MAIN PROJECT REPORT

IMPLEMENTATION OF NEURAL NETWORK ON RC CAR

BY

ASWIN T.R	20
BOBY JOSE	21
MUAHMMED SHANIB MT	41
RAMKUMAR M	47

DEPARTMENT OF ELECRONICS and COMMUNICATION ENGINEERING

Govt. Engineering College Thrissur

Thrissur, 680009

2017-2018

Acknowledgements

We are profoundly grateful to **Prof. Asha J** for her expert guidance and continuous encouragement throughout to see that this project reachs its target since its commencement to its completion.

We would like to express deepest appreciation towards **Prof. Dr. Thajudin Ahamed V. I.**, Head of Department of Electronics and Communication Engineering and **Prof. Latha K. N.**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Electronics and Communication Engineering Department who helped me directly or indirectly during this course of work.

ASWIN T R
BOBY JOSE
MUHAMMED SHANIB
RAMKUMAR M

ABSTRACT

Artificial neural networks (ANNs) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains.Such systems ”learn” (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as ”cat” or ”no cat” and using the results to identify cats in other images. They do this without any a prior knowledge about cats, e.g., that they have fur, tails, whiskers and cat-like faces. Instead, they evolve their own set of relevant characteristics from the learning material that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons (a simplified version of biological neurons in an animal brain). Each connection (a simplified version of a synapse) between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.

In this project an RC Car is first trained in a track and thus a data set is created.The data set so created is used to train a neural network and a model is created.Using this model the car is made to self drive in any new track it sees.Through this project we trying to learn machine learning concepts and tools and get accustomed with it.

Contents

List of Figures

Chapter 1

Introduction

Artificial intelligence is growing exponentially. There is no doubt about that. Self-driving cars are clocking up millions of miles, IBM Watson is diagnosing patients better than armies of doctors and Google Deepmind's AlphaGo beat the World champion at Go - a game where intuition plays a key role.

Deep Learning is very broad and complex and neural network forms an integral part of it. The further AI advances, the more complex become the problems it needs to solve. And only neural networks can solve such complex problems and that's why it's at the heart of Artificial intelligence.

Self-driving cars mark the biggest change in the automotive industry in the last 100 years. Every major car company is applying this technology and looking to hire talented individuals. Today, the machine learning algorithms are extensively used to find the solutions to various challenges arising in manufacturing of self-driving cars. With the incorporation of sensor data processing in a car, which is essential to enhance the utilization of machine learning to accomplish new tasks. The potential applications include evaluation of driving scenario classification through data fusion from different external and internal sensors like radars, cameras or the IoT (Internet of Things).

The algorithms used to train neural networks in a self-driving car model are classified as unsupervised and supervised algorithms. The Supervised algorithms make use of training on a dataset to learn and they continue to learn till they get to the level of confidence they aspire for (the minimization of the probability of error). The Unsupervised algorithms try to derive value from the available data, The reinforcement algorithms are another set of machine learning algorithms which fall between unsupervised and supervised learning

In the autonomous car, one of the major tasks of a machine learning algorithm is the continuous rendering of surrounding environment and forecasting the changes that are possible to these surroundings. These tasks are classified into 4 sub-tasks:

. The detection of an Object

. The Identification of an Object or recognition object classification

. The Object Localization and Prediction of Movement

Through this project, we are trying to implement some of the above features in a Remote Control Car so as to be familiar with computer vision and machine learning concepts that are applied to solving self-driving car problems.

Chapter 2

Literature Survey

2.1 Neural Networks

The human visual system is one of the wonders of the world. Consider the following sequence of handwritten digits:

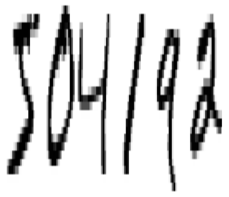

Figure 2.1: Digits

Most people effortlessly recognize those digits as 504192. That ease is deceptive. In each hemisphere of our brain, humans have a primary visual cortex, also known as V1, containing 140 million neurons, with tens of billions of connections between them. And yet human vision involves not just V1, but an entire series of visual cortices - V2, V3, V4, and V5 - doing progressively more complex image processing. We carry in our heads a supercomputer, tuned by evolution over hundreds of millions of years, and superbly adapted to understand the visual world. Recognizing handwritten digits isn't easy. Rather, we humans are stupendously, astoundingly good at making sense of what our eyes show us. But nearly all that work is done unconsciously. And so we don't usually appreciate how tough a problem our visual systems solve.

The difficulty of visual pattern recognition becomes apparent if you attempt to write a computer program to recognize digits like those above. What seems easy when we do it ourselves suddenly becomes extremely difficult. Simple intuitions about how we recognize shapes - "a 9 has a loop at the top, and a vertical stroke in the bottom right" - turn out to be not so simple to express algorithmically. When you try to make such rules precise, you quickly get lost in a morass of exceptions and caveats and special cases. It seems hopeless.

