University of Mumbai

Product Classifier using Embedded Systems and Machine Learning

Submitted in partial fulfillment of requirements

For the degree of

Bachelor of Technology

by

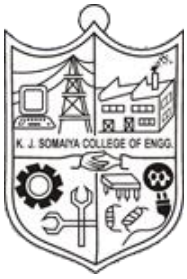
Hamza Sunasra (1613110)
Trivikram Umanath (1613125)
Rahul Singh (1613127)

Guide

Prof. Nitin Nagori

Co-Guide

Prof. Akshata Prabhu



Department of Electronics and Telecommunication Engineering

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Batch 2016 -2020

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate

This is to certify that the dissertation report entitled **Product Classifier using Embedded Systems and Machine Learning** submitted by –

1. Hamza Sunasra (1613110)
2. Trivikram Umanath (1613125)
3. Rahul Singh (1613127)

is a bona fide record of the dissertation work done under the guidance of Prof. Akshata Prabhu and Prof. Nitin Nagori of Department of Electronics and Telecommunication in partial fulfillment of the requirements for the Bachelor of Technology Degree in Electronics and Telecommunication Engineering of University of Mumbai

Prof. Nitin Nagori	Prof. Akshata Prabhu
Guide	Co- Guide
Dr. Shubha Pandit	Prof. Dr. Ameya K. Naik
Principal	Head of the Department

Date: 14 /06/2020

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate of Approval of Examiners

We certify that this dissertation report entitled **Product Classifier using Embedded Systems and Machine Learning** is bona fide record of project work done by –

1. Hamza Sunasra (1613110)
2. Trivikram Umanath (1613125)
3. Rahul Singh (1613127)

This project is approved for the award of Bachelor of Technology Degree in Electronics and Telecommunication Engineering of University of Mumbai.

_____

Internal Examiners

_____

External/Internal Examiners

Date: 14/06/2020

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

DECLARATION

We declare that this written report submission represents the work done based on our and / or others' ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

_____ **Signature of the Student** **Roll No. 1613110**	_____ **Signature of the Student** **Roll No. 1613125**
_____ **Signature of the Student** **Roll No. 1613127**	

Date: 14/06/2020

Place: Mumbai-77

Abstract

Object classification finds many applications in our day to day life. One such application is identification of FMCG products that are sold in stores. Barcode reading is currently the most employed method for billing and such, but suffers from many insufficiencies. Thus, machine learning can be a better alternative for the same.

In this dissertation, we implement one such Classification Algorithm using Convolution Neural Network (CNN), for classification of products. The scope of the products to be used has been narrowed down to soft drinks alone for better precision, and to add on to the extensiveness and complexity of the model. The dataset is, hence, prepared based on soft drinks alone. The dataset is sorted from a previously available dataset based on retail products. Over here, Picamera captures images and relays the data to the Raspberry Pi, where the previously trained model for products, to be evaluated or classified, has been saved beforehand. The Pi then predicts what the product is, and takes the required decision. The model was designed using CNN based on the dataset collected previously. Further, we design an automated bot that is capable of grabbing and dropping the products. This bot has the Pi Camera attached to it to take pictures on the go. It is essentially a line follower bot , and moves along a predetermined black line in order to reach its destination. It has a customized pair of claws, that act as the gripper, and help in grabbing and dropping the products.

Keywords: Machine Learning, CNN, Dataset, Raspberry Pi, Tensor Flow, Bot, Gripper, Chassis, Line Follower

Contents

List of Figures

List of Tables

Chapter 1

Introduction:

This chapter presents the general introduction of the project and the concept of design and working in a brief manner. The need of developing the idea and the expected outcome of the project is also discussed.

1.1 Background:

Machine Learning is a process of mapping the inputs to an algorithm and hence, predicting the output. It involves training machines or computers to learn certain functions on their own without human involvement. Now, machine learning can be broadly divided into two types, typically based on their functionalities, into either a Classification one or a Regression one. Classification Algorithms involve classifying data into two or more categories, based on certain properties or a specific pattern observed in their behaviour. But to do so, input must be fed to the machines to gather or note the behavioral patterns to be observed in the data. This is called training of machines, where a large dataset is fed to the machines and the machine comes up with its own pattern of identifying a pattern and weaving an algorithm. The way in which the machine is supposed to learn depends on the algorithm used. To test the accuracy of the pattern observed, another dataset, with the predetermined output, is fed to the system, and accuracy is determined. This data used is called the testing dataset. Overall, input and output is fed to the system and the machine is expected to come up with its own algorithm to solve the respective problem. Convolution Neural Network is a branch of deep learning, where the operation of convolution is used in place of matrix multiplication. The advantage lies in forming patterns based on hierarchy of data in consecutive layers. All the nodes are connected to all the nodes in the following layer.

1.2 Motivation:

Barcode readers are dominating the field of scanning and identifying products. They are pretty convenient, as all of the information regarding the products can be retrieved by simply scanning a given code. Thus, it finds a huge application in the commercial sector. It eliminates the involvement of humans and hence, also drastically reduces the scope of human errors. It is a pretty reliable technique which is too efficient, to not be used. The ease with which it is operated also favours its further use in the industry.

No matter how good the barcode system seems, it also has its own drawbacks and tradeoffs. For example, it requires a special device at both the ends i.e. a scanner and a special output port or terminal. No modification of the products is possible using the barcode. Barcodes that are damaged may be of no use as they cannot be processed for further uses. It also requires for the code to be scanned perfectly, with it lying perfectly in the line of sight. Skilled labour is also a requirement for the same. Production and time to time updation of the products in the database of the barcode system for it to be recognised is also a tedious work.

1.3 Scope of the Project:

The project implemented here tries to eliminate the drawbacks of Barcode Scanning by using Machine Learning as a tool of identification and classification. The advantages of using these ML Algorithms is that they take into account the finer details or aspects of images to be classified. These annotations range from trivial features like curves, edges etc. to more complex and detailed features. The more layers we keep on adding, the better the system gets for distinguishing characteristics. Rather than simply relying on the already fed data in the database, Learning Algorithms can swiftly recognize products not based on a code, but the actual live product. Since barcode scanning is vulnerable to forgery and theft, a real life analysis of the product is more suitable in such scenarios. Not to mention, there is always a need for automation to reduce the workload on humans by using bots as a substitute

for many tasks. Once a bot successfully learns the tasks, the time and resources spent on humans can be diminished, and the same goes for the scope of errors while performing these tasks.

1.4 Challenges and Improvements:

Preparation of Dataset:

An extensive dataset spanning over hundreds of images for each product is required for the classifying algorithm to learn. Preparation of such a dataset is infeasible because scanning, uploading and training these many images will consume too much time as well as resources.

To improve the same, we can come up with a threshold minimum amount of images required to efficiently train the model. A faster mechanism for scanning and feeding the images can be developed. A well sought out database should be formed where new products can be added on the go without much hassle. Instead of training for the entire set, only the results of the new ones must be added to the pre existing models.

Movement of Bot:

As of now, the bot simply follows a black line to deliver/ pickup products. To add more variations, we can add an obstacle sensor so that it can avoid running into things that may hinder its path. Also, the bot is simply capable of picking and dropping products across a single plane/ height. To overcome that, a mechanism which would allow the claws/ gripper to move vertically can be added.

Linear Range of Motion:

The bot is nothing more than a prototype, as we speak. It has been programmed to just pick up objects and drop them currently. It lacks the ability to implement instructions dynamically. So, an online queue which would get filled with queries of commands for the bot to be followed, can be implemented. An android app can be developed for the same, wherein the instructions provided by UI would be logged in the query list, and be expected to be dutifully completed by the bot in real time. These instructions or queries we speak of may consist of which products to handle, and whether to pick them up or drop them.

1.5 Brief Description of the Project:

This project attempts at implementing a similar Learning Algorithm for classifying products, rather than having their barcodes scanned. Over here, we train a model using images of products that were captured using cameras, all set at different angles. The dataset consisted of labeled images that were used to get the predicted values. This dataset is trained using CNN and hence, our model is prepared. This model can then be exported to various devices, where the only thing required would be to capture images. The captured images can be processed to suit the specifications of the trained model. This preprocessed image would be fed to the model, which would classify the image as per its history of data. The output of this model, as classified by the algorithm, can then be used by the bot for carrying this product and doing the necessary task, as instructed by the programmer.

1.6 Organization of the Report :

The organization of the report is done in a structured manner to explore the issue, ignite a possible solution, use of better, state of the art technology to achieve so, the design metric, implementation techniques. It also goes on to explain how this can be further improved by including unprecedented solution, which seem beyond the scope as of now. To do so, the report has five distinct chapters:

- The Introductory chapter (Chapter 1) gives the basic introduction to the present scenario for problems faced, the need for bringing up this thesis, a brief description of how to go about the solution, and other such challenges faced while implementing the same.
- Chapter 2 describes the materials used for reference. It goes in detail regarding the sub systems or what exact parts were referred from these papers. An attempt was made to word a few topics learnt. A brief idea about each sub system is provided in this chapter.
- Chapter 3 deals with the project design which includes the flow of the project (flowchart), block diagram of the system, the hardware components used and their detailed descriptions, the software design and tools used, the objectives we kept in mind while designing the system.
- Chapter 4 provides details of implementation and experimentation done.
- Finally, chapter 5 provides the conclusion drawn after working on the project. It also includes the future scope of the project.

Chapter 2

Literature Survey

This chapter presents the literature review done concerning the project design. It would include accomplished sub goals, problems encountered in the design and implementation, resources consumed and estimation of various other parameters. Literature survey would be a useful key resource in the idea development and would prove helpful in reducing redundant parameters.

One of the research papers under survey is Image Classification using Deep Learning by M Manoj Krishna,M Neelima,M Harshali,M Veenu Gopal Rao.

This paper gives an in depth analysis of how we can classify images on the basis of different algorithms.It gives a proper and apt definition of what is Classification.

The motive of the paper was to explore the study of image classification using deep learning.

The paper speaks about how we can use Artificial Neural Networks for image classification,these are networks of small neurons which are nothing more than a set of inputs, a set of weights, and an activation function. The neuron translates these inputs into a single output, which can then be picked up as input for another layer of neurons later on.This ANN is also called as Convolution Neural Network.

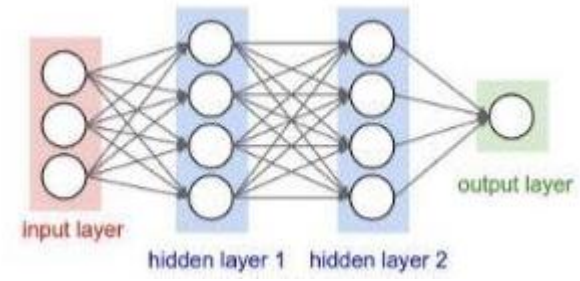


Fig i: Basic Deep Neural Network

Training the data in the networks by giving an input image and conveying the network about its output. Neural networks are expressed in terms of number of layers involved for producing the inputs and outputs and the depth of the neural network. Neural networks are involved in many principles such as fuzzy logic, Bayesian methods.These layers are generally referred to as the hidden notes and number of input and output it has.

The paper gave a concrete definition between deep learning and machine learning.The Machine learning neural networks has three layers only,input , hidden and an output layer. While a deep learning neural network has more than one hidden layer.

The paper further classifies Convolution Neural Networks intoLenNet and AlexNet which could also be used for image classification. The Lenet is expressed as a shallow Cnn which comprises 2 convolutional layers, 2 subsampling layers, 2 hidden layers and 1 output layer.The AlexNet comprises 5 convolutional layers, 3 sub sampling layers and 3 fully connected layers. While both of them are a form of CNN they differ in the way they extract features from an image . We use the non-linearity in the Feature Extractor module in AlexNet whereas Log sinusoid is used in LeNet. AlexNet uses dropout which is not observed in any other data sets of networking.

The paper concludes with testing four images using deep learning using the Alexnet architecture for classifying images,The classification is carried out for even for the portion of test images and it shows the effectiveness of deep learning.

2) The second research paper under survey is Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using Tensorflow and Keras by Karan Chauhan, Shrwan Ram.

In this paper, four different structures of CNN are compared on the CPU system, with a combination of different classifiers and activation functions, namely softmax, sigmoid classifiers and Relu, Tanh activation functions. For computation and processing we are using Tensorflow and Keras framework. Tensorflow is one of the libraries used for image classification in deep learning. Tensorflow is an open source software library developed by Google in 2015 for numerical computation. Keras is an open source neural network library written in python, it is capable of running on top of MxNet, Deep learning, Tensorflow, and Theano. It is designed to enable fast experimentation with deep neural

networks. The first section of this paper contains general introduction about deep learning, tensorflow, keras and dataset. Second section contains basic theory about CNN, classifiers and activation functions. Third section of this paper contains literature review, research methodology, and the final section contains experimental setup and results.

It gives us an apt definition of what a CNN is.

Convolutional Neural Network (CNN): Convolutional Neural Network is a special type of feed forward artificial neural network, which is inspired by the visual cortex. In CNN, the neuron in a layer is only connected to a small region of the layer before it, instead of all the neurons in a fully connected manner, so CNN handles fewer amounts of weights and also less number of neurons.

Relu Activation Function: Relu F(x) = max (x, 0), is mostly used in deep learning activation function, for hidden layers. A rectified linear unit has output „0" if the input is less than „0", and raw output otherwise". Relu is the simplest non-linear activation function. Research has shown that relu results are much faster for large networks training. Most frameworks like tensorflow, make it simple to use relu on hidden layers. Fig. 2.3 Relu activation function C.

