AUTOMATED DIRECTION DETECTING BOT

Project report submitted in partial fulfilment of the

requirement for the degree of

Bachelor of Technology

By

Aayush Mishra	**(16095001)**
Ashwini Raj	**(16095013)**
Kartikey Singh	**(16095029)**
Kushagra Sharma	**(16095033)**

Under the guidance of

Dr. V. N. Mishra

DEPARTMENT OF ELECTRONICS ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY (BHU), Varanasi

November 2019

CERTIFICATE

It is certified that the work contained in the project report titled “Automated Direction Detecting Bot” by the following student(s) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Name(s) of student(s)

Aayush Mishra	(16095001)
Ashwini Raj	(16095013)
Kartikey Singh	(16095029)
Kushagra Sharma	(16095033)

Dr. V. N. Mishra

Electronics Engineering

IIT(BHU), Varanasi

November 2019

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Aayush Mishra	(16095001)
Ashwini Raj	(16095013)
Kartikey Singh	(16095029)
Kushagra Sharma	(16095033)

Date: ____________________

Approval Sheet

This project report entitled "Automated Direction Detecting Bot", by the Aayush Mishra is approved for the degree of Bachelor of Technology, Electronics Engineering.

Name(s) of student(s)

Aayush Mishra	(16095001)
Ashwini Raj	(16095013)
Kartikey Singh	(16095029)
Kushagra Sharma	(16095033)

Examiners

____

____

____

____

____

Supervisor

Dr. V. N. Mishra

Head

Dr. V. N. Mishra

Date: ____________

Table of Contents

Chapter 1

Introduction

1.1 Aim of the Project

The aim of this project was to be able to create an automated wireless bot that processes the image in front of it and traverses according to the direction detected. The bot also uses object detection to avoid a collision if any obstruction is present in the path.

The bot uses Pi camera attached to raspberry pi for capturing images which are then classified using TensorFlow which uses Neural networks to predict whether the detected image is an upside, downside, left side or right-side arrow and according to the direction identified, The L298n motor driver is given input which drives both motors in that direction for a certain amount of time before again recapturing and restarting the process. The bot also uses SONAR sensors to detect objects to avoid collision and finds an alternate path to resume traversal.

We have used two power sources of 5V and 9V to drive Raspberry Pi and L298n motor driver respectively for providing proper power supply to make our project wireless.

1.2 Scope of the Project

Automated vehicles to move in a constrained environment like wildlife sanctuaries and recreational parks and avoid obstacles in the path.
Helping people with disabilities to charter the course.
Usage in hostile areas to help deliver good and pieces of equipment using the clues and directions left by others if communication is not possible using radios.
Able to detect different signs of danger and hazards and accordingly change direction from those paths and avoid barriers and hurdles in the path.

1.3 Problems Posed

1.3.1 Creating the dataset:

Manually captured hundreds of pictures in different setups.
Selecting different angles, lighting and distortion in images to create a diverse dataset.

1.3.2 Installing correct dependencies:

Raspberry OS uses a specific version of dependencies which are different than that on PC.
A lot of time was spent on debugging the problems faced in the python script.

1.3.3 Hardware problems:

Proper calibration of motor speed and making the bot wireless.
Calibration of SONAR to avoid obstructions at a proper distance.
Connections keep becoming loose due to the vibrations during motion of the bot.
Properly adjusting all the components on the bot chassis due to the problem of space and need to make it compact.
Properly alignment of castor and plastic wheels to make the bot stable.

1.3.4 Model Selection and Training:

Selecting suitable classifiers with minimum loss error.
Selection of the number of training steps for CNN.
Selection of a number of layers and the architecture of CNN model.

Chapter 2
Review of Literature

Very Deep Convolutional Networks for Large-scale Image Recognition by Karen Simonyan and Andrew Zisserman was reviewed.
Convolutional networks (ConvNets) have recently enjoyed great success in the large-scale image and video recognition) which has become possible due to the large public image repositories, such as ImageNet and high-performance computing systems, such as GPUs or large-scale distributed clusters.
As a result, they came up with significantly more accurate ConvNet architectures, which not only achieve the state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance.
Given an image, it can find an object name in the image. It can detect any one of 1000 images and takes input image of size 224 * 224 * 3 (RGB image). It is built using Convolutions layers (used only 3*3 size) and Max pooling layers (used only 2*2 size) and Fully connected layers at the end. It has a total of 19 layers. It has a very simple architecture where height and width channel half at every few intervals and channels increase at the same time. Letting the model discover various kind of features.

“A Survey on Deep Transfer Learning” by Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu was reviewed.
In some domains, like bioinformatics and robotics, it is very difficult to construct a large-scale well-annotated dataset due to the expense of data acquisition and costly annotation, which limits its development. Transfer learning relaxes the hypothesis that the training data must be independent and identically distributed (i.i.d.) with the test data, which motivated us to use transfer learning to solve the problem of insufficient training data.

So, as we do not have a large amount of data-set we applied transfer learning to the VGG19 model and got a SoftMax layer output for four types of objects with high accuracy.

Chapter 3
Report on the present investigation

3.1 Setup

1. Raspberry Pi 3 Model B

It is like a small CPU having Broadcom BCM2837 SoC, 1GB RAM and 4x ARM Cortex at 1.2 GHz. It has 40 GPIO pins, HDMI, 3.5mm audio and 4x USB and Ethernet ports. We wrote the code in python language and installed all the required libraries in it. It captures and processes the images.

2. Raspberry Pi Camera Module V2:

It has an 8 Megapixel 1080p camera based around Sony IMX219 image sensor. It is capable of 3280x2464 pixel static images and supports 1080p30 720p60 and 640x480p90 video. It acts as an external device connected to the raspberry pi and is used to capture the image (whenever raspberry pi sends command, in our case).

3. H-Bridge L298n Motor Driver Module

It has an integrated 5V power regulator, 5V – 35V drive voltage and a 2A max drive current, max power of 25W. It can drive up to 2 bidirectional DC motors. It is handy when controlling the direction of rotation of a DC motor.

4. HC-SR04 Ultrasonic Distance Measuring Sensor Module

It has working voltage of DC 5V, working current of 15mA, working frequency of 40Hz, minimum range of 2cm and a maximum range of 4m.To avoid the obstruction in the path.