Neural networks approach the problem in a different way. The idea is to take a large number of handwritten digits, known as training examples,

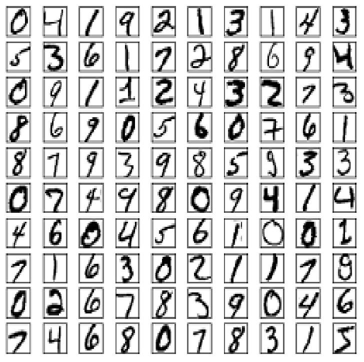

Figure 2.2: Training Set

and then develop a system which can learn from those training examples. In other words, the neural network uses the examples to automatically infer rules for recognizing handwritten digits. Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy.Just 100 training digits are shown above, perhaps we could build a better handwriting recognizer by using thousands or even millions or billions of training examples.

2.2 Self Driving Car

For generations, the automobile industry has been a source of innovation and economic growth. The ability to drive is a symbol of mobility and independence that spans generations. Clearly, automobiles play a significant role in our lives and afford many benefits to society.

Yet for all the benefits conferred on society, no other invention in the history of civilian technology has caused as much harm as the automobile. Every 30 seconds, someone dies in a traffic accident, adding up to well over 1 million deaths each year1. In the U.S., automobile accidents are the leading cause of death for people between the ages of 3 and 34. Moreover, human error is the cause of over 90 percent of automobile accidents.

In addition, the inefficiencies related with the automobile usage is staggering. Most automobiles sit unused more than 95 percent of their lifespan, and a freeway operating at maximum efficiency has automobiles on only 5 percent of its surface. In congested urban areas, 40 percent of all gasoline used is spent when cars circle to look for parking spaces4. Furthermore, in some U.S. cities, parking lots comprise more than a third of the land, becoming the single salient landscape feature of our built environment.

Data from the U.S. Department of Transportation also evokes concerns about quality of life. It estimates that people spend an average of 52 minutes of each working day commuting. The opportunity cost of this time is high, whether it is measured in lost productivity, the inability to spend more time with friends and family, or increased stress.

Autonomous vehicles could alleviate or completely solve these serious problems. The technology behind autonomous driving is typically divided into two categories: sensor-based implementation or a Connected Vehicle implementation. A sensor-based implementation is often an extension of current advanced safety features, while connected-vehicle technology (V2X) involves cars communicating with each other (V2V) and with infrastructure (V2I).

2.2.1 Existing Landscape

The existing landscape for autonomous vehicles consists of two parts:

- **Traditional players** are companies and industries already in the automotive business that are introducing autonomous features as a natural evolution of their product offering. Traditional players are commonly focused on incremental innovation. Major carmakers view driverless technologies as enablers to enhance the current driving paradigm and, more importantly, to preserve their existing business model. A gradual introduction of driverless technologies provides companies a long stream of premium-priced safety features that are consistent with current designs.

- **Disruptive players** are companies and industries that currently have no existing business model or revenue stream attached to the automotive industry. They typically favor pursuing innovation that moves directly to fully autonomous vehicles.

2.2.2 Traditional: Auto Manufacturers

The biggest players in the autonomous automobile industry as it exists today are major automotive manufacturers like Ford, GM, Toyota, BMW, Mercedes Benz, Audi and Volkswagen. Most manufacturers have recently introduced models with advanced safety features called advanced driver-assist systems (ADAS) that resemble partial self-driving capabilities. Several manufacturers have announced that they will release cars in the next two years that will be capable of driving themselves under certain conditions.

2.2.3 Traditional: Automotive Suppliers

Automobile manufacturers purchase components such as power train, electrical systems, and chassis for their vehicles from external suppliers. ABI research predicts that the global ADAS market will expand from 10Billion in 2011 to 130Billion in 2015, mainly due to the introduction of adaptive cruise control, lane-departure warning, and low-speed collision mitigation in non-luxury vehicles. Key suppliers of these technologies include Continental AG (one of the top five overall global OEM parts suppliers) and Hella and Bosch. All of these suppliers list technology for autonomous driving as one of their key strategic goals. For example, Continentals 2012 investor presentation lists safety as the companys primary megatrend. The company also issued a press release declaring the development of systems for automated driving to be one of the central themes of its long-term technology strategy5. Other key suppliers for ADAS and self-driving features include producers of microcontrollers (like Texas Instruments) and video decoders.

2.2.4 Disruptive: Google

The most deeply involved player in the autonomous automobile market from outside of the automobile industry is Google. Google has worked to develop autonomous cars for the past six years. CEO Larry Page approaches this as a big bet problem, noting the high accident rate, very low utilization of existing vehicles and the cost of car parking facilities6. Googles motivation for the car, as described by the lead developer of the project Sebastian Thrun are captured in the sidebar, Benefits of the Driverless Car

Googles stated mission is to organize the worlds information and make it universally accessible and useful. It is approaching autonomous vehicles as an opportunity to organize and process mapping and geographic information to many mobile computers - the vehicles themselves. Google sees this problem as parallel to the technical infrastructure and techniques required to organize very large, complex and rapidly changing data sets such as the Internet.

2.2.5 Disruptive: Research Programs

There are many other active research programs concerning autonomous vehicles, many of them featuring collaborations between universities and carmakers. Oxford University, for example, demonstrated a self-driving Nissan LEAF in 20127. Volkswagen and a research team from Stanford University have created a driverless Audi sports car, which has been zipping around US race tracks8. In another research project funded by the European Union, Volvo successfully drove a convoy of five vehicles that only had a human driver in the lead car.

