKWARD', 2: 'LEFT', 3: 'RIGHT', 4: 'STOP', 5: 'BRAKE'}

while True:
    _, frame = cap.read()
    # Simulating mirror image
    frame = cv2.flip(frame, 1)

    # Got this from collect-data.py
    # Coordinates of the ROI
    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, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
    # Extracting the ROI
    roi = frame[y1:y2, x1:x2]

    # Resizing the ROI so it can be fed to the model for prediction
    roi = cv2.resize(roi, (64, 64))
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
    cv2.imshow("test", test_image)
    # Batch of 1
```

Code for prediction of a hand gesture which is fed as an input to the code for controlling motor driver using raspberry pi.

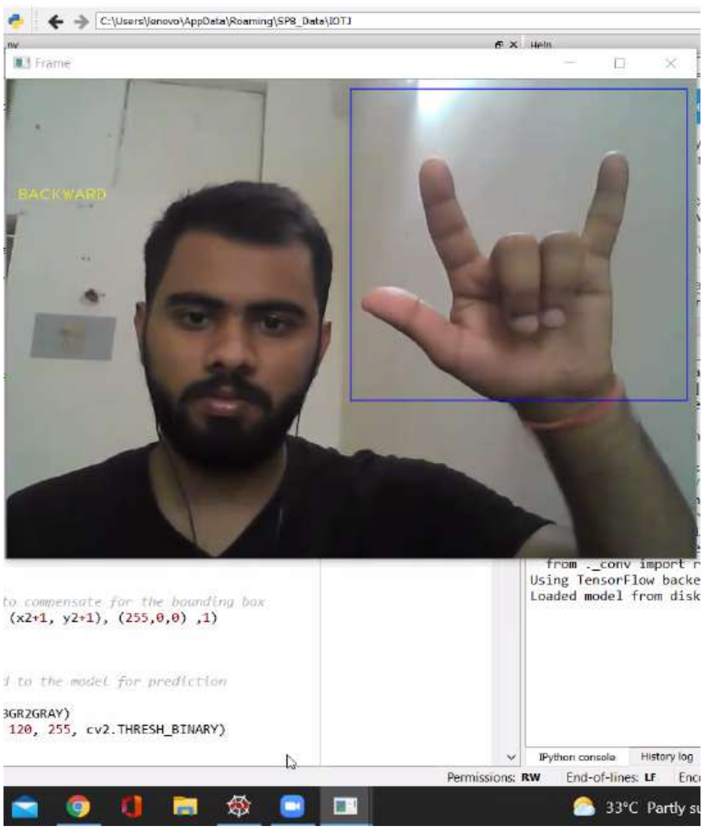


```python
import RPi.GPIO as gpio
import time
def init():
    gpio.setmode(gpio.BCM)
    gpio.setup(17, gpio.OUT)
    gpio.setup(22, gpio.OUT)
    gpio.setup(23, gpio.OUT)
    gpio.setup(24, gpio.OUT)
def forward(sec):
    init()
    gpio.output(17, False)
    gpio.output(22, True)
    gpio.output(23, True)
    gpio.output(24, False)
    time.sleep(sec)
    gpio.cleanup()
def reverse(sec):
    init()
    gpio.output(17, True)
    gpio.output(22, False)
    gpio.output(23, False)
    gpio.output(24, True)
    time.sleep(sec)
    gpio.cleanup()
def left_turn(sec):
    init()
    gpio.output(17, True)
    gpio.output(22, False)
    gpio.output(23, True)
    gpio.output(24, False)
    time.sleep(sec)
    gpio.cleanup()
def right_turn(sec):
    init()
    gpio.output(17, False)
```

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

Code for controlling the wheelchair using raspberry pi.

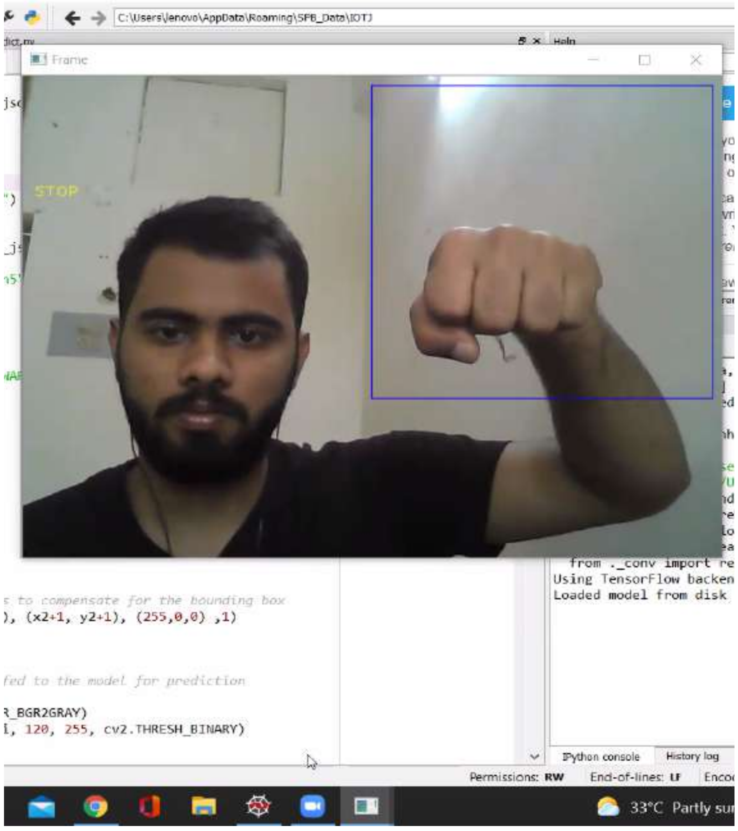
Hand gesture prediction using webcam



