ABSTRACT

Using supervised machine learning approach, A Convolution Neural Network is trained on images capture from a monocular front-facing camera, with human steering commands. This approach is astonishingly proved to be powerful. With least amount of training data from humans, the system learns to drive autonomously in existence of other cars and road signs on a self-made tracks comprising of solid and broken lane markings. It also operates on unseen tracks in different lighting conditions. The system consequently learns necessary internal processing such as detecting useful road features with only input of human steering angle as the training label. Model is never explicitly trained it to detect boundaries of roads. We have used python for building the architecture convolution Neural Network and a microcontroller for actuation of commands in real-time.

NVIDIA CUDA Toolkit with gtx 1080 is used which provides a development environment for creating high performance GPU-accelerated machine learning application. At 15 frames per second (FPS) the system operators conscientiously.

Table of Contents

Table of Figures

Chapter#1
Introduction

Introduction

Road safety is one of the most important issues which we face today. Due to our hectic lives, mobility plays quite an important role, as documented that majority of the deaths occur due to traffic accidents. The main idea behind our project is to offer an adequate solution to road safety. The Self-driving car is the solution to road safety. The Self-driving car provides safety and can be used by any type of user as the system does not need any previous knowledge for the use. The Self-driving car will be able to detect its lane and avoid collisions. The Self-driving car senses the painted lines on the road and detects the stop signal and traffic lights.

1.2 Scope

The Self-driving cars are driverless cars that make the decision itself and uses the image processing technique. For proper driving the concept of lane detection is introduced. Lane detection is not possible if lane-markings are not clearly visible on the road. This project only focuses on a single road sign i.e. stop sign. The Self-driving car also detects the presence of other cars in the environment and avoids collision.

1.3 Objectives

The principle target of this venture is to accomplish following assignments:

- Self-driving on the track
- Stop sign recognition
- Front collision avoidance

1.4 Description

The project is about Self-Driving cars or in other words to automate cars. The Self-driving cars are cost effective and provide better response time. A safer driver that is always alert and never gets distracted, in this way as there is less human intervention there are low chances of errors.

This involves supervised learning where the model learns from labelled training data set. The data set includes eight thousand images of the track along with the labelled directions i.e. left, right, forward. Once the model is trained it predicts steering direction from incoming stream of frames in real time using the algorithm Convolutional Neural Networks.

1.5 Definitions, Acronyms and Abbreviations

Different Acronyms and Abbreviations are mentioned in the Table 1.1

Term	**Definitions**
Cascade Classifier	Cascade classifier is a powerful object detection framework based on the concatenation of several Classifiers
Face Recognizer	Class for face recognition
OpenCv	Open Source Computer Vision
CNN	ConvNets or Convolutional Neural Network (CNN) is a supervised machine learning algorithm which belongs to a family of deep, feed-forward Neural Networks that has been proved very effective to analyzing visual imagery
Raspbian	Raspbian is Linux based operating system officially supported by Raspberry Pi hardware
Confusion Matrix	A confusion matrix is a contingency table of correct and incorrect classifications frequently used to define classification algorithm’s performance on a test data set for which the true values are known
SVM	SVM is a binary classifier formally defined by a separating hyper plane of two class by the largest margin and outputs an optimal hyper plane which categorizes new examples

Table 1. 1 Definitions, Acronyms and Abbreviations

1.6 Document Conventions

These conventions are used for the information of the developing software which a user is going to develop.

1.6.1 Formatting conventions

Different Conventions and its description are mentioned in the Table 1.2

Name	Conventions
Font	Times New Roman 12 point
Heading 1	Bold 14 points
Heading 2	Bold 13 points
Heading 3	Bold 12 points
Sections	Number 1-5
Subsections	Section no, Subsection no

Table 1. 2 Formatting Conventions

1.6.2 Naming Conventions

- Explanations of terms are explained in "()" in front of the term
- All the abbreviations are composed in Upper case letters