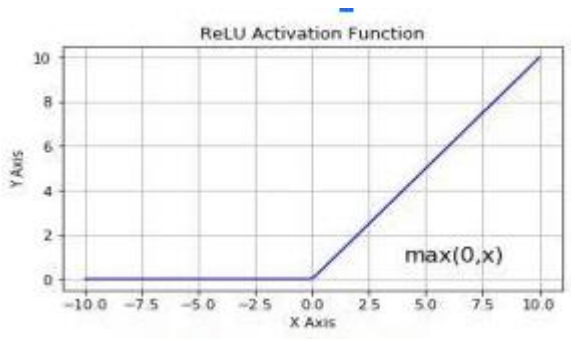


Fig ii: ReLU Activation function

Tanh activation function: Tanh function [tanh(x) = (ex - e -x) / (ex + e-x)] produces output in range of -1 to +1. It is a continuous function, which produces output for every „x" value. Fig. 2.4 Tanh activation function D.

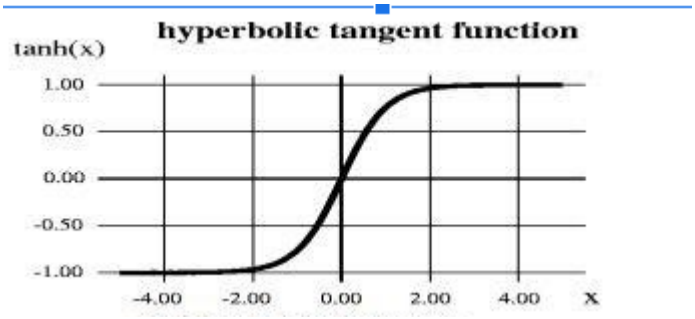


Fig iii: Tanh Activation function

Sigmoid classifier: Sigmoid classifier [F(x) = 1/ (1 + e-x)] takes any range of real numbers and returns the output value which falls in the range of „0"to „1". It produces the curve in „S" shape. Sigmoid classifier is mainly used for binary data classification. E.

Softmax classifier: The softmax classifier [F(x) = exi / (∑k j=0 e xj)] squashes the outputs of each unit to be between 0 and 1, just like a sigmoid classifier. But it also divides each output such that the total sum of the outputs is equal to 1. The output of the softmax classifier is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true. Softmax classifier is used for multiple data classification.

The paper gives us a flow chart of all the steps of the proposed method. . Each block of proposed flow diagram is clearly labeled and represents processing steps. Using this methodology, we compare four different structures of CNN, with four different combinations of classifiers and activation functions.

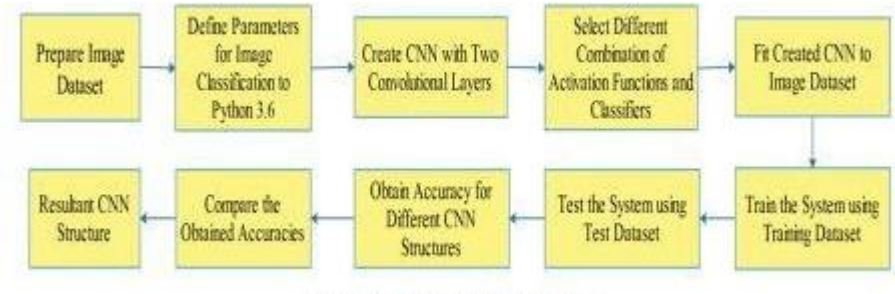


Fig iv: Proposed Methodolgy

Another research paper, discussing the basic implementation of a line follower bot, was surveyed. It has been titled “Line Follower Robot” by the authors Abhijit Kalbande and Miss Shraddha O. Koche.

This paper explains the three basic steps involved in the implementation of a line follower robot. These steps include understanding the crux of a line follower bot, ways to sense the lines to be traversed by the bot and the role of motors in doing so.

To begin with, we are introduced to the basic operation of the bot, and that is the working of IR senors to relay over the message to the motor driver.

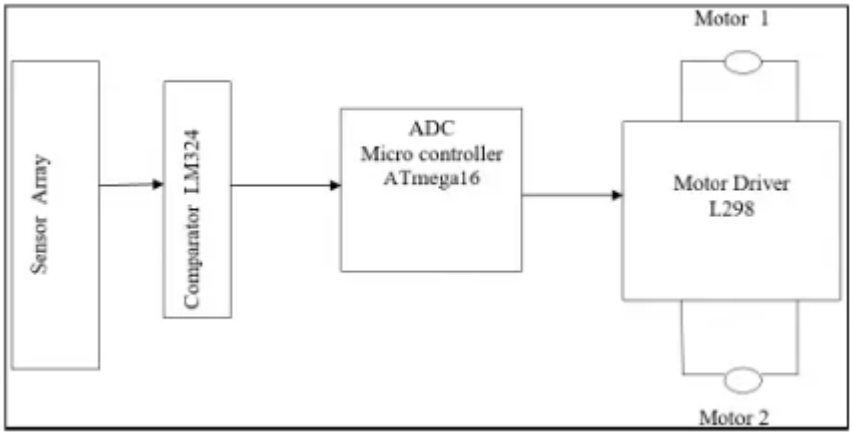


Fig v: Block Diagram of Robotic Car

Over here, they have generalised the steps of the bots in the following 2 steps:

1) Using the optical sensors to decide the position of the line. These sensors are supposed to be attached at the front end of the bot. This paper emphasizes on using a photocoupler to do the same. It also stresses on the fact that this sensing of the lines must be done with great precision and robustness.
2) To steer the bot in the desired direction, it must be in touch with the sensing action. The steering is brought about by the use of motors which have been attached by the wheels.

The paper then proceeds to explain the role of an IR sensor in the said project. It goes on about the photocoupler configuration of the sensor, which is made by the IR detector and IR transmitter, both in the forms of an IR LED and a phototransistor, respectively. The infrared phototransistors acts as a transistor with the base voltage determined by the amount of light hitting the transistor. Hence it acts

as a variable current source. Greater amount of IR light causes greater currents to flow through the collector emitter leads. The variable current traveling through the resistors causes a voltage drop in the pull up resistor. The voltage is measured as the output of the device. A photodetector.it detects ir energy emitted by the emitter and converts it into electrical energy. The main principle involved in the conversion of light energy to electrical energy is photoelectric effect.the output is taken at negative terminal of IR detector.

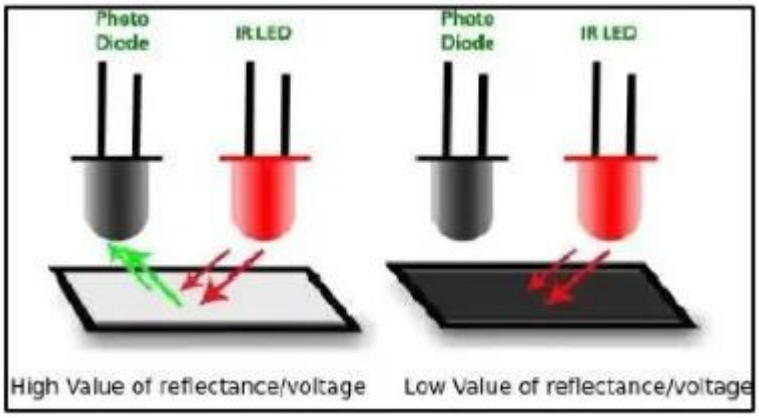


Fig vi:Working of IR Sensor

The motor driver, IC L293D, is then explained, and how it can be used for driving our motors. The working of the motors for three directions, that is taking a left turn, taking right turn and moving straight is explained. It is then concluded with the advantages and disadvantages.

Advantages

- Robot movement is automatic.
- Fit and forget system.
- Used for long distance application.
- Defense applications.
- Simplicity of building.

Disadvantages

- LFR follows a black line about 1or 2 inches in width on a white surface.
- low speed and instability on different line thickness or hard angles.

Chapter 3

Project Design

This chapter presents the design methodology adopted to satisfy the problem statement, which will lead to the development of the final system. It includes the design of both software and hardware fundamentals required to run the project as a whole. Choosing the correct platform for software design to get desired program flow and selecting the right components to ensure smooth working is all a part of design.

3.1 Introduction:

The barcode scanning can be a tedious procedure at times. In dim or dull places, where the line of sight might get obstructed, scanning might get difficult. In such scenarios, the clerk might have to manually enter the small numbers at the bottom of the code. In cases where the entire code has been damaged, there would be no other to evaluate or bill the product. Also, production cost of these barcodes is quite high, since a regular printer might not be able to print such finely detailed codes and lines.

Thus, we eliminate the use of the entire barcode systems by simply scanning the entire product instead of a small barcode representing the product. Real time evaluation of the live product is also helpful in the sense that it can scan multiple products at once. It also gets rid of humans at terminals to scan products.

3.2 Problem Statement:

The inefficiencies of the barcode system, as mentioned earlier, can be overcome by using Learning Algorithms. Further operations can be added to augment the applications of such a system. There is a need for an automated system for inventory management to reduce the scope of human inefficiencies. Thus, our aim is to successfully implement an object detecting system which can classify products, and implement an automated bot that can help in picking and dropping the said product.

3.3 Dataset Curation:

In our project we have considered three different products and we used Pi Camera to click many images of each of the products. We have for this project considered the background to be constant as it is in every shopping market. We used an iterative loop coded in Python to click a 1000 images of each and every product.As seen below we have set the resolution of the camera to 640*480 and framerate to 80. In a loop we capture a 1000 image of the same product to make our very own custom made dataset. We can change the name to Product1 to Product2 and so on and we can make our very own dataset which will be further used in Machine Learning and Deep Learning.

There are three products we have considered and a tester folder is made only for testing purposes i.e during recognition the product's image is captured and saved in this folder.

Given below are the three products which are considered.

Fig vii: Product 1

Fig viii: Product 2

Fig ix: Product 3

As seen in the above images we have considered three different products of completely different product categories and the background is kept the same for the project as we assume to make our own

dataset for every shopping market depending on the background but in general since the background is always white we have considered this in our project.

3.4 Selection of the Algorithm

To implement our image classification, there were several options in terms of the algorithm to be used. Now, to solve this dilemma, the entire dataset was trained for each of the classification algorithms known to us, and the one with the highest accuracy was used as the final algorithm. The dataset consists of the pixel values for each channel for all of the images, where the rows represent the image number whereas the columns represent the pixel number. To proceed with the testing, the dataset was divided into training and validation(testing) parts. The training data is fed to the algorithm for it to train on, and come up with its own model (reinforcement learning). This fitting is later verified by using the said model to classify the data stored in the validation dataset, and check for its accuracy. The training is usually done for several variations in the parameters in order to bring out the best of the algorithm. Another thing to be noted is the use of cross validation for training, and computing its accuracy. This ensures that the ratio of imbalance (of the classes or labels) is maintained even while training, to give us an accurate representation of the training data.

The training phase includes giving in the training input data and output data where the mapping and some rules or logic and sense is made of the relationship between the input attributes and the output values.These same rules are then used in the testing phase where the predictions are made on the testing data and compared with the testing output to find the accuracy.The accuracy of an algorithm signifies the performance of an algorithm.

Now, the algorithms used were :

1. Gaussian Naive Bayes:

Gaussian Naive Bayes is a variant of Naive Bayes, where the data constituting the dataset is in continuous values, and not discrete terms. The Naive Bayesian approach is based on calculating the probabilities of previously occurring events for each of the class labels, and then using the Bayes

Theorem to find the reverse/ class. The Naive approach is used to make the events appear independent of each other, and calculate the probability as a simple multiplication of all the said events. But in case of continuous values, definite probabilities cannot be calculated because the values might all be singular (not occurring again) for all we know. So, we assume the nature of the data to be continuous and represent the attributes with a normal distribution for each of the classes, based on the mean and variance. For testing, the PDF (Probability Density Function) is calculated from the curves and used as the posteriori. In our case, the values of pixels are very well continuous in nature, and hence we have used Gaussian Naive Bayes.The formula for finding posterior probability is given by as below where u is mean and sigma is standard deviation.

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Fig x: Formula for Posterior Probability

The normal distribution or the gaussian distribution follows the rules explained in the diagram where 68,95 and 99 percent of the data lies in the first range i,e from mean-std to mean +std,second range and hence the third range respectively. The probabilities associated with them are also given below.

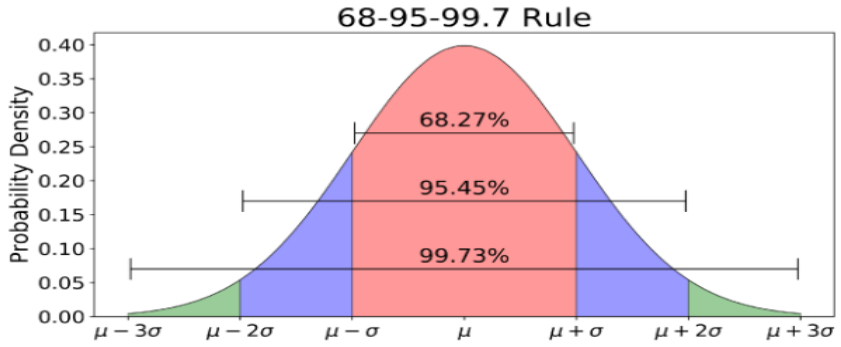


Fig xi: Normal Distribution

2. K Nearest Neighbours (KNN):

K Nearest Neighbours is one of the more simpler algorithms used as it doesn't have any convoluted mathematical equations. The first thing to be kept in mind is that no training is required for this specific algorithm. The testing or classification is done on the basis of the data availability as is. The values to be predicted are done on the basis of 'K' nearest neighbours. The nearest neighbours are calculated by a specific distance metric (Euclidean in our case) for all of the attributes combined. One thing to be remembered is that these neighbours already have their classes. The class with the most occurrences among the nearest neighbours of our data point is deemed as the label or class of that point. In our case, distance is calculated for all of the pixel values (3841). A vital part of this process is selection of the number of 'K'. If K is too small, then our model becomes too specific, and if it is too large then our model becomes too generalized. To find the apt number of K, we just select a range of K, test for all of the values, and select the one that yields the highest accuracy. The formula for the euclidean distance which is used in this algorithm is as given below where p is the testing datapoint and q is an iterable going through all the data points of the training set to find the best neighbours.P1,P2 are the coordinates of P on that axis.