2.2.6 Self-driving cars available NOW!

New features are typically introduced first in high-end automobiles before eventually trickling down to mainstream models. For the last 40 years, Mercedes Benz strategy has been to introduce innovations in their flagship S-Class (Sonderklasse or ”Special class”) first. Many features taken for granted today were first piloted in the S-Class, including padded interior in 1972, ABS 1978, airbag and seatbelt pre-tensioner in 1981, passenger airbag in 1988 and electronic stability control in 1995. The 1998 S-Class introduced Distronic, a cruise control system with sensors for measuring and maintaining the vehicle’s distance from the car in front of it.

The 2013 S-Class model is no exception to this 40-year-old strategy. It takes a decisive step toward autonomous driving, as it is capable of steering itself. This makes it the first to fulfill all the criteria of what is constituted as fully automated driving. However, it will do so only in congested traffic. When the vehicle is traveling at walking speed, the driver can choose to switch on cruise control and take his or her feet off the pedals and hands off the steering wheel. The car then automatically accelerates, brakes and steers. The Intelligent Drive: How it Works graphic highlights the technology subsystems involved to achieve self-driving status

2.2.7 Current Capabilities of Disruptive Players

Googles fleet of autonomous cars is continuously learning from each other through expanded digital mapping, and up-to-the second information from the Google cloud information ecosystem about road conditions, traffic and travel times. Googles model is simple: its vehicles communicate with other Google resources. In support of this approach, Google Streetview cars have driven more than 5 million miles across 50+ countries and in addition to taking high-definition photography have used light detection and ranging (LIDAR) technology to capture an incredibly detailed 3D map of the world to 15 cm resolution.

Google has also built the worlds largest traffic jam surveillance network by providing the operating system for some 500+ million smart phones. The mapping function on an Android device sends Google anonymous data on position and current speed that are used to calculate traffic flows. If multiple independent Android devices geo-positioned on a freeway that are traveling at 60 mph suddenly slow to a crawl, Google knows that traffic is particularly bad at that location. This information is tracked and trended over time. Additionally, as users travel from A to B using local knowledge to overrule a satellite navigation derived route, Google algorithms can learn and adapt to that better route and thinking process.

Chapter 3

Hardware Review

3.1 Arduino UNO

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

”Uno” means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino UNO is the optiboot bootloader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistortransistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

Figure 3.1: Arduino UNO

3.1.1 Power

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

* Vin. The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

* 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

* 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

* GND. Ground pins.

* IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

3.1.2 Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library

3.1.3 Programming

he Arduino Uno can be programmed with the (Arduino Software (IDE)). Select ”Arduino/Genuino Uno from the Tools ¿ Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP.

3.2 Optocoupler

In electronics, an opto-isolator, also called an optocoupler, photocoupler, or optical isolator, is a component that transfers electrical signals between two isolated circuits by using light.Opto-isolators prevent high voltages from affecting the system receiving the signal. Commercially available opto-isolators withstand input-to-output voltages up to 10 kV and voltage transients with speeds up to 25 kV/s.

A common type of opto-isolator consists of an LED and a phototransistor in the same opaque package. Other types of source-sensor combinations include LED-photodiode, LED-LASCR, and lamp-photoresistor pairs. Usually opto-isolators transfer digital (on-off) signals, but some techniques allow them to be used with analog signals.

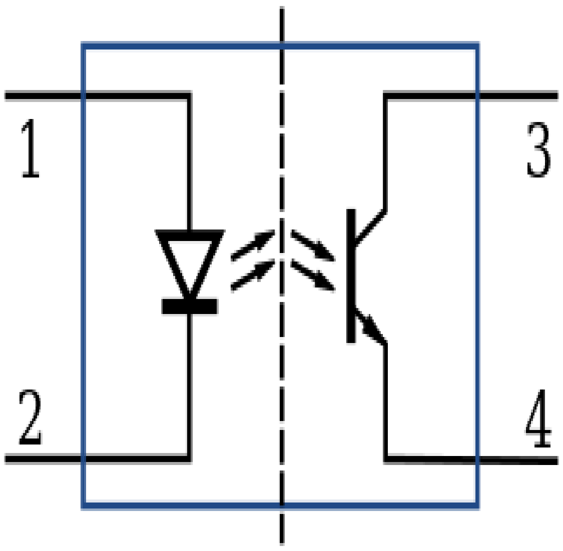

Figure 3.2: Optocoupler

3.2.1 Operation

An opto-isolator contains a source (emitter) of light, almost always a near infrared light-emitting diode (LED), that converts electrical input signal into light, a closed optical channel (also called dielectrical channel[7]), and a photosensor, which detects incoming light and either generates electric energy directly, or modulates electric current flowing from an external power supply. The sensor can be a photoresistor, a photodiode, a phototransistor, a silicon-controlled rectifier (SCR) or a triac. Because LEDs can sense light in addition to emitting it, construction of symmetrical, bidirectional opto-isolators is possible. An optocoupled solid-state relay contains a photodiode opto-isolator which drives a power switch, usually a complementary pair of MOSFETs. A slotted optical switch contains a source of light and a sensor, but its optical channel is open, allowing modulation of light by external objects obstructing the path of light or reflecting light into the sensor.