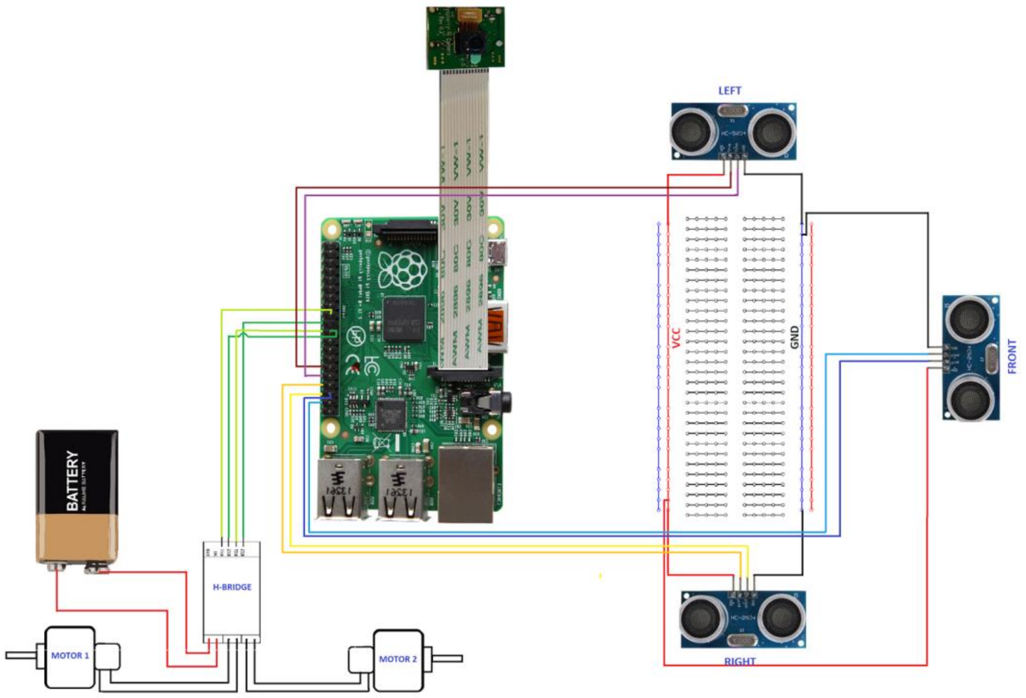
5. 100 rpm, 12 Volts DC Motor

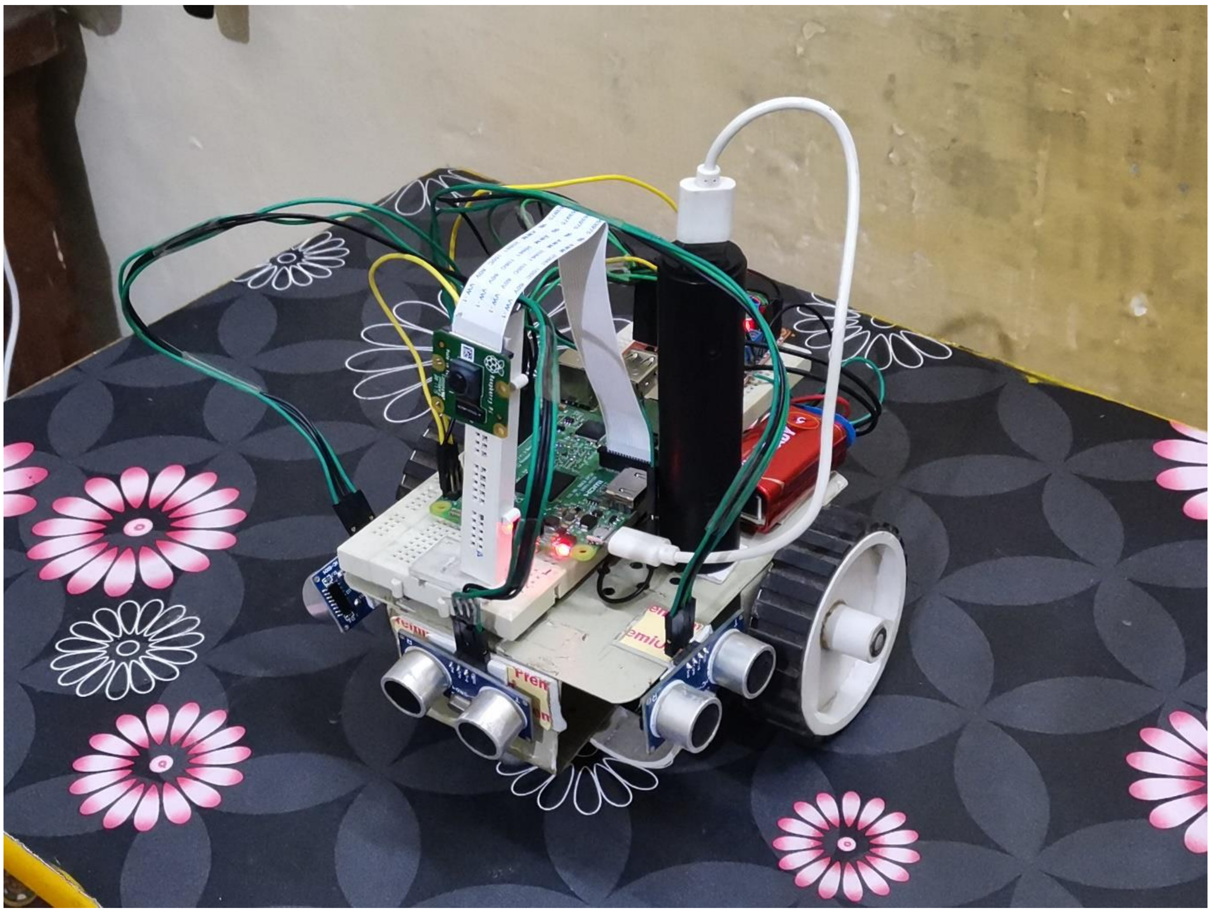
To run the plastic wheels at a suitable speed with a maximum of 100 rpm and is run of 12V DC supply. It is bidirectional in operation and is controlled by the motor driver L298n.