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Fig xii: Euclidean Distance Formula

A KNN representation figuratively is given below where k=3 and the star is the testing point where the three most neighboring points are considered.

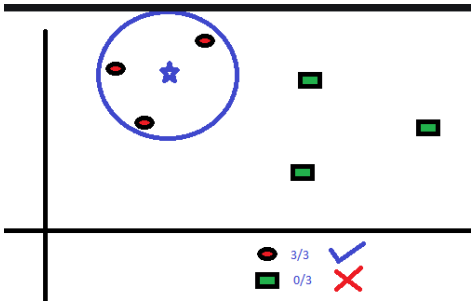


Fig xiii: K Nearest Neighbour approach

3. Decision Tree (DT):

Decision Tree is one of the intermediate algorithms that has a moderate time and space complexity when compared to the other algorithms.Like KNN and Naive Bayes Decision Tree is also a supervised learning algorithm that has a training phase and a testing phase.

Decision Tree works as a very ingenious algorithm where it focuses on the simplest concept of all i.e the impurity in the dataset or the entropy of the input variables. There are no assumptions made here regarding the dataset unlike Naive Bayes. The entropy gain for each attribute is calculated which in layman's term finds out those attributes who have a high diversity and have a close relationship with the output values or to be more simplistic it selects that attribute on which we can make a decision using a simple condition of any kind. The dataset is hence split into two child nodes from the root and this process continues until the child sample is 1 or such that the all the child samples have the same output. This process of traversing through a tree where the data is split into two new mini datasets is the algorithm of the Decision tree. In the training phase once the tree is made, the same tree is used in the testing phase where based on the input values the tree is being traversed until the output is found. The decisions are made hence because of the tree made hence called Decision Tree.A small example is given below where the Head node starts with the feature Person as it has the max entropy gain and then it traverses until fit or unfit.

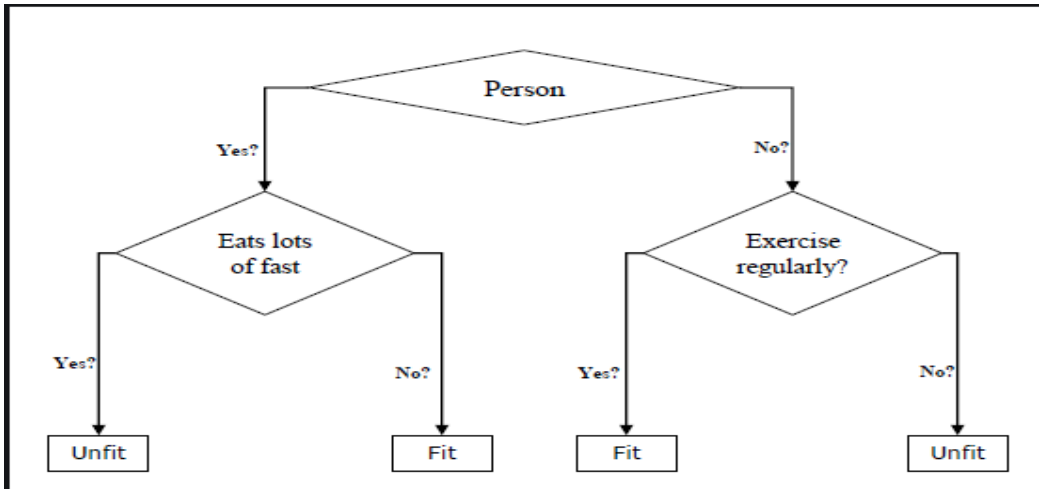


Fig xiv:Decision Tree approach

In this Project we are deciding on the basis of the Pixel values that is the center pixels logically of every image is the decision making pixels as they contain the features and information of the product which will always contrast and have much entropy or impurity which will be closely related to our output Product and hence a high entropy gain. The decisions are made and are started with these center Pixels. Grid Search with Hyperparameter tuning is also carried out to prevent from overfitting and finding those hyperparameters which shall aptly fit our model and hence give us the highest performance.

4. Random Forest (RF):

Random Forest is an advanced algorithm of the simple Decision Tree where the concepts of the are the same as that of the Decision tree with an additional concept of bagging or ensemble learning approach is used. So as the name suggest Random Forest is nothing but a forest and a forest has tree in them hence a random forest is nothing but a collection of Decision Trees.

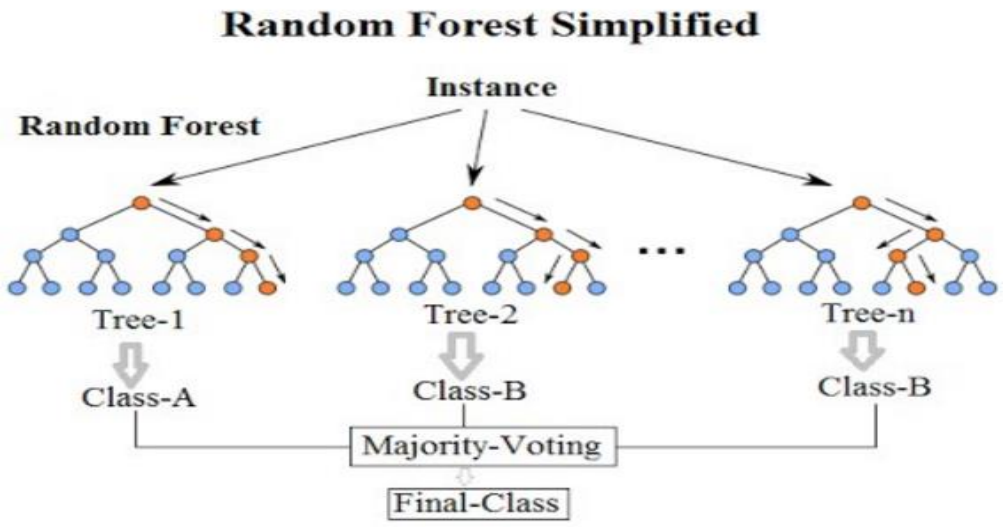


Fig xv: Random Forest approach

This collection of these trees is called bagging or ensemble learning where we have n number of different and unique trees where all of these trees predict an output and the average of these predictions are considered or most frequent ones as the final predictions. The catch is constructing n number of different trees, that is if the dataset is the name then the trees constructed will also be the same that's where the word random comes into play. Random Forest algorithm considers a sub section of the dataset of only some rows and columns of the data each for every tree being constructed and this creation of subsets on which trees are constructed is completely random and hence the name Random Forest. In this Project we have used Random Forest too where grid search is used to create many trees and take the ensemble of the predictions.

5. Voting Classifier (VT):

This algorithm uses the concept of ensemble learning. This is more of a method rather than a machine learning algorithm. The base concept of Voting classifier is the same as that of Random Forest. As the name suggests that there is voting in the classification, here in this approach we bag a few algorithms such as KNN, naive Bayes, random forest etc and after bagging them. We train all of them that is all the algorithms have a model of their own and the voting classifiers model consists of all these models. During the testing phase the testing data on which predictions are made is given to all

the models created of different algorithms and the ensemble or the most frequently predictions is considered as the final prediction. This classification takes the best of all the algorithms and gives us the most frequent value. In this project we have used various combinations of the best fit models of the above four algorithms and have got respective performances. An example is given below we have five classifiers on which the testing datapoint is fed in and all the three give out an output and as you can see the average of all the them is considered as the final output i.e Claas 2 as it's the most frequent or has the highest confidence.

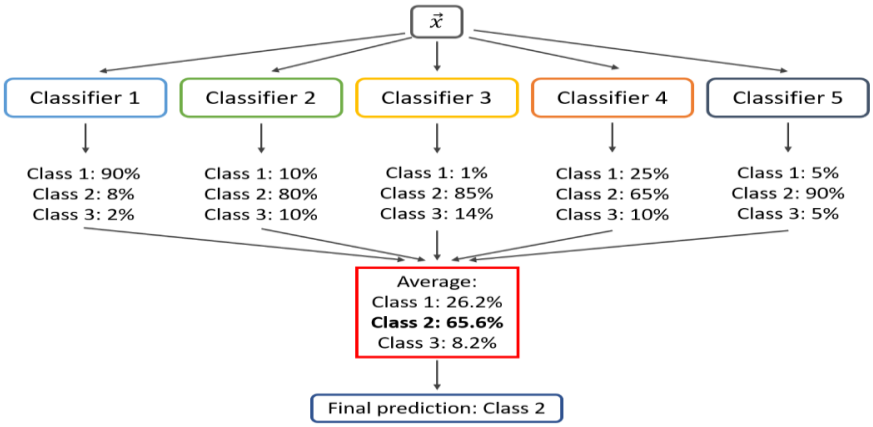


Fig xvi:Voting Classifier approach

3.5 The CNN Flow:

The CNN used consisted of 9 layers in total. As the name suggests, convolution layers were used. The feature learning was done by 6 hidden layers that consisted of convolution and pooling layers.

Convolution layers are formed by convolving the image with various different feature maps. The results lead to formation of one convolution layer. Following this, several layers are formed. Rectified Linear Units (ReLU) are used as activation functions. These units are preferred over the conventional sigmoid functions as they are pretty simple to calculate and take the gradients into account.

The convolution step extracts the main features of the image. The convolutional layer is always the first step in a CNN. We have an input image, a feature detector, and a feature map. We take the filter and apply it pixel block by pixel block to the input image. You do this through the multiplication of

the matrices. Convolution is a linear operation with things like element wise matrix multiplication and addition whereas, the real-world data that our CNN will learn, will be non-linear. Pooling layer is used to reduce the size of the data being transferred over to the next layer. Pooling consist of either max or average pooling. Pooling is nothing but using a mask of a constant size and taking either the average or max of the underlying values. This helps in streamlining the entire flow of data and to avoid overfitting of data. Pooling is used to reduce the parameters taken into consideration by the model. It also helps in taking spatial variance as another feature of the model. The most common example of pooling is max pooling. In max pooling, the input image is partitioned into a set of areas that don't overlap. The outputs of each area are the maximum value in each area. This makes a smaller size with fewer parameters. Max pooling grabs all the maximum value at each spot in the image. This gets rid of 75% of the information that is not the feature. The flatten layer is used to split it into columns.In this process we flatten the pooled feature map into a sequential column of numbers. This allows that information to become the input layer of an artificial neural network for further processing. The dense layer is then used to fully connect all the neurons.

Fully Connected layer depicts or computes which high level features were found in the image. For instance, it would compute the probabilities of all the weights and decide which has the maximum weight and hence come up with the most suitable features to be selected. These then help it predict or classify the object. This combines features and attributes that can predict classes better. The fully connected layer is a traditional Multi-Layer Perceptron. It uses a classifier in the output layer. The classifier is usually a softmax activation function. To use the features from the output of the previous layer to classify the input image based on the training data. The softmax function takes a vector of scores and squashes it to a vector of values between 0 and 1 that add up to 1. Cross entropy often goes hand in hand with softmax. The goal is to minimize the loss function so we can maximize the performance of our network.