```
to compensate for the bounding box
 (x2+1, y2+1), (255,0,0) ,1)

d to the model for prediction

3GR2GRAY)
 120, 255, cv2.THRESH_BINARY)
```

```
  from ._conv import r
Using TensorFlow backe
Loaded model from disk
```

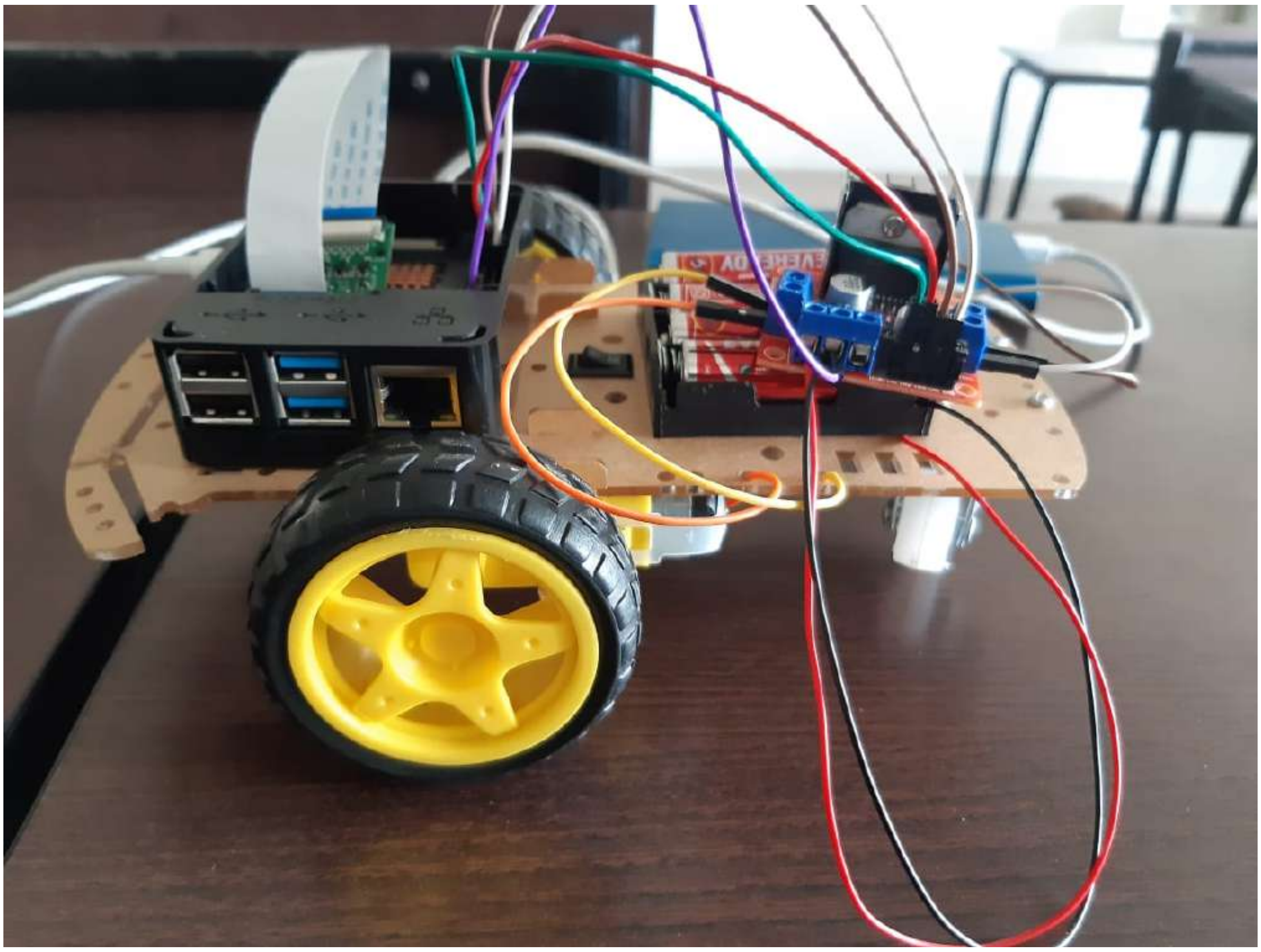
Permissions: RW   End-of-lines: LF   Enc