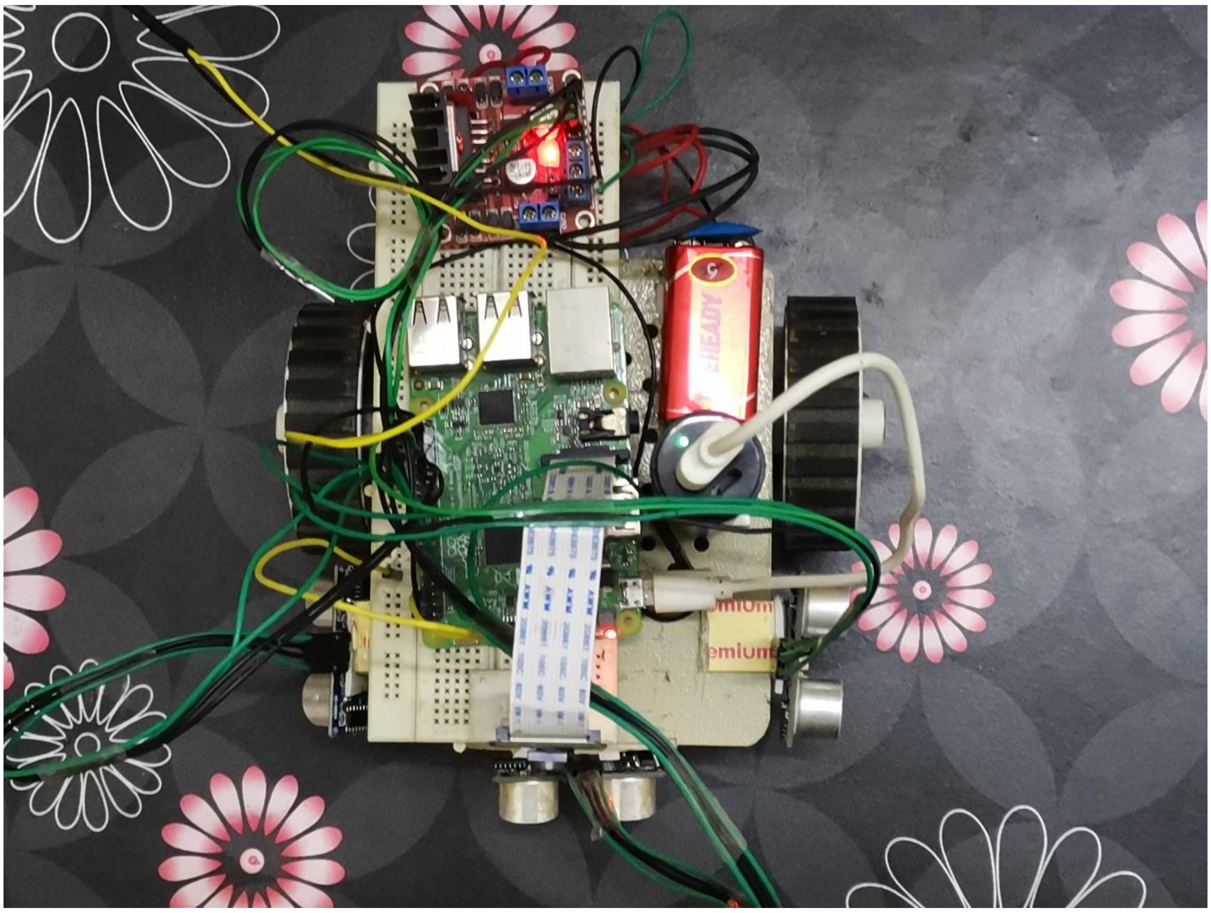
6. Super Jumbo Bot Chassis

Mechanical frame structure to provide support to all the pieces of our bot and assemble them in one place. It’s light weight, non-flexible and resistant to corrosion.

3.2 Circuitry







3.3 Procedure

The Raspberry Pi was installed with the Raspbian OS using the Etcher image. All the necessary compatible dependencies which were required to run the script were installed. Connection between the PC and Raspberry Pi was established using SSH (Secured Shell) and VNC (Virtual Network Computing) when both were connected over the same wireless network.

After this dataset of thousands of images were clicked using the Raspberry Pi Camera. It was made sure that the images had varying distances and background light. These images were then used to train the machine learning model. The fine tuning of machine learning was done to increase the accuracy on the dataset using the training and testing phases. The model was finally saved on the Raspberry Pi.

The wheels and the motors were attached on the chassis, proper connection was done between the motors and the L298n motor driver. Proper GPIO connections were made between the motor driver L298n and the Raspberry Pi. The ultrasonic sensors HC-SR04 were connected with the Raspberry Pi. The Raspberry Pi and motor driver were connected with 5V and 9V power supply respectively to make the system wireless. The setup was then calibrated for proper movement of the bot and the working of the ultrasonic sensors. Finally, the Raspberry Pi camera module was attached to the Raspberry Pi module.

Now to test the bot images of various directions was shown to the Raspberry Pi camera and the classification was done by the model trained. The accuracy of the classification was measured and it timing for the classification was noted. Various obstacles were placed in the path to test the collision aversion of the bot and was repeated multiple times for the properly calibration of the bot.