The optocoupler used in this project is IC 815

3.3 Remote Control Car

Since implementing the neural network on a real car is not practical for this project a remote control car is taken and its remote is so tweaked such that it could be controlled by a computer. The remote and computer are interfaced by arduino.

Figure 3.3: RC Car

Chapter 4

Programming Tools

4.1 Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

4.2 Pycharm

PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition released under a proprietary license - this has extra features.

4.2.1 Features

* Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes.
* Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages.
* Python refactoring: including rename, extract method, introduce variable, introduce constant, pull up, push down and others.
* Support for web frameworks: Django, web2py and Flask.
* Integrated Python debugger.
* Integrated unit testing, with line-by-line code coverage
* Google App Engine Python development
* Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with changelists and merge

Chapter 5

Working and Description

5.1 Block Diagram for Data Generation

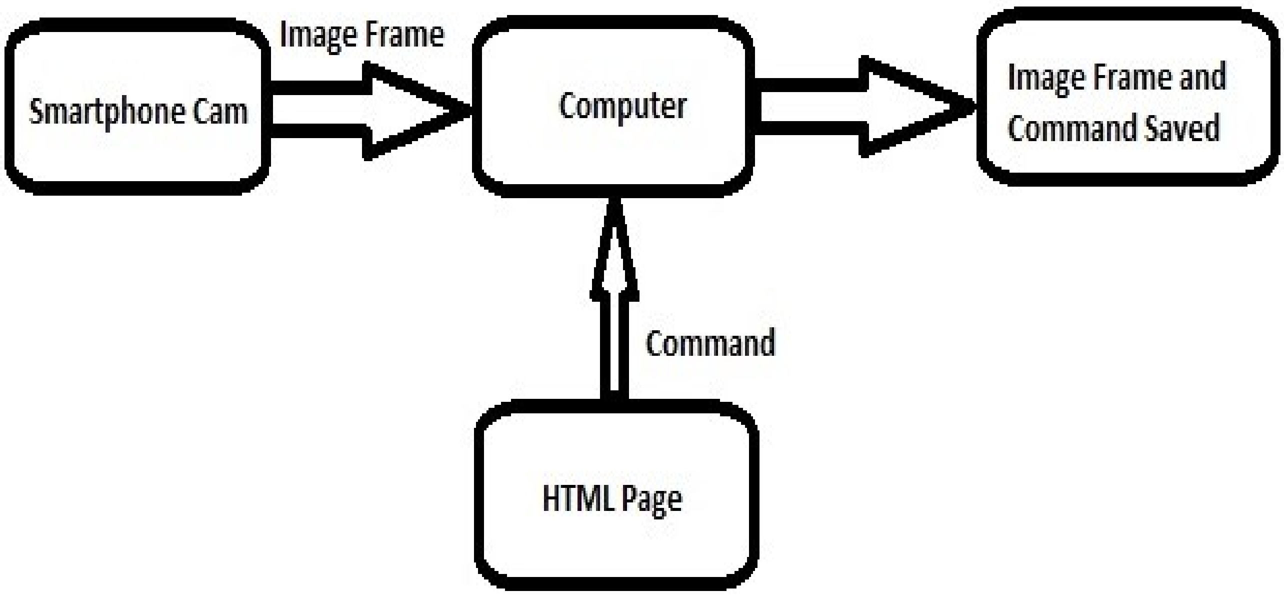

Figure 5.1: Block Diagram 1

5.2 Block Diagram for Self Driving

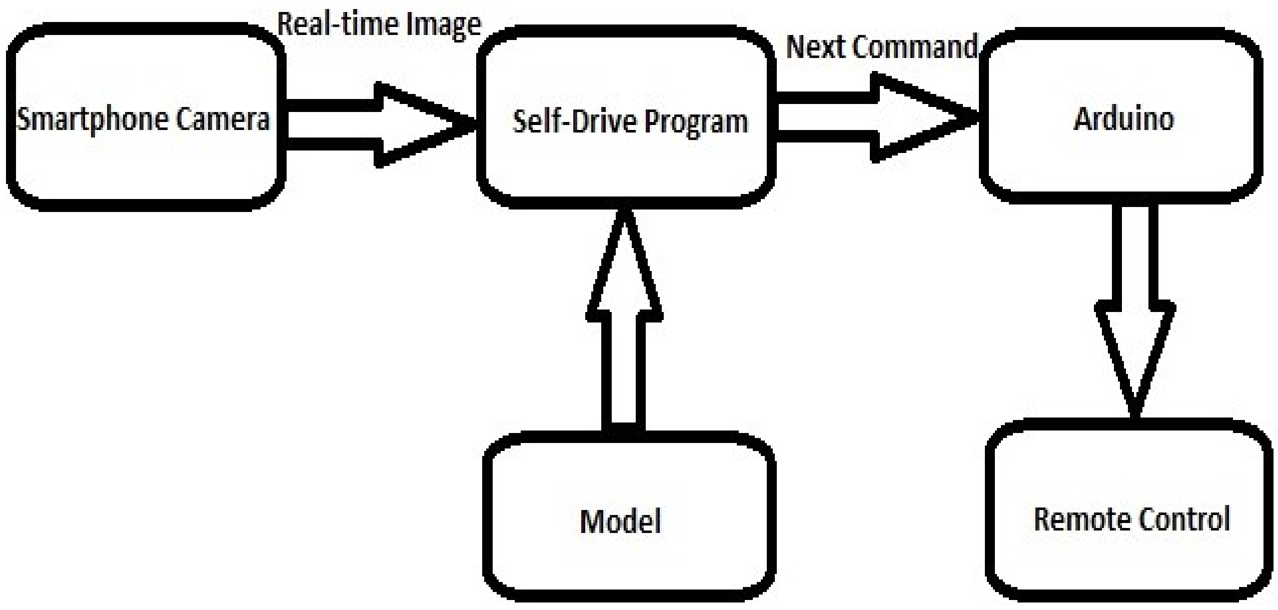

Figure 5.2: Block Diagram 2

5.3 Working

5.3.1 Data Generation

For any Neural Network to work efficiently it should be provided with sufficient data to work on.Hence in this project the first task at our hand is the process of Data Generation.For Data generation the components that we are going to use is the RC Car, Arduino,Smartphone and Computer.The remote control of the car