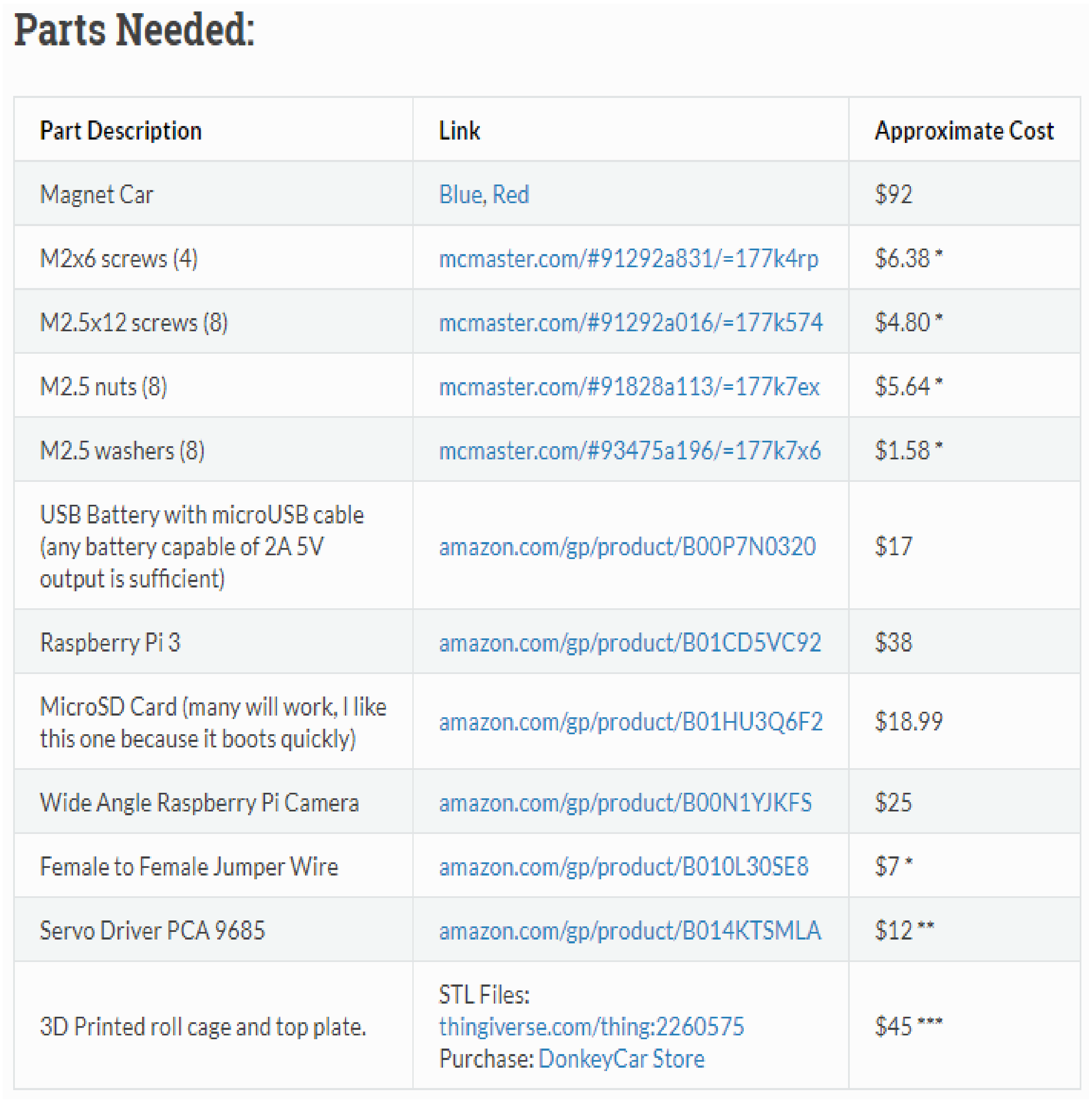
References are numbered as "$_{[1]}$" in subscript

Chapter #2

Project Components

2 Project Component

This chapter presents the detail of different hardware components that are being used in the project, reasons to choose these components and how the hardware will interact with the software components. The main component of the project is mentioned in Fig 2.1.



Parts Needed:

Part Description	Link	Approximate Cost
Magnet Car	Blue, Red	$92
M2x6 screws (4)	mcmaster.com/#91292a831/=177k4rp	$6.38 *
M2.5x12 screws (8)	mcmaster.com/#91292a016/=177k574	$4.80 *
M2.5 nuts (8)	mcmaster.com/#91828a113/=177k7ex	$5.64 *
M2.5 washers (8)	mcmaster.com/#93475a196/=177k7x6	$1.58 *
USB Battery with microUSB cable (any battery capable of 2A 5V output is sufficient)	amazon.com/gp/product/B00P7N0320	$17
Raspberry Pi 3	amazon.com/gp/product/B01CD5VC92	$38
MicroSD Card (many will work, I like this one because it boots quickly)	amazon.com/gp/product/B01HU3Q6F2	$18.99
Wide Angle Raspberry Pi Camera	amazon.com/gp/product/B00N1YJKFS	$25
Female to Female Jumper Wire	amazon.com/gp/product/B010L30SE8	$7 *
Servo Driver PCA 9685	amazon.com/gp/product/B014KTSMLA	$12 **
3D Printed roll cage and top plate.	STL Files: thingiverse.com/thing:2260575 Purchase: DonkeyCar Store	$45 ***

Figure 2. 1 Block Diagram

2.1 Hardware Components

The hardware components used in this project are as follows:

1. Camera
2. Raspberry pi
3. Servo motors
4. Gear motors
5. 16 Channel pwm i2c
6. Electronic speed controller

2.1.1 Camera

The camera is an essential component of this project as it is used to capture images of the objects. It captures images during live video streaming.