3.4 Source Code

3.4.1 Machine learning training

```python
from __future__ import absolute_import, division, print_function, unicode_literals

from keras import layers
from keras import models
from keras.applications import VGG16
from keras.layers import AveragePooling2D, MaxPooling2D, Dropout, GlobalMaxPooling2D, GlobalAveragePooling2D
from keras.layers import Input, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D
from keras import optimizers
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, TensorBoard, EarlyStopping
from keras.applications.vgg19 import VGG19
from datetime import datetime
import os
from keras.preprocessing import image
from keras.models import load_model
import tensorflow as tf
import IPython.display as display
from PIL import Image
import numpy as np
import pathlib
import matplotlib.pyplot as plt
AUTOTUNE = tf.data.experimental.AUTOTUNE
tf.__version__

from keras.models import Sequential, Model
from keras import backend as k

TRAIN_PATH = 'Train'
TEST_PATH = 'Test'

BATCH_SIZE = 16
IMG_HEIGHT = 224
IMG_WIDTH = 224
EPOCHS = 4


data_dir = pathlib.Path(TRAIN_PATH)
CLASS_NAMES = np.array([item.name for item in data_dir.glob('*')])


def data_generator(dir_path):
    data_dir = pathlib.Path(dir_path)
    CLASS_NAMES = np.array([item.name for item in data_dir.glob('*')])
    # print(CLASS_NAMES)
    image_count = len(list(data_dir.glob('*/*.jpg')))
    STEPS_PER_EPOCH = np.ceil(image_count/BATCH_SIZE)
    # The 1./255 is to convert from uint8 to float32 in range [0,1].
    image_generator = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                                                      brightness_range=[0.7, 1.0],)
    data_gen = image_generator.flow_from_directory(directory=str(data_dir),
                                                   batch_size=BATCH_SIZE,
                                                   shuffle=True,
                                                   target_size=(
                                                       IMG_HEIGHT, IMG_WIDTH),
    classes=list(CLASS_NAMES))
    return data_gen, STEPS_PER_EPOCH


def train_valid_generator(dir_path):
    data_dir = pathlib.Path(dir_path)
    CLASS_NAMES = np.array([item.name for item in data_dir.glob('*')])
    # print(CLASS_NAMES)
    image_count = len(list(data_dir.glob('*/*.jpg')))
    # The 1./255 is to convert from uint8 to float32 in range [0,1].
    image_generator = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
    validation_split=0.2,
    brightness_range=[0.7, 1.0],)

    train_gen = image_generator.flow_from_directory(directory=str(data_dir),
                                                    batch_size=BATCH_SIZE,
                                                    shuffle=True,
                                                    target_size=(
                                                        IMG_HEIGHT, IMG_WIDTH),
                                                    classes=list(CLASS_NAMES),
                                                    subset='training')

    valid_gen = image_generator.flow_from_directory(directory=str(data_dir),
                                                    batch_size=BATCH_SIZE,
                                                    shuffle=True,
                                                    target_size=(
                                                        IMG_HEIGHT, IMG_WIDTH),
                                                    classes=list(CLASS_NAMES),
                                                    subset='validation')

    return train_gen, valid_gen


train_data_gen, valid_data_gen = train_valid_generator(TRAIN_PATH)
test_data_gen, STEPS_TEST = data_generator(TEST_PATH)

#Load the VGG model
vgg_conv = VGG19(weights='imagenet', include_top=False,
                 input_shape=(IMG_HEIGHT, IMG_WIDTH, 3))

# Freeze the layers except the last 4 layers
for layer in vgg_conv.layers[:-4]:
    layer.trainable = False

# Check the trainable status of the individual layers
for layer in vgg_conv.layers:
    print(layer, layer.trainable)


      # Create the model
      model = models.Sequential()

      # Add the vgg convolutional base model
      model.add(vgg_conv)

      # Add new layers
      model.add(layers.Flatten())
      model.add(layers.Dense(1024, activation='relu'))
      model.add(layers.Dropout(0.5))
      model.add(layers.Dense(4, activation='softmax'))

      # Show a summary of the model. Check the number of trainable parameters
      print(model.summary())

      # Compile the model
```

```
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])

        # Train the model
        history = model.fit_generator(
            generator=train_data_gen,
            steps_per_epoch=train_data_gen.samples//BATCH_SIZE,
            validation_data=valid_data_gen,
            validation_steps=valid_data_gen.samples//BATCH_SIZE,
            epochs=EPOCHS)

        now = datetime.now()
        model.save('models/model_' + str(now) + '.h5')

        # Plotting the model
        acc = history.history['acc']
        val_acc = history.history['val_acc']
        loss = history.history['loss']
        val_loss = history.history['val_loss']

        epochs = range(len(acc))

        plt.plot(epochs, acc, 'b', label='Training acc')
        plt.plot(epochs, val_acc, 'r', label='Validation acc')
        plt.title('Training and validation accuracy')
        plt.legend()

        plt.figure()

        plt.plot(epochs, loss, 'b', label='Training loss')
        plt.plot(epochs, val_loss, 'r', label='Validation loss')
        plt.title('Training and validation loss')
        plt.legend()

        plt.show()
```

3.4.2 Direction Detection Script

```
import tensorflow as tf, sys
import matplotlib.image as img
from picamera.array import PiRGBArray
from picamera import PiCamera
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np

#hardware work
GPIO.setmode(GPIO.BOARD)

GPIO_TRIGGER1 = 29      #Left ultrasonic sensor
GPIO_ECHO1 = 31

GPIO_TRIGGER2 = 36      #Front ultrasonic sensor
GPIO_ECHO2 = 37

GPIO_TRIGGER3 = 33      #Right ultrasonic sensor
GPIO_ECHO3 = 35