is connected to the arduino so that the car could be controlled by the computer.The smartphone is connected to computer using WiFi.In the software part there is the data generation program which would be running in the computer and also an HTML code for controlling car with arrow keys.As the first step of data generation a track is made.The data generation program and HTML code is made to run in the computer.The user drives the car through the track.Each time when one of the arrow key is pressed the smartphone takes a photo and it is send through WiFi to the computer.The image frame on reaching the data generation program is named with a unique ID and the corresponding command the user gave for that image frame.Eg if the command was forward there would be a 0 with the name,for left it is 1 and for right it is 2.All the images thus obtained through driving the car by ourselves through the track is stored in a particular folder in the computer, thus we have the data set for our training program.The more the number of data we have the more efficient will be the working of neural network.

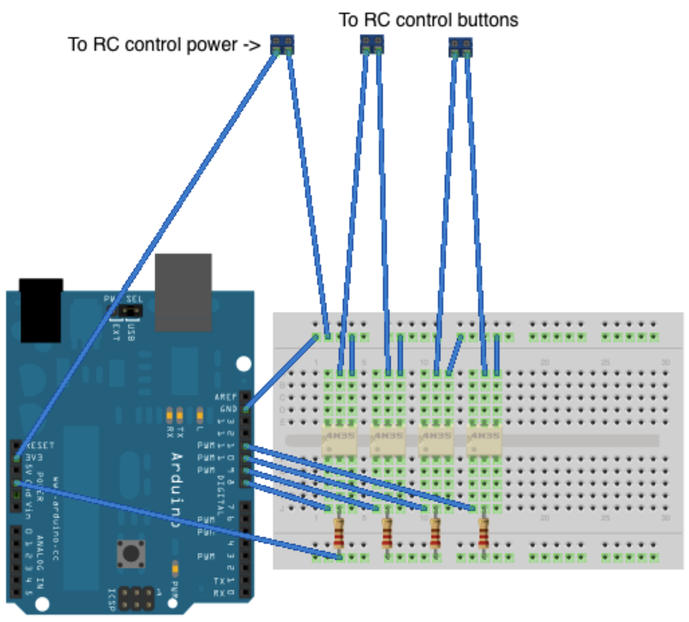

Figure 5.3: Remote Interfacing

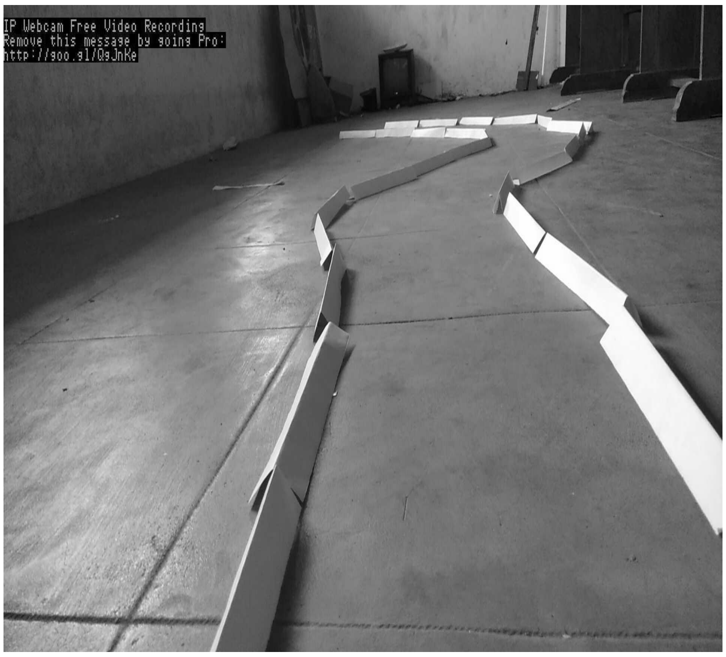
IP Webcam Free Video Recording
Remove this message by going Pro:
http://goo.gl/QgJnKe

Figure 5.4: Data Generation

5.3.2 Training

In the training part the training python program is made to run in the computer.The program takes the data set that we created as its input.The program then reduces the quality of the image by first converting it to black and white, then blurs the image, reduce the size and pixels.The output of the program is a model for the neural network.