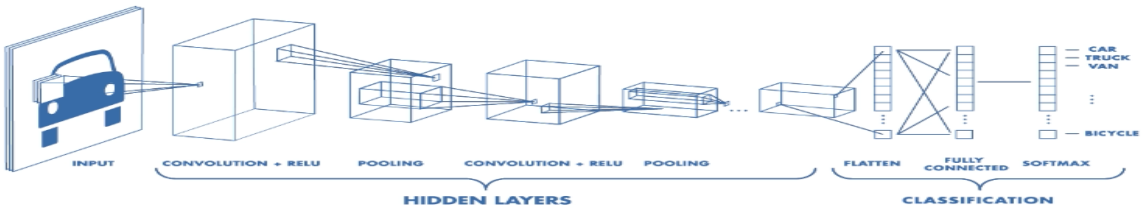


Fig xvii: CNN flow

3.6 Embedded Design Flow:

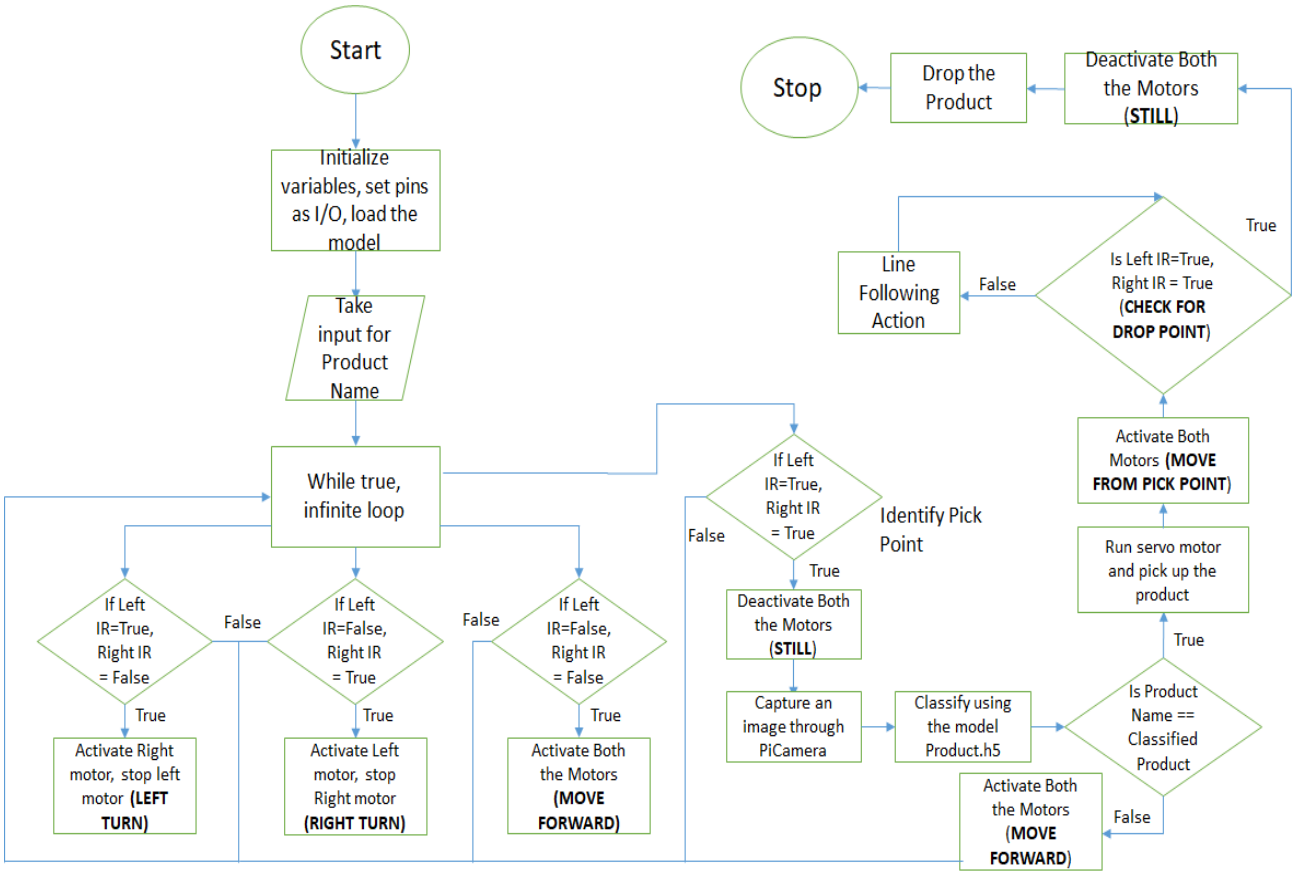


Fig xviii: Flowchart of Project

3.6.1 Capturing Image:

A Pi camera is used in this task for capturing the picture of the product. An image is captured using Picamera, capture and is saved in the test folder of the dataset from where it's further accessed and converted into numeric data. The data is available for the classification to be carried out so the next step is to load the model and use that model to predict the Products name. The prediction function of a CNN model returns a confidence array where for each and every product there is a confidence associated with it .

3.6.2 Line Following Bot:

The primary function of the bot is to detect the presence of a line and follow it. This is achieved by using two sets of IR sensors (each on either front end sides with the line in between), and a primary black line, which is the path we desire the bot to tread along. The IR LED emits IR light which is supposed to bounce off a surface and reflect back to the IR photodetector. In this way, we can test for the presence of an object (a black line in our case). Once light is detected in the reception range, the module outputs a low signal (0 V), but if it fails to detect any light, it outputs a digital high. In our case (line follower bot), the black lines absorb the light and hence, once detected, the module returns a high signal. Whereas for its normal path, it continuously receives the reflected light and outputs a low for most of the part. Under normal circumstances (straight line), neither of the sensors detect the black line, and the motors, as well as the wheels attached to it, keep on running, pushing the bot forward. When the path (black line) takes a turn on either side, the IR sensor (in whose direction the turn has been taken) is alerted, and sends a signal back to the controller. Now this signal can either be a 1 or 0, depending upon the configuration. Once the microcontroller is informed, the set of motors associated with that specific IR come to a stop, while the other one goes on running. This action forces the bot to take a turn until the IR sensor overcomes or avoids the black line, and puts the bot back on track.

3.6.3 Picking or Dropping:

The primary task of the gripper is to grab the objects, prior to or after identification, and safely drop it at the desired location without letting go of it. It consists of two mechanical, customized arms with jagged edges and a gear like structure at its base. The way this works is that one of the arms has the rotating shaft of the servomotor attached to its end which gives it a rotating motion, when the motor is turned ON. The arms are placed so that the movement of one arm, forces the other to move in the opposite direction. This gearing action is caused due to the interlocked teeth of both the gears. A predetermined time and direction, for which the motor is supposed to run, is set based on the objects' width. When the motor rotates anti-clockwise, the arms close in on the object and grab it. Now the

time for which the motor suns decides how far the arms will close in. The gripper has been modelled to grip objects by their neck, instead of anywhere in between. This is because most of the objects we deal with have a circular surface, making it difficult to be gripped. In case of cubical or other shaped objects, we can configure the handling of the gripper accordingly. The arms have to be far apart in order to avoid coming in line of sight of the camera, but after identification they must be able to close in on the object and get a hold of it. In order to avoid the objects from slipping, once they have been grabbed, the arms have been provided with jagged edges to increase friction.

3.7 Components List:

Sr. No.	Component	Quantity
1.	Raspberry Pi Model 3B+	1
2.	Raspberry Pi Camera	1
3.	Chassis	1
4.	DC Motors	4
5.	Servo Motor	1
6.	IR Sensor Module	2
7.	Motor Driver IC	2
8.	12 V Battery	1
9.	Claw	2

1. Raspberry Pi Model 3B+

Fig xixi:Raspberry Pi 3

It is a series of small single-board computers. The technical specs of the above Computer are as follows: All models feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). Equipped with a 1.2 GHz 64-bit quad core processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities. Recently the Raspberry Pi 3 Model B+ was launched with a faster 1.4 GHz processor and a three-times faster gigabit Ethernet or 2.4 / 5 GHz dual-band 802.11ac Wi-Fi (100 Mbit/s). Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MB to 1 GB random-access memory (RAM), with up to 4 GB available on the Pi 4. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards have one to five USB ports.

2. Raspberry Pi Camera:

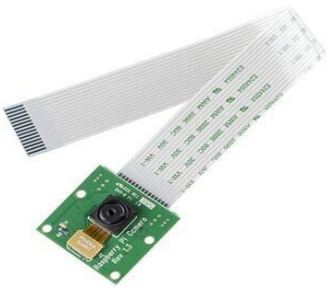


Fig xx:Raspberry Pi Camera

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. It can also be used for time-lapse, slow-motion and other video cleverness.Other libraries also can be used to get the desired result. The module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.The camera consists of a small (25mm by 20mm by 9mm) circuit board, which connects to the Raspberry Pi's Camera Serial Interface (CSI) bus connector via a flexible ribbon cable. The camera's image sensor has a native resolution of five megapixels and has a fixed focus lens. The software for the camera supports full resolution still images up to 2592x1944 and video resolutions of 1080p30, 720p60 and 640x480p60/90.It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library.It also supports live video.

3. Chassis

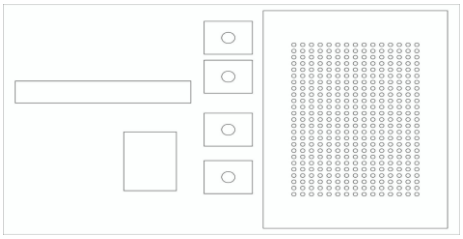


Fig xxi:Design of Chassis

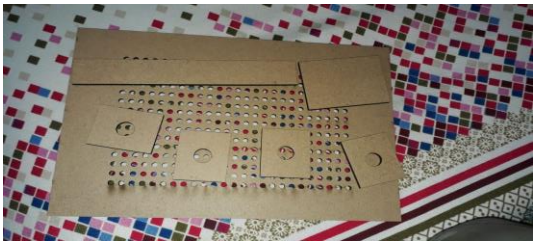


Fig xxii:Chassis

The main frame of the bot is called the chassis. The chassis needs to be durable enough to withstand the weight of all components combined. At the same time, the weight of the chassis itself must not be too high to prevent adding extra, unnecessary weight to the bot. The chassis used was designed in order to be wide enough to encumber all the components without needing to cramp them up. It has a dimension of 30cm x 12 cm x 4cm (the height of the part supporting the wheels). It is made of Medium Density Fibreboard, which is a lightweight and durable material. It is porous, with the pores wide enough for the components to be wired through, in and out.

4. DC Motors

Fig xxiii: DC Motor

A Dc motor contrary to a DC generator is used and functioned to generate a mechanical force or for a mechanical output where electric and magnetic components are provided and placed accordingly so as an electromagnetic force is generated. The Dc Motor used in the project is a motor with a RPM(Rotations per Minute) of 300,an operating Voltage of 12V DC with a gearbox with an attached Plastic and a gearbox Shaft diameter of 6mm with internal hole Torque of about 2 kg-cm and no-load current of 60 mA(Max) and a load current of 300 mA(Max).

5. Servo Motor

Fig xxiv: Servo Motor

A servo motor is an electronic device which can rotate with a great precision unlike Dc motor.A servo motor can be either DC or AC but in this project a DC servo motor is used because of it's high torque and it's small and lightweight package. Servo motor works on principle of pulse width

modulation which is a method where the speed of the motor is controlled by driving it with a series of 1 and 0 or on and off pulses with varying duty cycles. The servo motor used in the project is a x Towerpro MG90S Mini Digital 360 Degree Servo with a Set of Horns of One Point, Two Point and Four Point and x Set of Screws as shown below. The features are as follows:

- 1) Weight: 13 gm
- 2) Operating voltage: 4.8V~ 6.6V
- 3) Stall torque @6.6V : 2.2kg-cm

6. IR Sensor Module

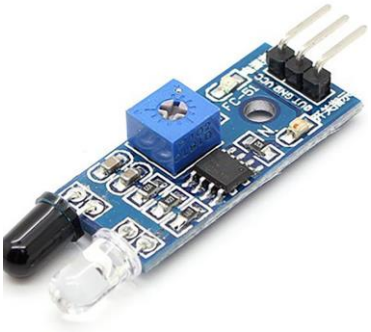


Fig xxv:IR Sensor

The IR Sensor is mainly used to detect or listen to Infrared light, which is not possible for human eyes to see. This property is exploited in various fields like security, wireless communication, remote controls etc. The IR sensor module has a builtin IR Receiver, IR Transmitter and a potentiometer. The various pins attached to the module are VCC, Ground and an Output Pin.

Specifications:

- Operating Voltage: 3.0V – 5.0V
- Detection range: 2cm – 30cm (Adjustable using potentiometer)
- Current Consumption:at 3.3V : ~23 mA,at 5.0V: ~43 mA

- Active output level: Outputs Low logic level when obstacle is detected

Hardware Connections:

- Vcc to 5V
- Gnd to Gnd
- Out to digital pin 7

The IR Transmitter is present in the form of an IR LED which emits IR light, not visible to us humans. IR Receivers are usually in the form of photodiodes or phototransistors. Together, the combination is known as photocoupler. This receiver is really specific as it only accepts IR light. A potentiometer is also present to vary the range of light it is able to detect. This module has a digital OUT i.e. the output is in the form of a high or low pulse. It essentially converts detected light and represents it in the form of a pulse.

7. Motor Driver IC

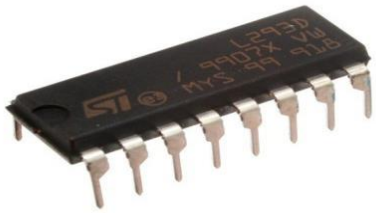


Fig xxvi:Motor Driver IC

As we are aware of the issue of microcontrollers not being able to produce enough voltage or current to drive motors (which may require higher voltage specification), there is a need for intermediate circuitry. That is provided by a Motor Driver. A motor driver (L293D) can be used to run at most 2 DC motors at once, that too, in any direction.

Specifications:

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)

In our project, we have used a DC motor requiring 12V with max load current of 300mA. The microcontroller fails to output such power hence we use L293D to not only raise the required specifications, but also configure two such motors at once and make it easier to operate at one stop.

8. Battery

Fig xxvii : 12V Battery

Specifications:

- F1 Terminals (3/16" wide)
- Rechargeable, recyclable, and no memory effect.
- Sealed, maintenance free with a long service life. Built to last in the roughest situations

9. Arms

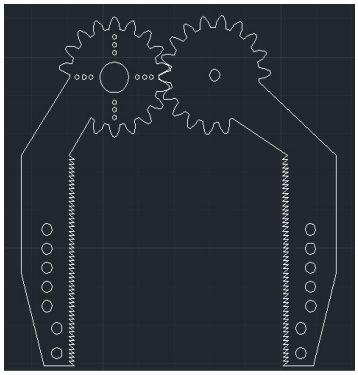


Fig xxviii: Design of Arm

Fig xxix: Arm

The primary task of the gripper is to grab the objects, prior to or after identification, and safely drop it at the desired location without letting go of it. It consists of two mechanical, customised arms with jagged edges and a gear like structure at its base. The arms are placed so that the movement of one arm, forces the other to move in the opposite direction. This gearing action is caused due to the interlocked teeth of both the gears. The arms have to be far apart in order to avoid coming in line of sight of the camera, but after identification they must be able to close in on the object and get a hold of it. In order to avoid slipping of the objects, once they have been grabbed, the arms have been given jagged edges to increase friction.