```

```python
MOTOR1B=22  #Left Motor
MOTOR1E=18

MOTOR2B=19  #Right Motor
MOTOR2E=21

LED_PIN=13

# Set pins as output and input
GPIO.setup(GPIO_TRIGGER1,GPIO.OUT)  # Trigger
GPIO.setup(GPIO_ECHO1,GPIO.IN)      # Echo
GPIO.setup(GPIO_TRIGGER2,GPIO.OUT)
GPIO.setup(GPIO_ECHO2,GPIO.IN)
GPIO.setup(GPIO_TRIGGER3,GPIO.OUT)
GPIO.setup(GPIO_ECHO3,GPIO.IN)
GPIO.setup(LED_PIN,GPIO.OUT)

# Set trigger to False (Low)
GPIO.output(GPIO_TRIGGER1, False)
GPIO.output(GPIO_TRIGGER2, False)
GPIO.output(GPIO_TRIGGER3, False)

GPIO.setup(MOTOR1B, GPIO.OUT)
GPIO.setup(MOTOR1E, GPIO.OUT)
GPIO.setup(MOTOR2B, GPIO.OUT)
GPIO.setup(MOTOR2E, GPIO.OUT)

def sonar(GPIO_TRIGGER,GPIO_ECHO):
      start=0
      stop=0
      # Set pins as output and input
      GPIO.setup(GPIO_TRIGGER,GPIO.OUT)  # Trigger
      GPIO.setup(GPIO_ECHO,GPIO.IN)      # Echo

      # Set trigger to False (Low)
      GPIO.output(GPIO_TRIGGER, False)

      # Allow module to settle
      time.sleep(0.01)

      #while distance > 5:
      #Send 10us pulse to trigger
      GPIO.output(GPIO_TRIGGER, True)
      time.sleep(0.00001)
      GPIO.output(GPIO_TRIGGER, False)
      begin = time.time()
      while GPIO.input(GPIO_ECHO)==0 and time.time()<begin+0.05:
            start = time.time()

      while GPIO.input(GPIO_ECHO)==1 and time.time()<begin+0.1:
            stop = time.time()

      # Calculate pulse length
      elapsed = stop-start
      # Distance pulse travelled in that time is time
      # multiplied by the speed of sound (cm/s)
      distance = elapsed * 34000

      # That was the distance there and back so halve the value
      distance = distance / 2

      print("Distance : %.1f" % distance)
      # Reset GPIO settings
      return distance

#Function for alternate path
```

```
def avoid():
    stop()
    leftturn()
    time.sleep(0.8)
    stop()
    forward()
    time.sleep(2)
    stop()
    rightturn()
    time.sleep(0.7)
    stop()
    forward()
           time.sleep(2)
           stop()

       # Function to detect Obstacle
       def detect():
           start = time.time()
           time.sleep(0.5)
           while(True):
           distance = sonar(GPIO_TRIGGER2,GPIO_ECHO2)
           if distance<15 and distance>8:
               print('too close')
               avoid()
               return 0
           now = time.time()
           if (now - start) > 5:
               print('time over')
               stop()
               return 0

       def forward():
             GPIO.output(MOTOR1B, GPIO.HIGH)
             GPIO.output(MOTOR1E, GPIO.LOW)
             GPIO.output(MOTOR2B, GPIO.HIGH)
             GPIO.output(MOTOR2E, GPIO.LOW)

       def rightturn():
             GPIO.output(MOTOR1B,GPIO.HIGH)
             GPIO.output(MOTOR1E,GPIO.LOW)
             GPIO.output(MOTOR2B,GPIO.LOW)
             GPIO.output(MOTOR2E,GPIO.HIGH)


       def reverse():
             GPIO.output(MOTOR1B, GPIO.LOW)
             GPIO.output(MOTOR1E, GPIO.HIGH)
             GPIO.output(MOTOR2B, GPIO.LOW)
             GPIO.output(MOTOR2E, GPIO.HIGH)


       def leftturn():
             GPIO.output(MOTOR1B,GPIO.LOW)
             GPIO.output(MOTOR1E,GPIO.HIGH)
             GPIO.output(MOTOR2B,GPIO.HIGH)
             GPIO.output(MOTOR2E,GPIO.LOW)

       def stop():
             GPIO.output(MOTOR1E,GPIO.LOW)
             GPIO.output(MOTOR1B,GPIO.LOW)
             GPIO.output(MOTOR2E,GPIO.LOW)
             GPIO.output(MOTOR2B,GPIO.LOW)


       dict={}
       #Function for Classification
```

```python
def classify(image_path):
    # Read in the image_data
    image_data = tf.gfile.FastGFile(image_path, 'rb').read()
    # Loads label file, strips off carriage return
    label_lines = [line.rstrip() for line
    in tf.gfile.GFile("retrained_labels.txt")]
    # Unpersists graph from file
    with tf.gfile.FastGFile("retrained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')
    # Feed the image_data as input to the graph and get first prediction
    score_max=0
    ab=""
    with tf.Session() as sess:
            softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

            predictions = sess.run(softmax_tensor,{'DecodeJpeg/contents:0': image_data})
            # Sort to show labels of first prediction in order of confidence

            top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
            for node_id in top_k:
                human_string = label_lines[node_id]
                score = predictions[0][node_id]
                human_string=str(human_string)
                dict[human_string]=score
                if score>score_max:
                    ab=human_string
                    score_max=score
                #print('%s (score = %.5f)' % (human_string, score))

    return (ab)


#Getting a Camera Instance
camera = PiCamera()
camera.start_preview()
time.sleep(5)
camera.capture('/home/pi/btp4/image.jpg')
camera.stop_preview()

image_path = 'image.jpg'
s=classify(image_path)
print("cvlsidfvbslkdnfv"+s+"kanskuevfCSJECGVL")

if s=="left" or s=="right":
    image=img.imread(image_path)
    im=np.rot90(image)
    img_path=image_path+"2"
    img.imsave(img_path,im)
    abc=classify(img_path)
    if abc=="up":
    abcd="right"
    accuracy=dict["up"]
    else:
    abcd="left"
    accuracy=dict["down"]

else:
    abcd=s
    accuracy=dict[s]

print(str(abcd)+" with accuracy "+str(accuracy))
s=abcd

```

```
        if s=="up":
            reverse()
            time.sleep(3)
            stop()

        elif s=="down":
            forward()
            time.sleep(3)
            stop()

        elif s=="left":
            rightturn()
            time.sleep(1.4)
            stop()

        else:
            leftturn()
            time.sleep(1.4)
            stop()

        GPIO.cleanup()
```

3.5 Technology

GPIO (General Purpose Input/Output) - GPIO is a set of generic pins on an integrated circuit or computer board. The libraries are required to be installed and imported for these pins to interact with the code written in python.

Time - This is required to import time modules in python. The image that is saved on the server is named on the basis of time and date.

SSH (Secured Shell)- Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line login and remote command execution, but any network service can be secured with SSH.

It helped us use various commands of Raspberry Pi using our laptop which eliminated the use of the LED screen. This was established using the IP address of the Raspberry Pi.

VNC (Virtual Network Computing)- Sometimes it is not convenient to work directly on the Raspberry Pi. Maybe you would like to work on it from another device by remote control.

VNC is a graphical desktop sharing system that allows you to remotely control the desktop interface of one computer (running VNC Server) from another computer or mobile device (running VNC Viewer). VNC Viewer transmits the keyboard and either mouse or touch events to VNC Server and receives updates to the screen in return.

You will see the desktop of the Raspberry Pi inside a window on your computer or mobile device. You'll be able to control it as though you were working on the Raspberry Pi itself.

It helped us view the display of raspberry pi on our laptop itself which again eliminated the use of 5inch screen and enabled us to use the bot using only the laptop.