```
s to compensate for the bounding box
), (x2+1, y2+1), (255,0,0) ,1)

fed to the model for prediction

R_BGR2GRAY)
1, 120, 255, cv2.THRESH_BINARY)
```

```
  from ._conv import re
Using TensorFlow backen
Loaded model from disk
```

Permissions: RW   End-of-lines: LF   Encod



CONCLUSION

The proposed project was successfully carried out and the demonstration of the smart wheelchair was completed.It will be controlled using hand gestures and the demonstration was carried out which is done with the help of a small bot as well as input was taken using raspberry pi-cam.In order to make the wheelchair more reliable, user-friendly, affordable, safer and also accurate, we can also further work on different ideas in all the mentioned papers.
We successfully fed the dataset (around 100 pictures for each gesture)into the —using openCV and then we created the CNN model.We also trained the dataset.We were successfully able to predict some of the hand gestures like stop,forward,backward,left and right.These gestures can be used to perform various actions which can help in controlling various devices.The demonstration of the smart wheelchair will be done using a smart IoT bot.The hand gesture is recognized using raspberry pi-cam.Moreover,the movement of the bot is controlled with the help of raspberrypi which is a microprocessor.

FUTURE SCOPE

Since we came up with a device which will have the capability to make the movement of the person easier and without any dependency on any other person.The further price of the smart wheelchair can be reduced by analysis of the same. Also,the device can be further made more technology and human friendly if it is controlled using Bluetooth inbuilt in an android based smartphone.User can interact with the wheelchair more efficiently.Buzzer can also be present for notifying the people around during a risky situation for a panic alarm.Moreover,health monitoring system can also be embedded into the smart wheelchair for getting more information about the person.The real-time data related to the smart wheelchair and health can be monitored with the help of a smartphone.Moreover the health data can be stored in cloud or any other data storing platforms for the health analysis.Also, an ultrasonic sensor will be embedded for detecting the obstacle in the pathway.This wheelchair can have LED light connected to it, so anyone can move freely in dark conditions also.

REFERENCES

[1].https://www.sciencedirect.com/science/article/pii/S1877050920313715
[2].https://www.sciencedirect.com/science/article/pii/S1877705812026094
[3].https://ieeexplore.ieee.org/document/6804529
[4].https://ieeexplore.ieee.org/document/6717883
[5].https://ieeexplore.ieee.org/abstract/document/6629911
[6].https://ieeexplore.ieee.org/abstract/document/8714373
[7].https://www.sciencedirect.com/science/article/pii/S1877705811010381
[8].https://ieeexplore.ieee.org/document/7888236
[9].https://www.sciencedirect.com/science/article/pii/S2090447921002082
[10].https://www.sciencedirect.com/science/article/pii/S1877050919313766
[11].https://ieeexplore.ieee.org/document/7849501
[12]https://www.electronicshub.org/raspberry-pi-l298n-interface-tutorial-control
-dc-motor-l298n-raspberry-pi/