3.8 Libraries used:

1. TensorFlow:

It is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use.

While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics

processing units).TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

3. Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing Deep Neural Network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows the use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU) principally in conjunction with CUDA.

4. OS:

The OS module in python provides functions for interacting with the operating system. OS, comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact

with the file system.It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

3. Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in an easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits

5 Pandas:

This library is the base of almost all the data analysis, data science, machine learning and deep learning projects. It's responsible for representing the data in a dataFrame an object of this library which has rows and columns where rows indicate all the data points of the given data and the columns indicate all the attributes of the data. This library is very powerful as it can convert csv,json,excel,sql tables to a dataframe. All the Machine Learning algorithms compute on the data and this data is best represented by a dataframe.Additonal functions such as join, groupby , value_counts are also what make it even more widely known because of it's broad range of functions.

6. Numpy:

Numpy is a library for Python Programming. This library is widely used to represent arrays or matrices of huge size of data with multi dimesional features involved in them. This library too is used

widely and will be found in almost any Machine Learning project.This library has a wide range of mathematical functions from finding the sigmoid of the data to finding the maximum element of an array which makes it even more popular.

7. Pathlib:

Pathlib provides various classes representing file system paths with semantics appropriate for different operating systems. This is a standard library used in Python Programming language .Path classes are divided into pure and concrete paths. Pure paths provide only computational operations but do not provide I/O operations, while concrete paths, which are derived from pure paths, provide booth computational as well as I/O operations.

8. OpenCv:

OpenCV is an open-source library used mainly for computer vision, deep learning, machine learning, and image processing applications and projects.. The library supports a wide variety of programming languages like Python, C++, Java,. Tasks such as object recongization, face recognition,video and image processing can be carried out using the functions of this library. Especially when it's integrated with other libraries ,such as Pandas,Numpy,Tensorflow the computations are faster and tasks are easier to accomplish.

9. Sklearn:

Scikit-learn or sklearn is the most widely used library for almost in all of the machine learning applications. This library is built on Numpy, Matplot and Scipy.

Various modules are comprised in this library such as classification, regression, clustering preprocessing, ensemble method, model selection etc. In the Project, we deal with a few specific modules such as Standard Scaler, which comes in processing of the data. StandarScaler is responsible for scaling down the data using Z score. All the algorithms are imported from this library from gaussian naive bayes to knn and decision Tree to the ensemble method algorithms such as random forest and voting classifier.Model Selection is also used in the project using the library Grid Search CV which is responsible for selecting the best model out of the lot.

Chapter 4

Implementation and Experimentation:

This chapter presents a more detailed explanation of hardware. It includes detailed work of hardware design including the circuit diagram of line follower and the codes required to program the controller.

4.1 Machine Learning approach:

The Machine Learning approach approaches the problem statement in a machine learning perspective and gives a machine learning solution to it. Various algorithms are trained and tested to understand their performance and their explanations are given below with the code. The coding is done in Jupyter Notebook and each and the codes are segregated in cells with specific objectives and functions.

1. Prepping the Dataset

We import all the relevant libraries that will be used further, in the first cell. The second cell is responsible for reading all the files in the training folder and validation folder.

In the train folder we have three folders named Product 1,2,3 and those folders have those products images. So to access all these images the second cell of our code is dedicated to this objective.

```
In [4]: testDirect,trainDirect,validationDirect=[p for p in pathlib.Path('C:\\Users\\Trivikram\\.keras\\Products').iterdir()]
        trainFiles=[files for files in trainDirect.iterdir()]
        validationFiles=[files for files in validationDirect.iterdir()]
```

Accessing the dataset of both training and validation

Fig xxx: Accessing datasets

The trainFiles and validationFiles are stored in the declared variables and now our third objective is to read the image and convert it into data that our computer will understand or the algorithms may

compute. To do this we use the functions of the library called opencv to read an image in Red, Green and Blue format and then hence make a dataframe of it where each and every pixel value and their components too are stored in the dataframe hence making our data preparation complete for any algorithm to work.

Hence our trainData and validationData is all ready to be worked on.

```
train_Data
```

	Red Component of PixelNo 0	Green Component of PixelNo 0	Blue Component of PixelNo 0	Red Component of PixelNo 1	Green Component of PixelNo 1	Blue Component of PixelNo 1	Red Component of PixelNo 2	Green Component of PixelNo 2	Blue Component of PixelNo 2	Red Component of PixelNo 3	...	Red Component of PixelNo 1277	G Compo of Pix
0	58	64	68	70	68	66	66	65	68	65	...	151	
1	51	50	50	50	47	49	49	49	50	50	...	114	
2	74	70	67	62	59	63	60	61	62	63	...	136	
3	65	63	68	68	70	70	66	67	68	67	...	130	
4	65	63	64	61	64	66	64	66	64	65	...	131	
5	67	65	66	66	63	66	66	68	70	69	...	129	
6	52	54	57	64	66	65	64	66	64	64	...	133	
7	62	63	64	65	63	64	63	62	65	64	...	133	
8	59	64	64	64	65	66	63	65	70	70	...	128	
9	61	59	58	63	64	64	63	65	65	62	...	130	
10	66	62	66	65	65	67	64	66	69	66	...	131	
11	63	57	62	64	65	69	67	68	65	64	...	131	
12	67	64	62	61	64	67	64	65	67	66	...	129	
13	66	66	63	64	64	65	63	60	66	67	...	134	
14	59	59	59	64	67	68	67	70	68	67	...	136	
15	65	66	67	68	67	65	66	68	69	66	...	134	
16	64	67	67	66	65	68	64	67	67	70	...	128	
17	66	65	70	67	66	68	66	68	68	68	...	132	
18	66	63	68	67	64	68	64	65	63	63	...	131	
19	60	59	64	64	64	66	64	64	67	66	...	130	
20	63	61	67	68	66	63	64	66	68	65	...	128	
21	68	65	64	65	64	63	65	68	70	71	...	127	
22	58	58	59	62	65	64	62	63	65	64	...	134	
23	69	68	64	65	66	64	68	67	66	67	...	135	
24	63	58	61	61	61	62	65	63	63	63	...	131	
25	63	60	61	65	68	65	62	61	65	64	...	132	

Fig xxxi: Dataset

The dataset is made in a numeric format now but all these values of the attributes do not belong to the same range of values which poses a problem in some of the algorithms to be used. So the next cell is used to scale down all the values to make the computation easier and faster and work out effectively and efficiently for some algorithms. The Standard Scaler scales down the entire dataset into a normal

distribution using the Z score formula where x-mean/std is used for each and every attribute. So the scaler is fit and transformed on train data and transformed on validation data. Our dataset is already ready for any algorithm to work on so the next step is to declare two empty lists, which will be used throughout the code, these two lists shall store the algorithm or algorithms used and their respective accuracies.

2. Gaussian Naïve Bayes

The next step is to run the first algorithm called as Naive Bayes. As explained above in any supervised machine learning algorithm there are two phases training and testing where in training all the rules are made looking at the output and the data and in testing these rules are put to the test. Naive Bayes algorithm considers two assumptions the first that all the attributes are independent of each other and the second all of them follow a normal distribution. We are able to achieve an accuracy of about 63 percent from Naive Bayes which is stored in the algorithm list and accuracy list.

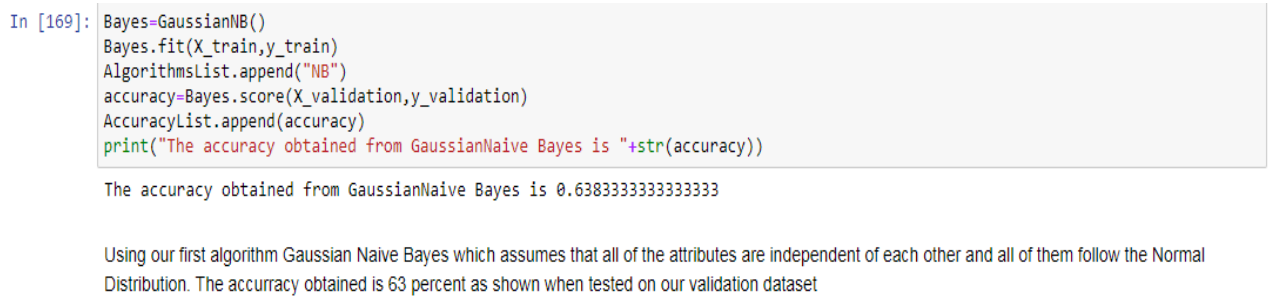


Fig xxxii: Gaussian Function and Accuracy

3. Decision Tree

In the next cell we declare a new variable called decreaseList which stores a list of all the minimum impurity decreases or the minimum entropy gain's value which is to be used in the decision tree. The next step deals with the second algorithm to be used called Decison Tree. Now as explained above how a decision tree works it generally has a tendency to overfit on the training dataset because of it's default values so we use a new concept called Grid SearchCV. This method is used to prevent overfitting where a list of values are provided for the hyperparameters of the Decision Tree such as minimum impurity decrease, min sample split i.e the minimum samples require for a split to occur

else termination of splitting and minimum sample leaf i.e the minimum number of samples required in a leaf so as a split will occur else no split. Employed with Stratified K fold with a value of 5 which divided the entire datasets into k number of folds and trains the algorithm with the hyperparameters given on k folds and k-1 folds are trained on and a model is formed on which the kth fold is tested and a stratified k fold keeps the proportion of the output values in every fold the same as the proportion of the training dataset.

After successfully tuning the algorithm and hence finding the best model out of the lot with the best hyperparameters for the dataset. We calculate the accuracy and print the best estimator to keep it in reference and store accuracy and algorithm in algorithm list and accuracy list as show.

```
In [12]: print("The accuracy obtained from Decision Tree is"+str(grid_search.score(X_validation,y_validation)))
         print("The best estimator is "+str(grid_search.best_estimator_))

         The accuracy obtained from Decision Tree is0.7966666666666666
         The best estimator is DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=6,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=42, splitter='best')

         The accuracy obtained is around 80 percent and the best estimator is as above

In [170]: dt=DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=6,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=42, splitter='best')
          dt.fit(X_train,y_train)
          AlgorithmsList.append("DT")
          AccuracyList.append(dt.score(X_validation,y_validation))

          Storing the best estimator in dt and the algorithm in algo list and also the accuracy in acc list
```

Fig xxxiii: Decision Tree Function and Accuracy

4. Random Forest

A Random Forest is a bag of decision trees who all are fed the testing data and predict independently and the final predictions is the average of all the predictions. We use Grid Search CV with stratified k fold with a value of 5. The hyperparameters we consider in this algorithm are minimum sample split

and minimum impurity decrease just like decision tree and a new hyperparameter called n estimators which is nothing but the number of trees to be considered and constructed by randomly splitting the dataset into subsections on which trees are made. The Grid search is run and the best fit is found.

The accuracy is computed on the validation dataset and the best estimator is printed for future references. The algorithm and the accuracy is stored in the two lists.

```
In [13]: param_grid = {'min_samples_split': range(10,30),'n_estimators': [10,30,50,60,65,70,75,80,90,100,110],'min_impurity_decrease':decr
         rf =RandomForestClassifier(criterion='entropy',random_state=42)
         grid_search = GridSearchCV(estimator = rf, param_grid = param_grid, cv = 5,verbose=10)
         grid_search.fit(X_train,y_train)
```

```
In [14]: print("The accuracy obtained from Random Forest is "+str(grid_search.score(X_validation,y_validation)))
         print("The best estimator is "+str(grid_search.best_estimator_))

         The accuracy obtained from Random Forest is 0.84
         The best estimator is RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=15,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=42, verbose=0,
                                warm_start=False)

         The accuracy obtained from Random Forest is 84 percent and the best estimator is as above
```

Fig xxxiv: Random Forest Function and Accuracy

5. K Nearest Neighbour

The next algorithm which is used is called K Nearest Neighbour which is also called as the lazy algorithm. This algorithm uses the simplest concept of distance formula to compute the K nearest neighbours and takes the mode of the output values of them as the final predictions. Just like the above two algorithms we use grid Search CV with stratified k fold of 5 but unlike the above two there is one significant hyperparameter to be considered called the n_neighbours or the k value of the numbers to be considered for the Final Predictions which is iterated from 2 to 25. The accuracy and the best estimator are found and stored hence in the lists.

```
In [64]: print("The best estimator is "+str(clf.best_estimator_))
         acc=clf.score(X_validation,y_validation)
         print("The validation score obtained is "+str(acc))

         The best estimator is KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                             weights='uniform')
         The validation score obtained is 0.885
```

The best estimator is as above and the score is around 88.5 percent

```
In [172]: knn=KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                                  metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                                  weights='uniform')
          knn.fit(X_train,y_train)
          AlgorithmsList.append("Knn")
          AccuracyList.append(knn.score(X_validation,y_validation))
```