Remmina- Remmina is a remote desktop client for computer operating systems. It supports the Remote Desktop Protocol(RDP), VNC, NX, XDMCP, SPICE and SSH protocols.

Remmina is in the package repositories for Debian versions 6 and later and for Ubuntu versions since 10.04. VNC was established using Remmina on Ubuntu 18.04.

Numpy: NumPy 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. We used Numpy to process images for training the dataset.

BalenaEtcher- This software was used to write the downloaded image of Raspian on the memory card for installation on raspberry pi.

3.6 Methods Developed and Used

3.6.1 Classification

Image classification refers to the task of extracting information classes from a multiband raster image. The resulting raster from image classification can be used to create thematic maps. Depending on the interaction between the analyst and the computer during classification, there are two types of classification: supervised and unsupervised.

3.6.2 Supervised classification

Supervised classification uses the spectral signatures obtained from training samples to classify an image. We can easily create training samples to represent the classes we want to extract.

3.6.3 Unsupervised classification

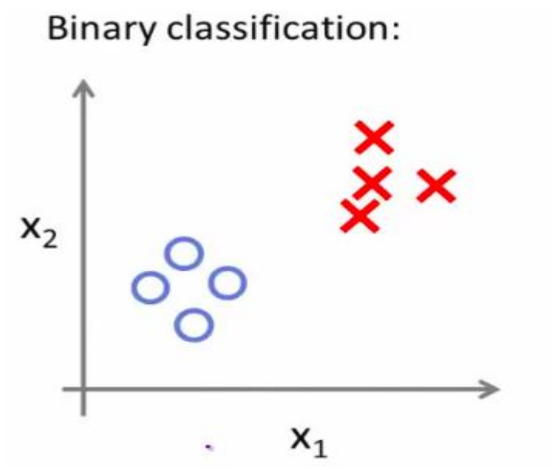
Unsupervised classification finds spectral classes (or clusters) in a multiband image without the analyst’s intervention. It aids in unsupervised classification by providing access to the tools to create the clusters, capability to analyze the quality of the clusters, and access to classification tools.

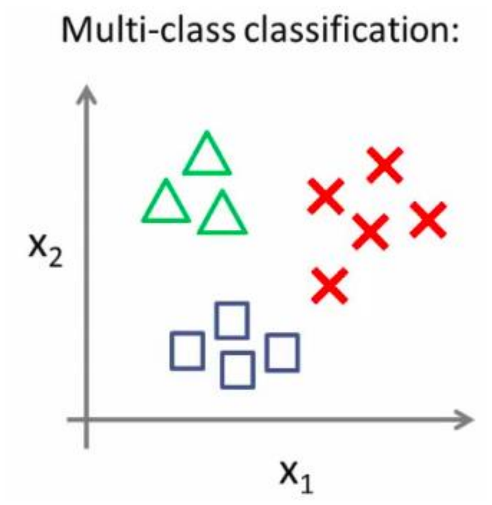
3.6.4 Binary Classification

This type of classification classifies a given data into one of the two specified classes.

3.6.5 Multiclass Classification

This type of classification classifies a given data into more than two of the specifies classes.





Chapter 4

Results and Discussions

4.1 Results

After running the experiment multiple times, we found that the classification of the image was satisfactory.

The accuracy was found to be more than 90% in all cases. The average time taken by the bot to detect the image and move accordingly was found to be around 10-12 seconds. So the classifier provided by TensorFlow was precise. The accuracy also supported the fact that the database created was adequate. Even on changing the position of the arrow and varying the environment parameters the classification was still on point.

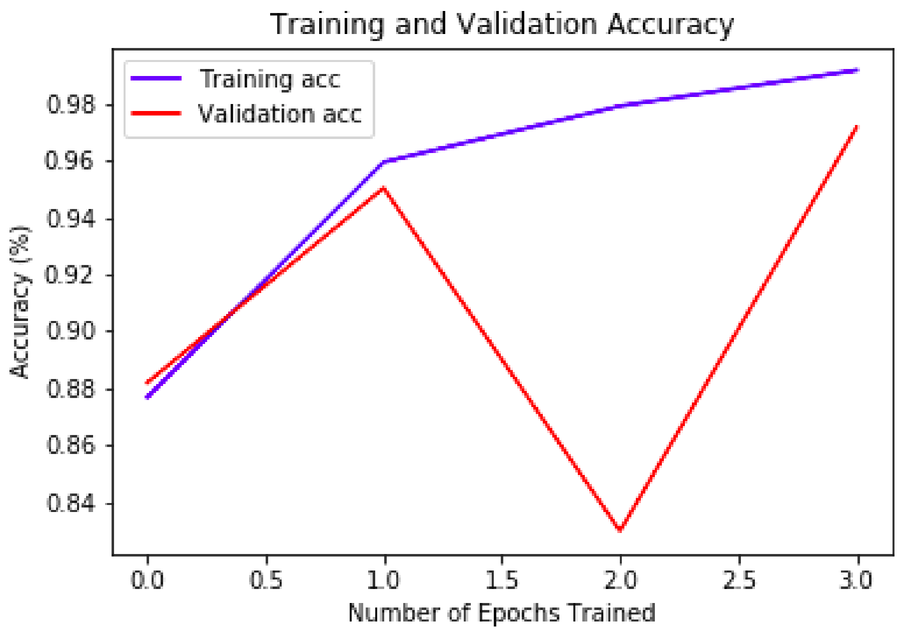
The quality of the pi camera was on point as observed by the image saved.