5.3.3 Self Driving

In the self driving part the self driving program is made to run in the computer.When the self driving program is made to run in the computer, the smartphone takes a photo and sends it to the program.The program then gives this photo as the input to the neural network.The neural network then predicts the apt command for the image frame received and this command is passed to the remote control of the car through arduino.

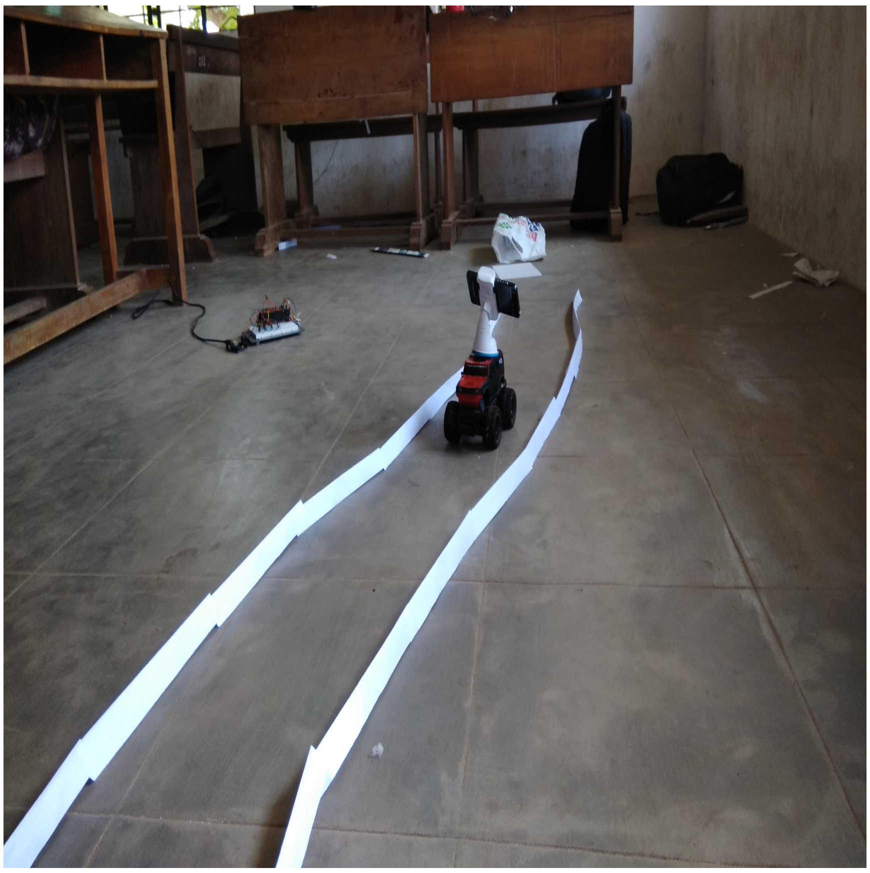

Figure 5.5: Self Driving

Chapter 6

Programming

6.1 Flowchart

6.1.1 Data Generation

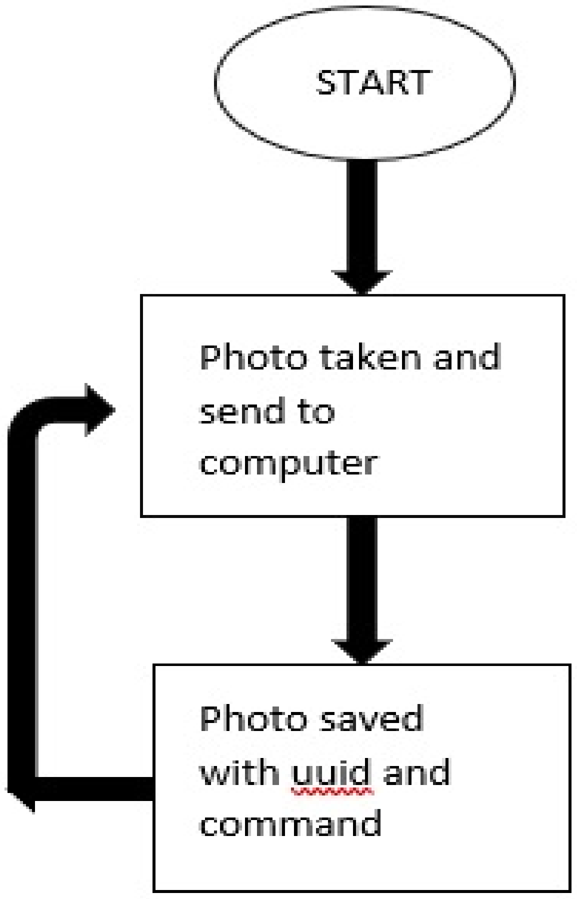
START
Photo taken and send to computer
Photo saved with uuid and command