Storing the best estimator in knn and the algorithms and accuracy

Fig xxxv: KNN Function and Accuracy

6. Voting Classifier

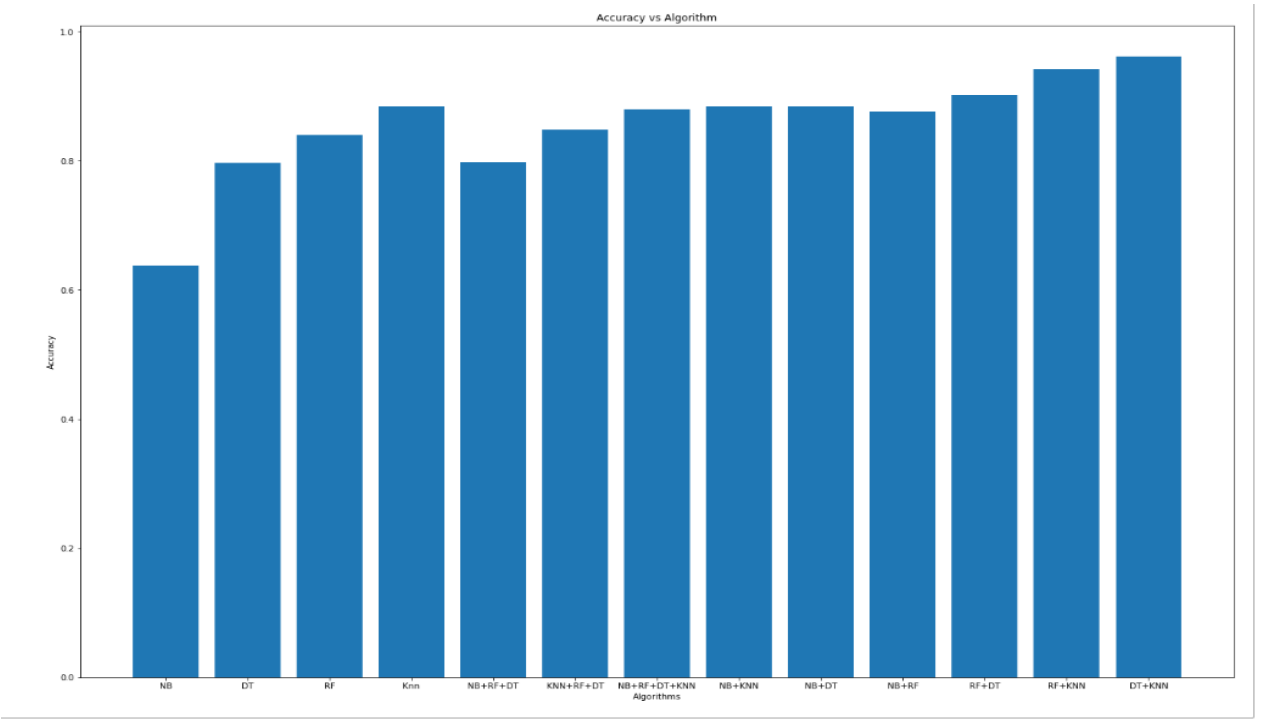
The next algorithm to be used is called the Voting Classifier, which is really nothing but a combination of all the algorithms we have used above. We have considered the best models we could make above and tried out various combinations, just like Random Forest Voting classifier uses ensemble learning with a hard voting giving each and every algorithm's out equal weightage.

Lots of combinations are tried out as below and their accuracies and algorithms used are all stored.

```
In [179]: vt=VotingClassifier(estimators=[('NaiveBayes',Bayes),('Random Forest', rf)], voting='hard')
          vt.fit(X_train,y_train)
          AlgorithmsList.append("NB+RF")
          acc=vt.score(X_validation,y_validation)
          AccuracyList.append(acc)
          print("The Accuracy of Voting Classifier is "+str(acc))

          The Accuracy of Voting Classifier is 0.8766666666666667

In [180]: vt=VotingClassifier(estimators=[('Random Forest', rf),('Decision Tree', dt)], voting='hard')
          vt.fit(X_train,y_train)
          AlgorithmsList.append("RF+DT")
          acc=vt.score(X_validation,y_validation)
          AccuracyList.append(acc)
          print("The Accuracy of Voting Classifier is "+str(acc))

          The Accuracy of Voting Classifier is 0.9016666666666666

In [181]: vt=VotingClassifier(estimators=[('Random Forest', rf),('KNN',knn)], voting='hard')
          vt.fit(X_train,y_train)
          AlgorithmsList.append("RF+KNN")
          acc=vt.score(X_validation,y_validation)
          AccuracyList.append(acc)
          print("The Accuracy of Voting Classifier is "+str(acc))

          The Accuracy of Voting Classifier is 0.9416666666666667

In [182]: vt=VotingClassifier(estimators=[('Decision Tree', dt),('KNN',knn)], voting='hard')
          vt.fit(X_train,y_train)
          AlgorithmsList.append("DT+KNN")
          acc=vt.score(X_validation,y_validation)
          AccuracyList.append(acc)
          print("The Accuracy of Voting Classifier is "+str(acc))

          The Accuracy of Voting Classifier is 0.9616666666666667
```

Fig xxxvi: Voting Classifier and Accuracy

We draw a bar graph of accuracies vs algorithms to represent the performance of each algorithm:



A bar plot depicting accuracies with all the algorithms and combination of algorithms tried out.

Fig xxxvii: Bar plot of accuracy vs algorithms

Conclusion: So after successfully trying out algorithms and all the combinations we have achieved with the highest accuracy of 95 percent with a vt of Decision Tree and KNN ,this is still considerably good but it's not enough to effortlessly categorize products which is why we move on to Deep Learning.

4.2 Deep Learning Approach

Hence, we continue with the CNN or the deep learning algorithm, which yields a higher accuracy while requiring less computational time.

1. Initialization:

The os package is used to read files and directory structure.NumPy is used to convert python list to numpy array and to perform required matrix operations Matplotlib.pyplot to plot the graph and display images in the training and validation data.

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

```
import tensorflow as tf
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import numpy as np
import matplotlib.pyplot as plt
```

2. Loading the Directory:

Start by assigning the directory path to a variable to be used further. After extracting its contents, assign variables with the proper file path for the training and validation set.

```
[ ]  PATH = os.path.join("C:\\Users\\Trivikram\\.keras", 'Products')
```

After extracting its contents, assign variables with the proper file path for the training and validation set.

```
[ ]  train_dir = os.path.join(PATH, 'train')
     validation_dir = os.path.join(PATH, 'validation')
     test_dir=os.path.join(PATH,'test')
```

```
[ ]  train_product_1_dir = os.path.join(train_dir, 'Product1')  # directory with our training Product1 pictures
     train_product_2_dir = os.path.join(train_dir, 'Product2')  # directory with our training Product1 pictures
     train_fanta_dir = os.path.join(train_dir, 'Product3')  # directory with our training Product1 pictures

     validation_product_1_dir = os.path.join(validation_dir, 'Product1')  # directory with our validation Product1 pictures
     validation_product_2_dir = os.path.join(validation_dir, 'Product2')  # directory with our validation Product2 pictures
     validation_fanta_dir = os.path.join(validation_dir, 'Product3')  # directory with our validation Product3 pictures
```

3. Understanding the Dataset:

Let's look at how many product images are in the training and validation directory:

For convenience, set up variables to use while pre-processing the dataset and training the network.

```
num_product_1_tr = len(os.listdir(train_product_1_dir))
num_product_2_tr = len(os.listdir(train_product_2_dir))
num_product_3=len(os.listdir(train_fanta_dir))

num_product_1_val = len(os.listdir(validation_product_1_dir))
num_product_2_val = len(os.listdir(validation_product_2_dir))
num_fanta_val = len(os.listdir(validation_fanta_dir))

total_train = num_product_1_tr + num_product_2_tr+num_product_3
total_val = num_product_1_val + num_product_2_val+num_fanta_val
```

```
[ ]  batch_size = 128
     epochs = 10
     IMG_HEIGHT = 150
     IMG_WIDTH = 150
```

4. Data preparation:

Format the images into appropriately pre-processed floating point tensors before feeding to the network:

1. Read images from the disk.
2. Convert them into floating point tensors.
3. Rescale the tensors from values between 0 and 255 to values between 0 and 1, as neural networks prefer to deal with small input values.

All these tasks are done with the ImageDataGenerator class provided by tf.keras. It can read images from disk and preprocess them into proper tensors. It will also set up generators that convert these images into batches of tensors helpful when training the network.

```
[ ]  image_gen_train = ImageDataGenerator(
                        rescale=1./255,
                        rotation_range=45,
                        width_shift_range=.15,
                        height_shift_range=.15,
                        horizontal_flip=True,
                        zoom_range=0.5
                        )
     validation_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our validation data
     test_image_generator = ImageDataGenerator(rescale=1./255)
```

After defining the generators for training and validation images, the flow_from_directory method load images from the disk, applies rescaling, and resizes the images into the required dimensions.

```
train_data_gen = train_image_generator.flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(IMG_HEIGHT,IMG_WIDTH),
                                                           class_mode="categorical")
```

```
Found 384 images belonging to 4 classes.
```

```
val_data_gen = validation_image_generator.flow_from_directory(batch_size=batch_size,
                                                              directory=validation_dir,
                                                              target_size=(IMG_HEIGHT,IMG_WIDTH),
                                                              class_mode="categorical")
```

```
Found 204 images belonging to 4 classes.
```

5. Creating the model:

The model consists of three convolution blocks with a max pool layer in each of them. There's a fully connected layer with 512 units on top of it, that is activated by a relu activation function. The model outputs class probabilities based on binary classification by the sigmoid activation function.

```
[ ]  model_new= Sequential([
        Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
        MaxPooling2D(),
        Conv2D(32, 3, padding='same', activation='relu'),
        MaxPooling2D(),
        Conv2D(64, 3, padding='same', activation='relu'),
        MaxPooling2D(),
        Flatten(),
        Dense(512, activation='relu'),
        Dense(3, activation='softmax')
    ])
```

6. Compiling the mode:

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

7. Viewing all the layers of the network:

```
:  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 20736)             0
_________________________________________________________________
dense (Dense)                (None, 512)               10617344
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 2052
=================================================================
Total params: 10,642,980
Trainable params: 10,642,980
Non-trainable params: 0
```

8. Train the model

Use the fit_generator method of the ImageDataGenerator class to train the network.

```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)
```

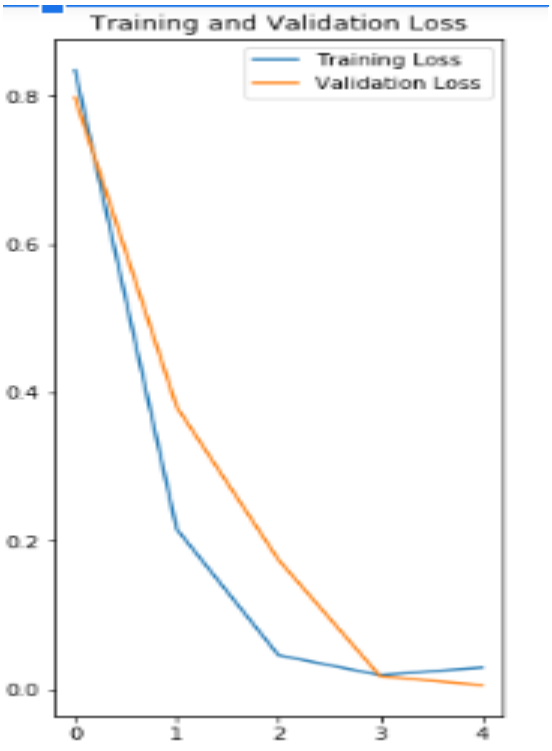
9. Training Result:

```
Epoch 1/5
75/76 [============================>.] - ETA: 0s - loss: 1.4092 - categorical_accuracy: 0.4866Epoch 1/5
76/76 [==============================] - 55s 719ms/step - loss: 1.3974 - categorical_accuracy: 0.4908 - val_loss: 0.4930 - val_
categorical_accuracy: 0.8450
Epoch 2/5
75/76 [============================>.] - ETA: 0s - loss: 0.1436 - categorical_accuracy: 0.9572Epoch 1/5
76/76 [==============================] - 48s 629ms/step - loss: 0.1417 - categorical_accuracy: 0.9578 - val_loss: 0.1692 - val_
categorical_accuracy: 0.9650
Epoch 3/5
75/76 [============================>.] - ETA: 0s - loss: 0.0159 - categorical_accuracy: 0.9973Epoch 1/5
76/76 [==============================] - 64s 845ms/step - loss: 0.0157 - categorical_accuracy: 0.9974 - val_loss: 0.1600 - val_
categorical_accuracy: 0.9550
Epoch 4/5
75/76 [============================>.] - ETA: 0s - loss: 0.0032 - categorical_accuracy: 1.0000Epoch 1/5
76/76 [==============================] - 55s 725ms/step - loss: 0.0032 - categorical_accuracy: 1.0000 - val_loss: 0.2850 - val_
categorical_accuracy: 0.9000
Epoch 5/5
75/76 [============================>.] - ETA: 0s - loss: 0.0018 - categorical_accuracy: 1.0000Epoch 1/5
76/76 [==============================] - 41s 542ms/step - loss: 0.0018 - categorical_accuracy: 1.0000 - val_loss: 0.1817 - val_
categorical_accuracy: 0.9550
```