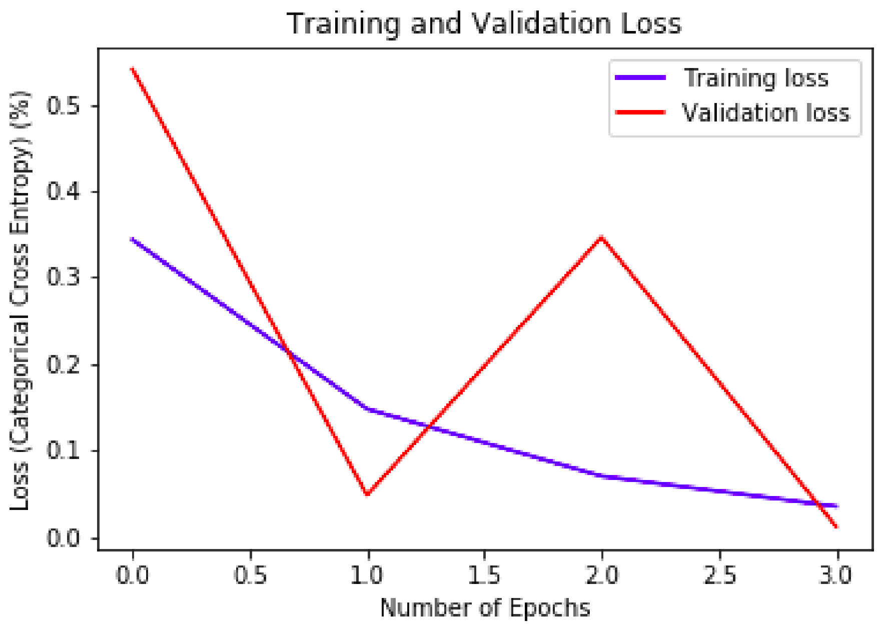
The SONAR sensors were accurate on point and avoided the obstructions in the path to find an alternate path and resumed traversing. Whenever the sensors detected an obstruction in the path nearer than about 15cm then the bot finds an alternate path and starts traversing again.

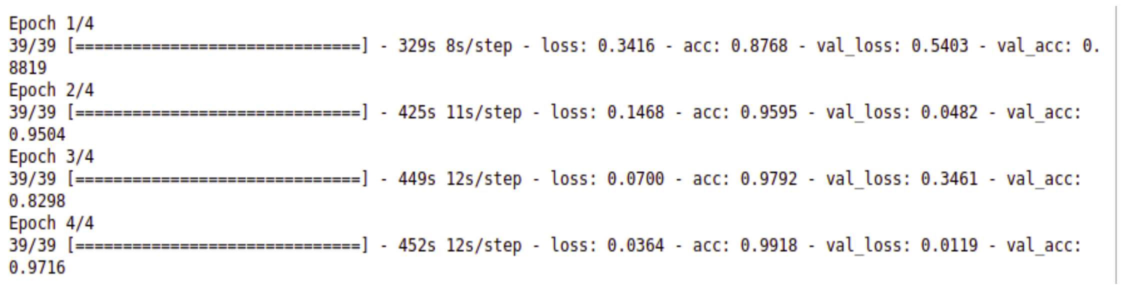
Also, after the image was classified and the direction was detected the motor driver drove both the motors accordingly. Calibration done during the procedure was found to be optimum, and the turning of the bot for left, right directions was correct.

Both the power sources were sufficient enough to drive the motor driver and Raspberry Pi.

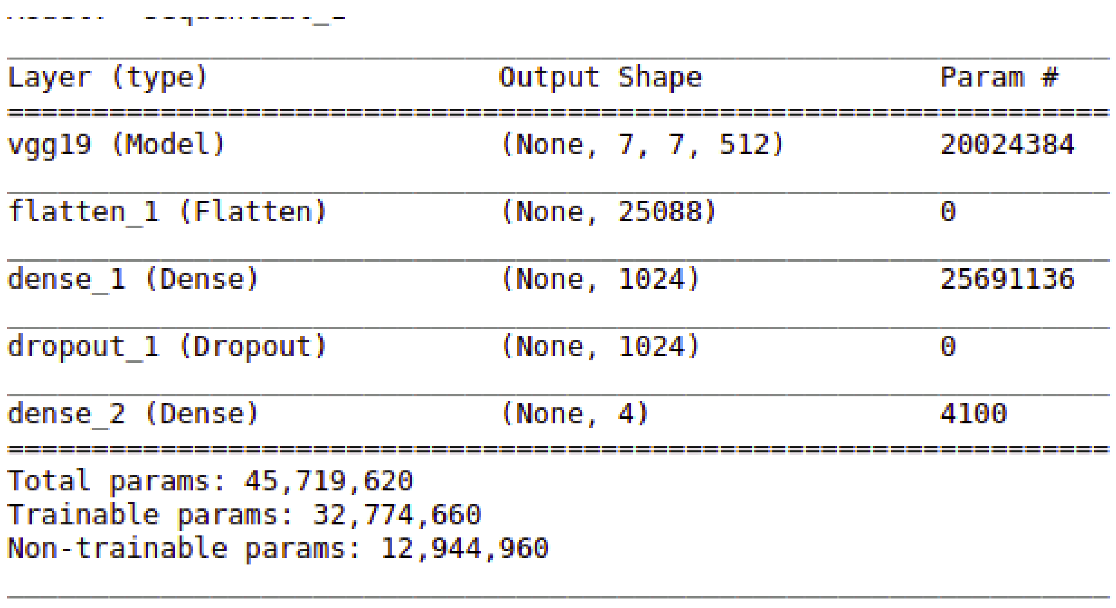
Hence such a bot can be used for automated direction detection and the respective traversal, independently without any human aid. Also, it can be trusted to provide you correct results almost all the time and provide you with the feedback of the accuracy by which the image was detected.


Training and Validation Accuracy
Training acc
Validation acc
Accuracy (%)
0.84
0.86
0.88
0.90
0.92
0.94
0.96
0.98
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Number of Epochs Trained


Training and Validation Loss
Training loss
Validation loss
Loss (Categorical Cross Entropy) (%)
0.0
0.1
0.2
0.3
0.4
0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Number of Epochs



```
Epoch 1/4
39/39 [==============================] - 329s 8s/step - loss: 0.3416 - acc: 0.8768 - val_loss: 0.5403 - val_acc: 0.
8819
Epoch 2/4
39/39 [==============================] - 425s 11s/step - loss: 0.1468 - acc: 0.9595 - val_loss: 0.0482 - val_acc:
0.9504
Epoch 3/4
39/39 [==============================] - 449s 12s/step - loss: 0.0700 - acc: 0.9792 - val_loss: 0.3461 - val_acc:
0.8298
Epoch 4/4
39/39 [==============================] - 452s 12s/step - loss: 0.0364 - acc: 0.9918 - val_loss: 0.0119 - val_acc:
0.9716
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Model)                (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              25691136
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4100
=================================================================
Total params: 45,719,620
Trainable params: 32,774,660
Non-trainable params: 12,944,960

_________________________________________________________________
```

4.2 Discussions

After going through the results, we can discuss the scope of possible further future work for this project.