Figure 6.1: DataGen Flowchart

6.1.2 Self Drive

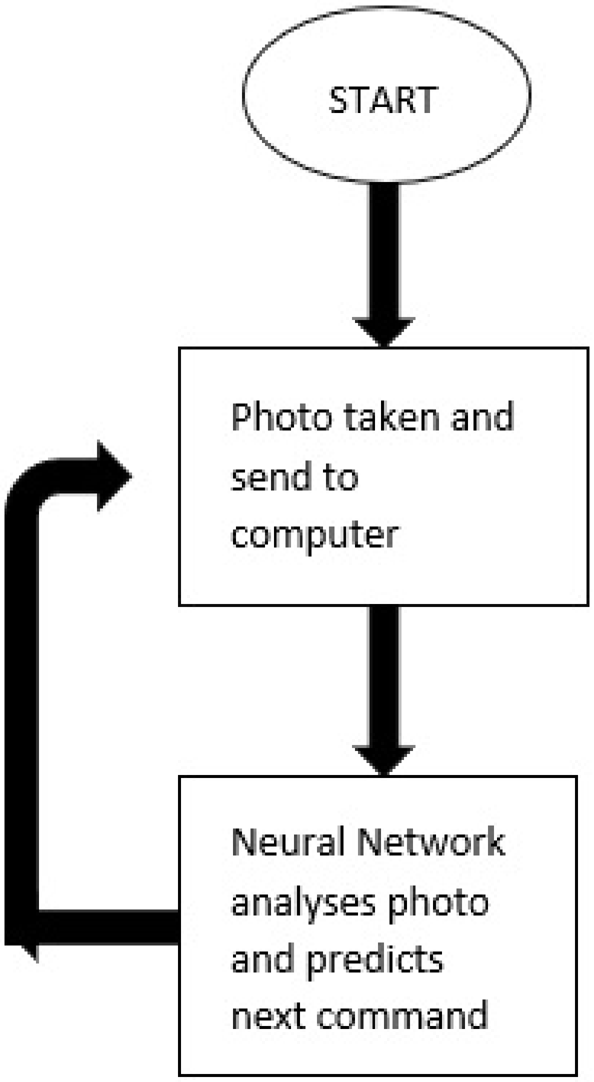
START
Photo taken and send to computer
Neural Network analyses photo and predicts next command

Figure 6.2: Self-Drive Flowchart

6.2 Program

6.2.1 Data Generation

```python
from serial import Serial
import requests
from PIL import Image
from io import BytesIO
import time
import uuid
import cv2
import numpy as np
arduino = Serial('COM3', 250000)
#wait for the serial port to be ready
time.sleep(2)

from flask import Flask
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

# check if everything is working

arduino.write(b'forward_on\n')
time.sleep(0.05)
arduino.write(b'forward_off\n')
time.sleep(0.05)
arduino.write(b'backward_on\n')
time.sleep(0.05)
arduino.write(b'backward_off\n')
time.sleep(0.05)
arduino.write(b'left_on\n')
time.sleep(0.05)
arduino.write(b'left_off\n')
time.sleep(0.05)
arduino.write(b'right_on\n')
time.sleep(0.05)
arduino.write(b'right_off\n')



def save_data(command):
    # request the image many times
    # for some reasons the app doesn't always return
    # the latest image if you request it only once   \_(   )_/
    for _ in range(0, 1):
        response = requests.get('http://192.168.43.1:8080/shot.jpg')
        # save the image for future use
        # the direction is stored at the end of filename
        Image.open(BytesIO(response.content)).convert('L').save('C:/Users/John
            Doe/d-ff/Desktop/sdcfinal/right/{}_{}.jpg'.format(uuid.uuid1(),
            command))
    ''' img = cv2.imread('Image.jpg')
```

```
        img = cv2.blur(img, (5, 5))
        retval, img = cv2.threshold(img, 140, 255, cv2.THRESH_BINARY)
        img = cv2.resize(img, (25, 25))
        image_as_array = np.ndarray.flatten(np.asarray(img))
        cv2.imshow('img',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()'''

@app.route("/forward")
def forward():
    save_data(0)
    arduino.write(b'forward_on\n')
    time.sleep(0.15)
    arduino.write(b'forward_off\n')
    return 'forward'

@app.route("/left")
def left():
    save_data(1)
    arduino.write(b'left_on\n')
    time.sleep(0.5)
    arduino.write(b'forward_on\n')
    time.sleep(0.15)
    arduino.write(b'forward_off\n')
    time.sleep(0.5)
    arduino.write(b'left_off\n')
    return 'left'

@app.route("/right")
def right():
    save_data(2)
    arduino.write(b'right_on\n')
    time.sleep(0.5)
    arduino.write(b'forward_on\n')
    time.sleep(0.15)
    arduino.write(b'forward_off\n')
    time.sleep(0.5)
    arduino.write(b'right_off\n')
    return 'right'

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

6.2.2 Training

```python
import numpy as np
from os import listdir
from os.path import isfile, join
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import cv2
from sklearn.externals import joblib
X = []
y = []

# load all the images and convert them
files_name = [f for f in listdir('data') if isfile(join('data', f)) and f != '.
    DS_Store']
for name in files_name:
    try:
        # load the image
        print (name)
        img = cv2.imread(join('data', name))
        print (img)
        # blur to remove details
        img = cv2.blur(img, (5, 5))
        # convert to binary
        retval, img = cv2.threshold(img, 210, 255, cv2.THRESH_BINARY)
        # resize to improve performance
        img = cv2.resize(img, (24, 24))
        # convert to array
        image_as_array = np.ndarray.flatten(np.array(img))
        # add our image to the dataset
        X.append(image_as_array)
        # retrive the direction from the filename
        y.append(name.split('_')[1].split('.')[0])
    except Exception as inst:
        print(name)
        print(inst)