10. Plotting the training parameters:

```
acc = history.history['categorical_accuracy']
val_acc = history.history['val_categorical_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Output Plot:



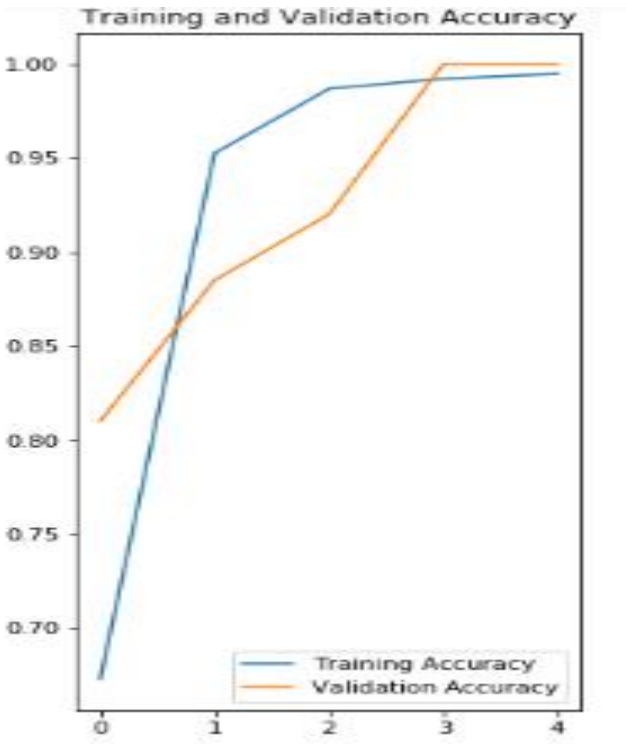


Fig xxxviii: Training and Validation loss vs iterations

Fig xxxix: Training and Validation accuracy vs iterations

11. Saving the model:

```
model.save("classifer.h5")
```

4.3 Hardware Implementation

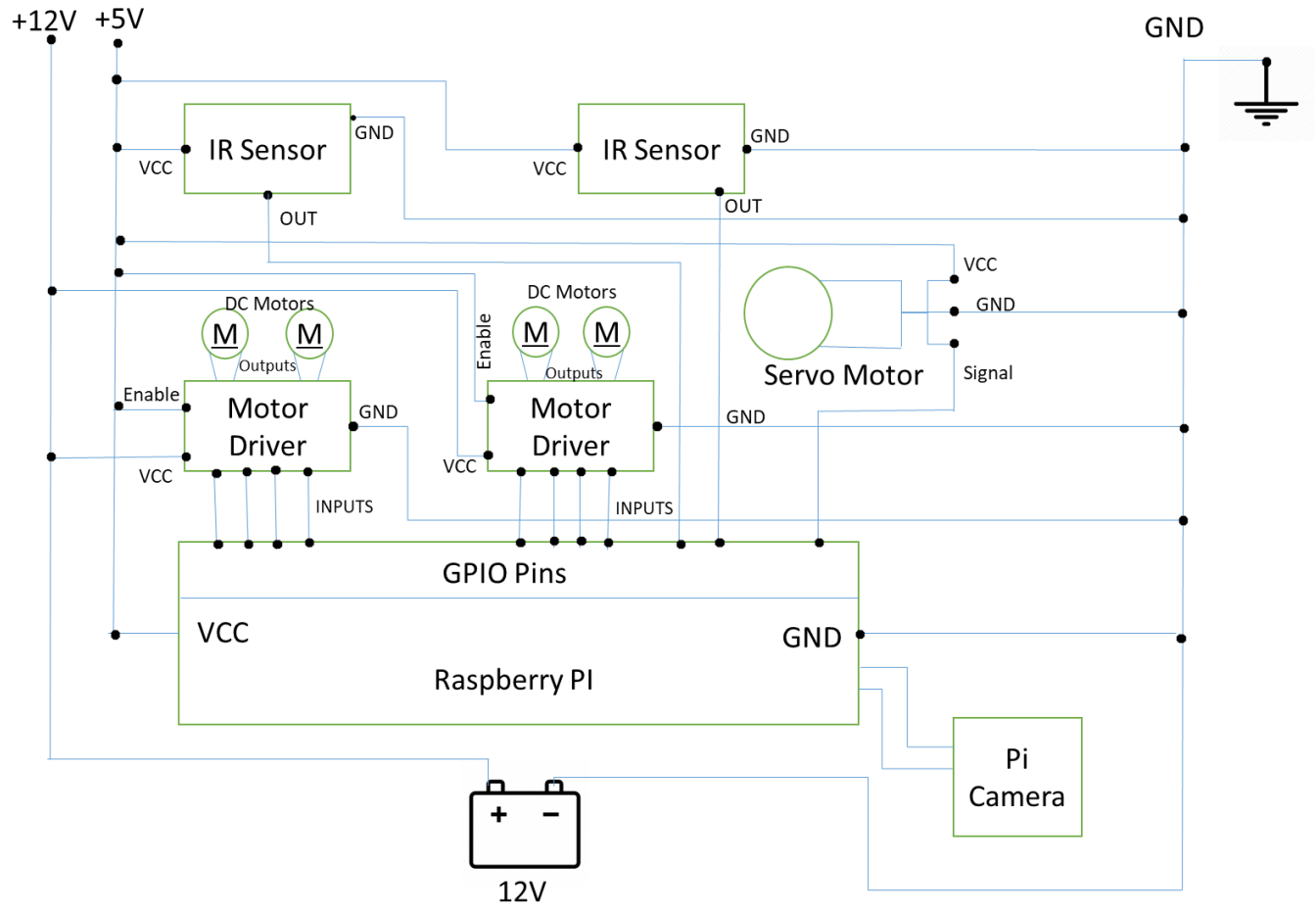


Fig xl: Circuit Diagram

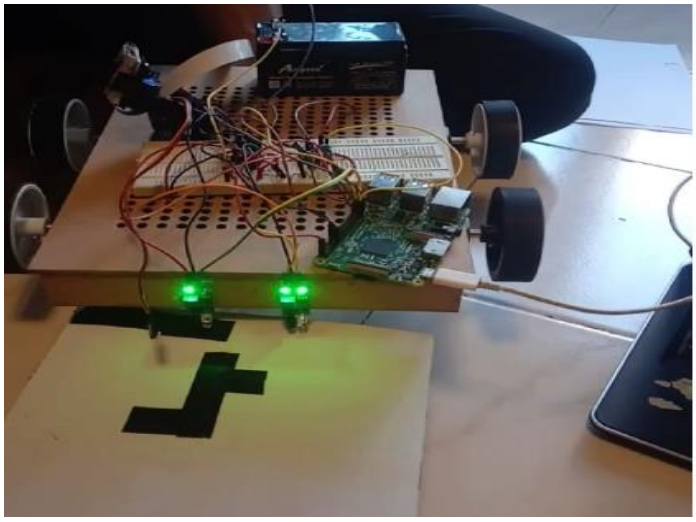


Fig xli: Robotic Car

The Project has a robotic car built on a single board computer Raspberry Pi with a claw, camera,four Dc Motors, one servo motor and two IR Sensors and their circuit diagram is shown as above.

The Pi Camera, which is used to capture an image of the product, has a connector strip, which is specifically made only for a Raspberry Pi 3.The connector strip is inserted in the sort of Raspberry Pi 3 in the standard way where the blue film of the strip is in the direction of the Ethernet port.

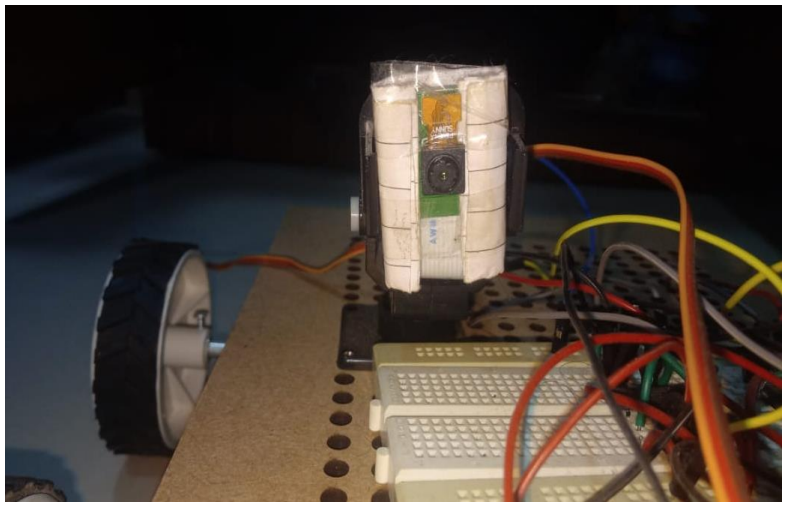


Fig xlii: Connection of Pi Camera to Raspberry Pi

A Servo Motor is also connected to the Raspi as shown above, this motor is used attached to the claw through a screw. A servo Motor has three terminals called VCC, Ground and a signal pin. The VCC and Ground are connected to those of the Pis. The signal pin is connected to an I/O pin, which is programmed in a PWM mode and output mode for RaspberryPi. This Pin is responsible for the sending of the signal and hence with programming we can control the movement of the claws for gripping and release.

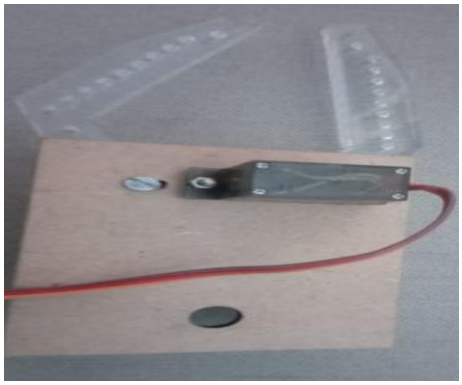


Fig xliii: Connection of Servo Motor to Arm

Two IR Sensors are also connected to the Raspberry Pi, which plays an important role in keeping the robotic car on the right track. The sensor has three terminals VCC, Gd and a signal pin. The Vcc and gd are connected to that of Raspberry Pi. The signal pin is responsible for sending a binary signal i.e either 1 or 0 depending on the setting set on the IR Sensor and the color of the surface below the IR sensor. The sensor returns a 1 if there is black spot below it and hence if it detects it using IR rays. So this signal terminal is connected to a GPIO pin which acts as an input pin with respect to the Raspberry Pi and this concept is the base for the line follower system.

Fig xliv: Connection of IR Sensors to Raspberry Pi

There are four DC motors whose ending each are inserted in the wheels of the robotic car and hence these motors have the ability to control the movement of the car. These motors have two terminals positive and negative terminal who if got high and low shall move clockwise and anticlockwise for the contrary and ideal for both as low.T hese four motors need a higher voltage than what Raspberry Pi is capable of i.e 5V which is why a DC Battery of 12V is used. The battery can't be directly connected to the motors as the speed will be uncontrollable which is why we use an Integrated Circuit called L293D or more commonly known as the motor driver IC. This IC acts as an interface between the motors and the battery and Raspberry Pi. A single motor IC can control two motors because of which two motors driver ICs are used. The IC has 8 pins out of which 2 are input,2 output 2 grounds ,1 vcc and 1 enable. The two inputs pin are connected to the I/O pins of the Pi set as output for the Pi from which signals are sent from Pi to control the movement of the car.

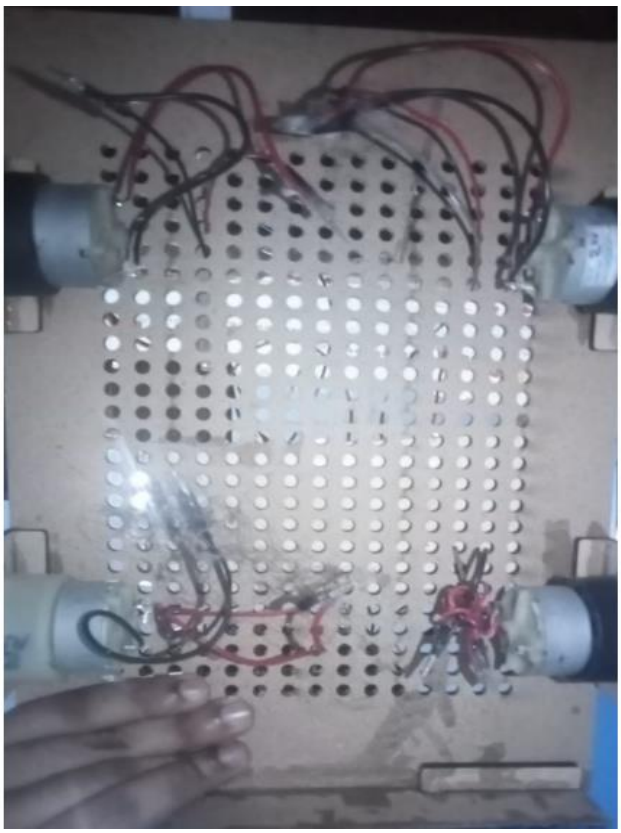


Fig xlv: Connection of DC Motors to Circuit

Fig xlvi: Attachment of DC Motors to Wheels

The two output pins are connected to the motor's positive and negative terminals(shorted).The Vcc and group is connected to that of PI's and the enable pin is capable of starting the IC which by default is active high which is why it's connected to VCC.

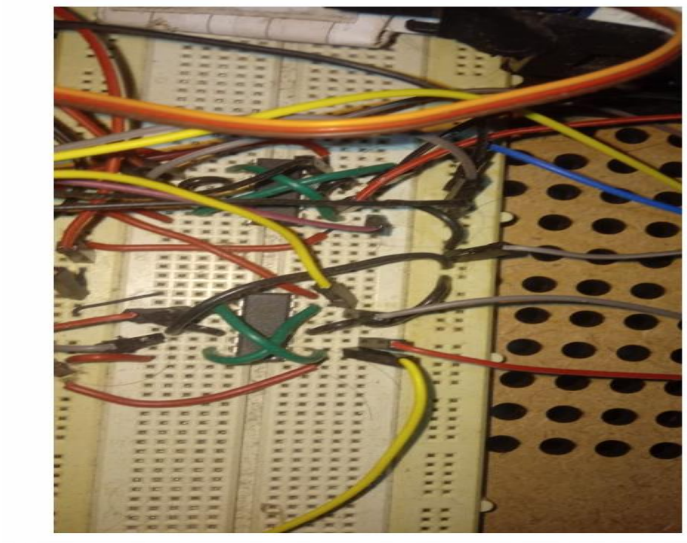


Fig xlvii: Connection of Motor Driver IC to Motors

The embedded implementation has three major parts associated with it that's explained below. The embedded implementation is carried out on RASPBERRY PI 3, which can be considered as a single board computer that works on the LINUX operating system. The board has many features and a terrific processing speed which makes it suitable for advanced applications. PI board is specifically designed for IOT (Internet of Things) applications. We are using it as a controller that does three important tasks.

4.4 Embedded Implementation:

4.4.1 Classifying Products

The first task is to classify the products. This task is carried out by using the model made on our laptop using Deep Learning which is explained above. The same model is saved in H5 format and copied

into the single board computer called Pi where we use this model to classify our products. The model called Product.h5 as shown below is responsible for the process of classification.

Before putting the camera to use, we test it out by taking a demo image and verifying the same. An image is captured using Picamera.capture and is saved in the test folder of the dataset from where it's further accessed and converted into numeric data using image_generator function. The data is available for the classification to be carried out so the next step is to load the Product.h5 model and use that model to predict the Products name. The prediction is in the form of an array with confidence percentage for each class stored in the model (an array with 3 values, each representing the confidence of the product to be classified as that respective class). A dictionary is used to map the indices of these arrays to the product name. The product with the highest confidence percentage is deemed to be the classified product, only if it crosses a threshold of 98%.

If the highest confidence is less than 98 percent then the model isn't successful in classifying the product could be because there is a new Product or because there is nothing there so we print No Product.

This is the task of classification and given below is the image that was captured and the output of the code.

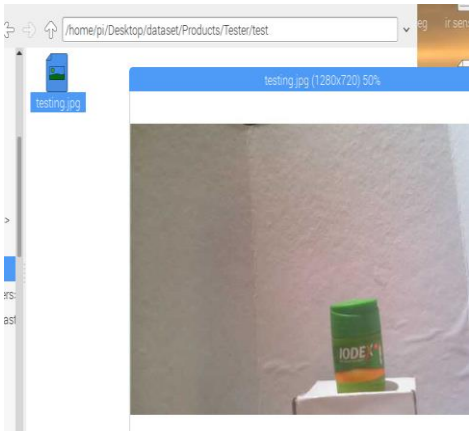


Fig xlviii: Image Captured by PiCamera

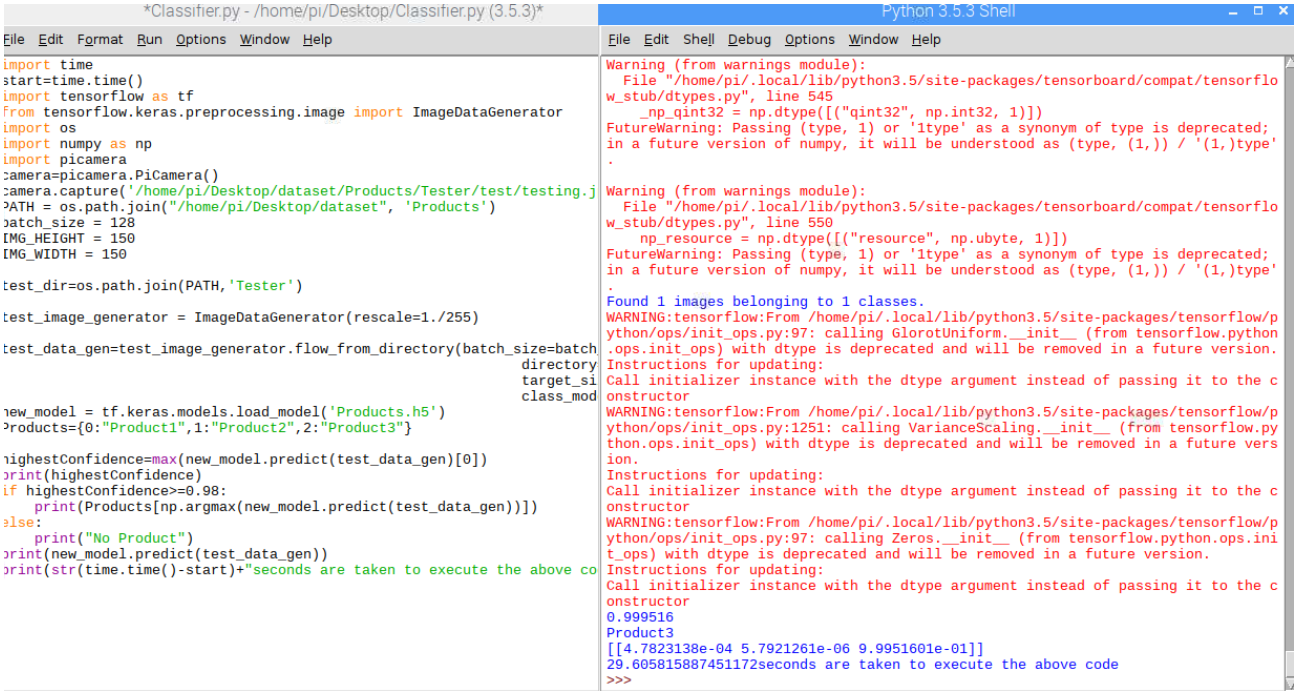


Fig xlix: Output Showing Classification of Product

As shown above the Product 3 is predicted and the testing image and the Product 3 is shown above to validate our classification. The confidence is also printed as 99 and the total time and the confidence array is printed too.

4.4.2 Line Following Mechanism

The second important task to our Project is the concept of the line follower system. The idea explored here is that in every store there are pathways between shelves which is where we intend to tape a black line which is the pathway for our robotic car to move from one pathway to another and hence from one Product to another. To detect a line and stay on the line two IR Sensors are used to detect the black line and stay on them. And to move around, four wheels which are each attached to a DC motor are used to move around.