Majorly this project could be helpful in the field of automated transportation with automatic obstruction aversion. Mostly in constrained areas where the path to be followed is linear and repetitive such as wildlife sanctuaries, zoos and recreational parks. In these places, a track dedicated for such transport can be made consisting of arrow signals at proper positions.

One of the significant modifications that can be made is the addition of ultrasonic sensors which would help in preventing collisions and accidents by detecting an object in the path ahead. Addition of voice control can also add another dimension to the transport. On the one hand it can provide us with us override controls such as to stop the car in case of an emergency but on the other hand, can also be helpful to visually impaired persons. And not only them, but this project can also be useful in the lives of differently-abled people and assist them in their day-to-day lives.

There are also a few means to increase the efficiency of the project. Building a larger, better and diverse dataset is the first way to increase the accuracy of classification. A different dataset would mean taking images that vary in distance, position, angle and brightness. Reducing the time required for the classification of the pictures is another point in which improvement can be made. Some little changes can also be made to algorithm to increase both the accuracy and the time efficiency of the operation.

In addition to this, to handle many such bots or car a centralized server can be made which would handle mass classification requests and then after processing those requests, send the proper commands to the respective bot/car. By using such a method, we would not require separate processor for each bot/car instead a single master processor would replace them.

Chapter 5

Summary and Conclusion

5.1 Summary

To summarize we created an automated wireless bot that processes the image in front of it and after determining the direction of the arrow traverses accordingly and avoids the hurdle in the path and traverses along the alternate path. Raspberry Pi used as the processing unit. Motor driver L298N was used to drive the motors according to the input provided by the Raspberry Pi. A Pi camera was attached to the Raspberry Pi as an external device and was responsible for capturing the images which would later be processed. Three HC-SR04 ultrasonic sensors are used to avoid the obstruction in the path. Individual power sources were required to make the bot wireless.

This report starts us with the aim and scope of the project followed by the problem posed. The setup was presented which included the circuit diagram and the components used. Procedure pursued was mentioned along with the techniques used.

Finally, we covered the results of the project and discussed the scope of possible further future work for this project.

5.2 Conclusion

The accuracy of the classification of images was about 90% denoting that the classifier was precise.

The average time taken for classification was about 10-12 seconds.

The motor driver was calibrated for optimum movement of the bot.

Ultrasonic sensors were calibrated for collision aversion.

Accuracy of the result depends on the quality of the image.

Power sources were used to make the bot wireless.

Further upgrades planned include speeding up the classification, increasing the accuracy further.

A centralized server usage can help in controlling several bots at the same time to reduce the number of processors.

Chapter 6

Appendix

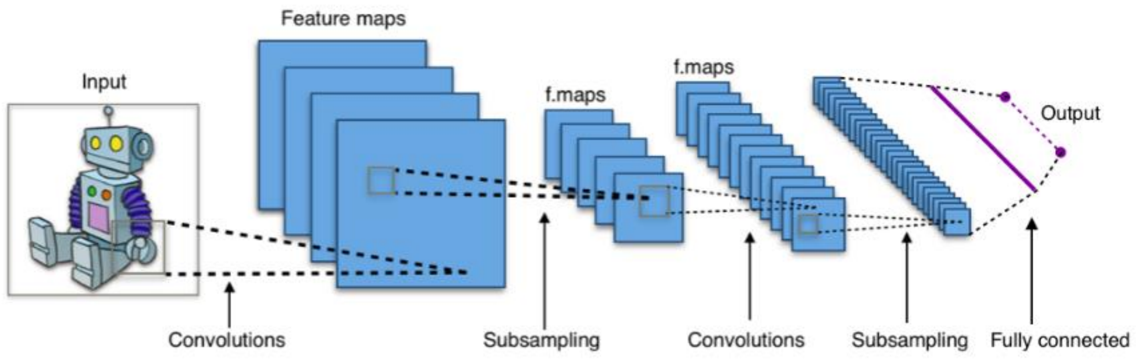
Our project, we have used Convolutional Neural Networks (CNNs) to train our image dataset.

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analysing visual imagery.

CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extremity.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.



Working of CNNs:

Convolutional Layer:

It computes the output of neurons that are connected to local regions of receptive fields in the input, each computing a dot product between their weights and a small receptive field to which they are connected to the input volume. Each computation leads to the extraction of a feature map from the input image.

The objective of subsampling is to get an input representation by reducing its dimensions, which helps in reducing overfitting.

ReLu Layer:

In this layer, we remove every negative value from the filtered images and replace them with zeros. This is done to avoid the values from summing up to zero.

Pooling Layer:

One of the techniques of subsampling is max pooling. With this technique, you select the highest pixel value from a region depending on its size.

Fully Connected Layer:

This is the final layer where the actual classification happens. Here we take our filtered and shrunk images and put them into a single list. The objective of the fully connected layer is to flatten the high-level features that are learned by convolutional layers and combining all the features. It passes the flattened output to the output layer where you use a SoftMax classifier or a sigmoid to predict the input class label.

References

Piddler in the Root
http://www.piddlerintheroot.com/l298n-dual-h-bridge/

Raspberry Pi OS
https://www.raspberrypi.org/downloads/

Very Deep Convolutional Networks for Large-scale Image Recognition By Karen Simonyan and Andrew Zisserman.
https://arxiv.org/pdf/1409.1556.pdf

A Survey on Deep Transfer Learning by Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu
https://arxiv.org/abs/1808.01974

Fritzing Software (for Diagram)
http://fritzing.org/

SSH and VNC
https://www.raspberrypi.org/documentation/remote-access/vnc/

HC-SR04
https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Acknowledgement

I would like to express our deep sense of gratitude and indebtedness to Dr. V. N. Mishra, Department of Electronics Engineering, IIT (BHU), Varanasi, for providing his invaluable guidance, comments and suggestions throughout the course of the project.

I am also thankful to all the other faculty & staff members of the department for their kind co-operation and help. I also extend our sincere thanks to all the friends who have patiently helped in accomplishing this undertaking.

Lastly, I would like to express my deep apperception towards classmates and indebtedness to my parents for providing me with moral support and encouragement.