# split for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
    random_state=42)

# scale the data
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)


clf = MLPClassifier(solver='lbfgs', alpha=100.0, random_state=1,
    hidden_layer_sizes=50)
clf.fit(X_train, y_train)
print('score: ', clf.score(X_test, y_test))
```

```
clf = MLPClassifier(solver='lbfgs', alpha=100.0, random_state=1,
    hidden_layer_sizes=50)
clf.fit(X, y)
print('score2: ', clf.score(X, y))
joblib.dump(clf, 'Model.pkl')
```

6.2.3 Self Drive

```
from skimage import io
import numpy as np
from sklearn.externals import joblib
from serial import Serial
import time
import cv2
import requests
from PIL import Image
from io import BytesIO
import time
import uuid
#from urllib import urlretrieve

arduino = Serial('COM3', 250000)
time.sleep(2)

arduino.write(b'forward_on\n')
time.sleep(0.1)
arduino.write(b'forward_off\n')
time.sleep(0.1)
arduino.write(b'backward_on\n')
time.sleep(0.1)
arduino.write(b'backward_off\n')
time.sleep(0.1)
arduino.write(b'left_on\n')
time.sleep(0.1)
arduino.write(b'left_off\n')
time.sleep(0.1)
arduino.write(b'right_on\n')
time.sleep(0.1)
arduino.write(b'right_off\n')

CAMERA_URL = 'http://192.168.43.1:8080/shot.jpg'
ARDUINO_SERVER = 'http://localhost:5000'

clf = joblib.load('model.pkl')
print('model loaded')

def send_command(result):
    if result == '0':
        arduino.write(b'forward_on\n')
        time.sleep(0.05)
        print (result)
        arduino.write(b'forward_off\n')
    if result == '1':
        arduino.write(b'left_on\n')
        time.sleep(0.5)
        print (result)
        arduino.write(b'forward_on\n')
        time.sleep(0.05)
        arduino.write(b'forward_off\n')
        time.sleep(0.5)
        arduino.write(b'left_off\n')
    if result == '2':
        arduino.write(b'right_on\n')
        time.sleep(0.5)
        print (result)
        arduino.write(b'forward_on\n')
        time.sleep(0.05)
        print (result)
        arduino.write(b'forward_off\n')
        time.sleep(0.5)
        arduino.write(b'right_off\n')

def drive():
    response = requests.get(CAMERA_URL)
    print ("image send")
    Image.open(BytesIO(response.content)).convert('L').save('C:/Users/John Doe/d
       -ff/Desktop/sdcfinal/selfdrive/img.jpg')
    img = cv2.imread('img.jpg')
    print ('succes')

    cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.blur(img, (5, 5))
    retval, img = cv2.threshold(img, 210, 255, cv2.THRESH_BINARY)
    img = cv2.resize(img, (24, 24))
    retval, img = cv2.threshold(img, 210, 255, cv2.THRESH_BINARY)
    image_as_array = np.ndarray.flatten(np.array(img))
    result = clf.predict([image_as_array])[0]
    print(result)

    send_command(result)

    time.sleep(0.5)
    drive()

print('start driving')

drive()
```

Chapter 7

Advantages of taking this project

By taking up a project in machine learning one of the latest technologies in the current world. We were able to understand the concept and working of machine learning technology.By working in this project we were able to make an understanding of python programming.By learning new things for the project we were able to equip us with the necessary tools for a career in machine learning.We understood how a neural network works, how it learns, how the amount of data given to it for training has a direct effect on how efficiently it does the work it is assigned to do.We consider this project as a stepping stone to the big world of machine learning and Artificial intelligence.

Chapter 8

Conclusion

The aim of the project was to implement a neural network on an RC car so that it could self drive on any track. The system built for this task, although relatively simple, worked well and helped to learn various machine learning tools used in the present time. The results showed that car could self drive with little error in its judgement of the next command if sufficient data set is given for training it can perform well in any new track.

The project demonstrated that while attempting to self drive the car with little data set the self driving is prone to errors frequently, which could be reduced considerably by increasing the data set on which it trains.But with large data set also there is latency in the car response, which is caused by network delay in WiFi.

In the end the car was able to self drive with little errors on many new tracks.These errors are reduced by increasing the data set for training the neural network.The latency can reduced by video streaming rather than image streaming.

References

[1] *A Simulation for Comparing Training and Evolving of Agents*;Christopher Davenport

[2] *A Guide to Recurrent Neural Networks and Backpropagation*;Boden

[3] *A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, Vol. 2, No. 1, March 1986, pp. 1423. Artificial Intelligence Laboratory, Massachusetts Institute of Technology.*;Brooks, R.

[4] *Using Backpropagation Algorithm for Neural Adaptive Control: Experimental Validation on an Industrial Mobile Robot. Theory and Practice of Robots and Manipulators (ROMANSY 11), Udine, Italy, July 1996*;Henaff, P., Delaplace, S.

[5] `https://medium.com/@florianherrengt/building-a-basic-self-driving-rc-car-bca6a7521753`

[6] `https://faculty.washington.edu/jbs/itrans/self_driving_cars[1].pdf`

[7] `http://scikit-learn.org/stable/tutorial/basic/tutorial.html#machine-learning-the-problem-setting`

[8] `https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html#smoothing`