```
IO.setwarnings(False)
IO.setmode(IO.BCM)
IO.setup(21,IO.IN)#GPIO 21 For stopping and picking objects
IO.setup(2,IO.IN) #GPIO 2 -> Left IR out
IO.setup(3,IO.IN) #GPIO 3 -> Right IR out
IO.setup(26,IO.OUT) #GPIO 4 -> Motor Left terminal A
IO.setup(20,IO.OUT) #GPIO 14 -> Motor Left terminal B
```

```
IO.setup(17,IO.OUT) #GPIO 17 -> Motor Right terminal A
IO.setup(18,IO.OUT) #GPIO 18 -> Motor Right terminal B
```

Before beginning, we run a quick test run to check up on the IR sensors and motors. We check whether the IR sensor outputs a 1 or True after detecting a black spot. Similarly, the motors are checked by seeing whether they run after being powered up. The two left motors are internally shorted and hence they both shall run if the their input pin is powered up, similarly is for the two right motors. Pins 4 and 14 are the positive and negative terminal of the two right motors, similarly 2 and 3 are for left. The first mini code where the robotic car is expected to move forward is when both 4 and 2 are high and the rest two are low meaning all the motors are active and ready to run in the forward direction. There is a 5 second delay i.e for five second the car shall move forward and then it shall reverse its direction as the negative terminals are high and positive are low leading to a reverse drive.

The next step is to write the logic of a line following bot. The first line indicates an infinite loop i.e the code shall keep on running inside this loop until there is a break statement or an interrupt.

```
while i<3:

    if(IO.input(2)==False and IO.input(3)==False): #both white move forward

        IO.output(26,True) #1A+
        IO.output(20,False) #1B-
        IO.output(17,True) #2A+
        IO.output(18,False) #2B-

    elif(IO.input(2)==True and IO.input(3)==False): #turn right

        IO.output(26,True)
        IO.output(20,False)
        IO.output(17,False)
        IO.output(18,False)

    elif(IO.input(2)==False and IO.input(3)==True): #turn left

        IO.output(26,False)
        IO.output(20,False)
        IO.output(17,True)
        IO.output(18,False)
```

The conditions are based on turns that may appear in the path. In an ideal state, the black line remains in between the IR sensors, and both the sensors get their reflected light back, returning logical Low.

In such a condition, the bot is programmed to keep running, that is, if both the sensors return 0, then the respective side motors will be turned on. But, if one of the sensors return a high or 1, it is understood that the path has turned in the direction of the motor. In such a case, we need the bot to make a turn in the said direction, and to do so, we turn OFF the motors on that side, while other ones keep running. This action forces the bot to take a turn. This has been encoded in the first 2 case loops that is, if left IR returns a 1, we deactivate the left motors and keep the right ones running.

The last condition is when both the IR sensors detect a black spot that's when the car stops moving and hence all the motors are rendered inactive. That's all there is to the second important task to our Project.

Fig l: Line Following Mechanism

4.4.3 Picking the Product

The classification of Products and the line following system is carried out flawlessly, the third and the last task of our Project is once the Product is classified picking of the Product using claws custom designed and made which are attached to a servo motor. This motor through its precision movement is used in clawing or picking up the Product by its neck and then releasing it later on,

The concept of PWM is used in picking up and releasing of objects. The pin No 11 is used as a PWM pin where the frequency of 50 HZ is coded.

With the changes in duty cycle the Product is first clawed with a change of 2.5 width in the cycle because of which the motor rotates by 90 degree and hence picks up the Product. A 5 second delay is provided in which the Product is clawed by our claw. After the 5 seconds, the claws are rotated by 180 degrees with a chance of 12.5 in the duty cycle and this leads to the release of the Product.

Fig li: Picking the Product

Chapter 5

Conclusion and Scope for Further Work

This chapter presents the Conclusion drawn with regard to all the work done. The purpose of implementing this idea is briefly discussed along with its advantages. The future work is mentioned, which includes all the relevant developments in enhancing various features.

5.1 Conclusion:

Starting with the inefficiencies of barcode system, we went on to discover how much of an advantage an automated classifying ML algorithm would have. After testing for various algorithms, we found CNN architecture, a Deep Learning Algorithm, to be the best-suited one for our dataset. The way it learns to distinguish between classes based on the results of countless feature maps, gives it a huge advantage over other algorithms. By doing so, it can map an n number of features, without any human intervention, and classify objects quite successfully. Thus, after employing this technique, the classification used in this project has become quite precise. The reason being a high average confidence in most of the testing set. Therefore, we can conclude that we were successful in distinguishing between products using CNN.

As for the automated bot moving on its own, carrying products from one location to the other, is an unforeseen yet innovative technique being used here. Having such a system installed in stores can eliminate the need of humans at the workplace. Picking up items from the shelves and dropping it at the counter or vice versa, extends our simple classifying algorithm to many more potential opportunities, in order to reach complete automation of the tasks.

5.2 Incomplete Tasks:

Due to some unforeseen circumstances, we were unable to see our project come to fruition. A few of the tasks remain unaccomplished, like:

1. Defining a proper path for the bot to follow, including the pickup and drop locations.

2. Deciding a proper mechanism to inculcate both, pickup and drop features.

3. Integrate all the embedded logic and circuitry (even though achieved in parts).

4. Complete, real time testing of the entire bot, and project itself.

5.3 Scope for Future Work:

There is a dire need for an extensive dataset, spanning data of hundreds of products. The more data used, the better the model is trained and hence, an even more robust system is obtained. Identification of several products at once, with the help of object isolation, might be plausible in the near future. As an extension, we might want to include other functionalities like billing of the products, learning customer behavior from previous order etc.

There is a huge scope of improvement in terms of the bot. The bot can see improvement in terms of its movement, where it can be capable of several other things like avoiding obstacles, being capable of lifting the object vertically while carrying. Another important feature might be the linking of product and its shelf location, for the bot to know its predestined location. A longshot feature can be use of a server to query all the requests in the hopes of being implemented by the bot. Overall, an app that encompasses all of the software additions, mentioned above, can be used. It can perform all the tasks like catering the products available in the store, acting as a shopping list for the customers, managing the store etc. A database, to keep a track of the products available, can be linked with this app, and the entirety of our system can be handled by the app, too.

Bibliography:

1) Sameer Khan and Suet-Peng Yong "A Deep Learning Architecture for Classifying Medical Image of Anatomy Object", Annual Summit and Conference, ISBN 978-1-5386-1543-0, pp. 1661-1668, 2017

2) Rui Wang, Wei Li, Runnan Qin and JinZhong Wu "Blur Image Classification based on Deep Learning", IEEE, ISBN 978-1-5386-1621-5 pp. 1-6, 2017

3) Teny Handhayani, Janson Hendryli, Lely Hiryantyo "Comparison of Shallow and Deep Learning Models for Classification of Lasem Batik Patterns", ICICoS, ISBN 978-1-5386-0904-0, pp. 11-16, 2017

4) Classifying Cats and Dogs by sentdex

5) Tensorflow, Sk Learn, OpenCV, Pi Camera documenation

6) Edge electronics

7) Wikipedia

8) Geeks for Geeks

9) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition book by Aurélien Géron

10) Deep Learning and Machine Learning course by Andrew NG Coursera

Acknowledgement:

First of all, we would like to thank our Co-Guide, Prof. Akshata Prabuhu, for her constant support throughout the entire project. Her guidance, encouragement and monitoring is what made the completion of the project possible.

We would also like to thank our guide, Prof. Nitin Nagori for his care and support.

Last but not the least, a huge thanks to the entire team for being motivated and driving enough this project to the current stage.