2.1.2 Raspberry Pi

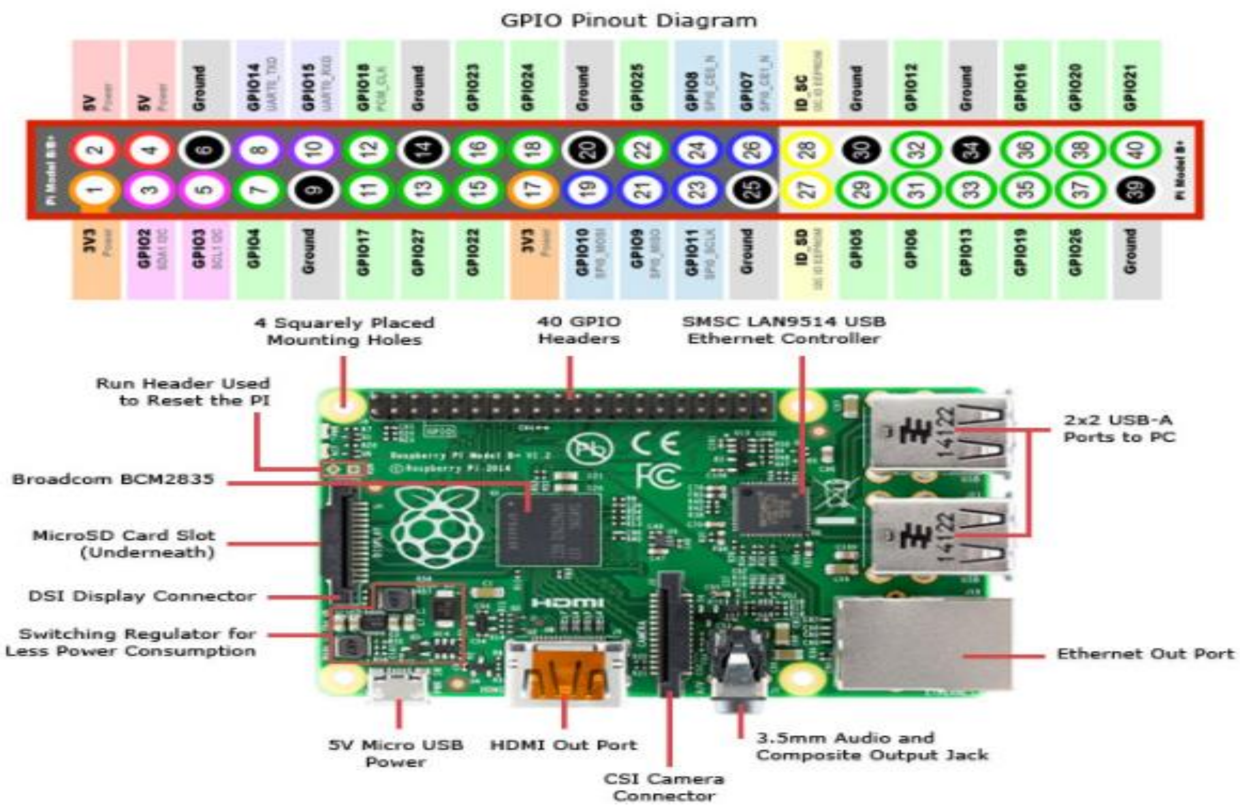


Figure 2. 2 Raspberry Pi board

The Raspberry Pi is a credit card size computer essentially a wireless Internet capable system-on-a-chip (SoC) with connection ports, a Micro SD card slot and 1 GB RAM, camera and display interfaces and an audio/video jack. Quad Core 1.2GHz Broadcom BCM2837 64bit CPU Raspberry Pi 3 is faster and more powerful than its ancestors. It has 40 GPIO pins including UART, I2C bus, SPI bus, 5V and GND

Specifications

1. Processor
2. 1.2GHz Quad-Core
3. BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
4. GPU
5. Memory
6. 1GB
7. Operating System
8. Operating system is installed on class 10 or higher Micro SD card, running various distribution of the Linux operating system

2.2 Hardware Quality Attributes

2.2.1 Reliability

System doesn't fail very often, but still is not reliable under different circumstances and conditions such as unstable Wi-Fi connection.

2.2.2 Extendibility

It is easy to extend this project by adding more features to it other than the ones already included.

2.2.3 Testability

The system performance and features are tested by providing a specific environment.

2.2.4 Response Time

The response time of the system will be made speedy and accurate to a great extent.

1. video stream performance : 10fps

2. prediction response time: 0s(almost instant)

2.3 Assumptions and Dependencies

1. This product is dependent upon clearly marked lane
2. It only identifies the stop sign
3. It only detects pedestrians, not animals
4. The product solely depends upon the operating environment
5. For object detection, there should be good lightning

Chapter#3

System Design

3 System Design

In this chapter, project functionality is explained through the diagram within the shape of the Use case diagram, data Flow diagram, deployment diagram and activity diagram

3.1 Data flow Diagram

There are various types of flow diagrams, some of them are mentioned below.

3.1.1 Context Flow Diagram

The context flow diagram is a zero level data flow diagram. It contains only one process node that is used to describe the relationship of the entire system with the external entities.

In this context flow diagram, the main process is Self-Driving Car where the two external entities are camera and vehicle as shown in the Fig 3.1

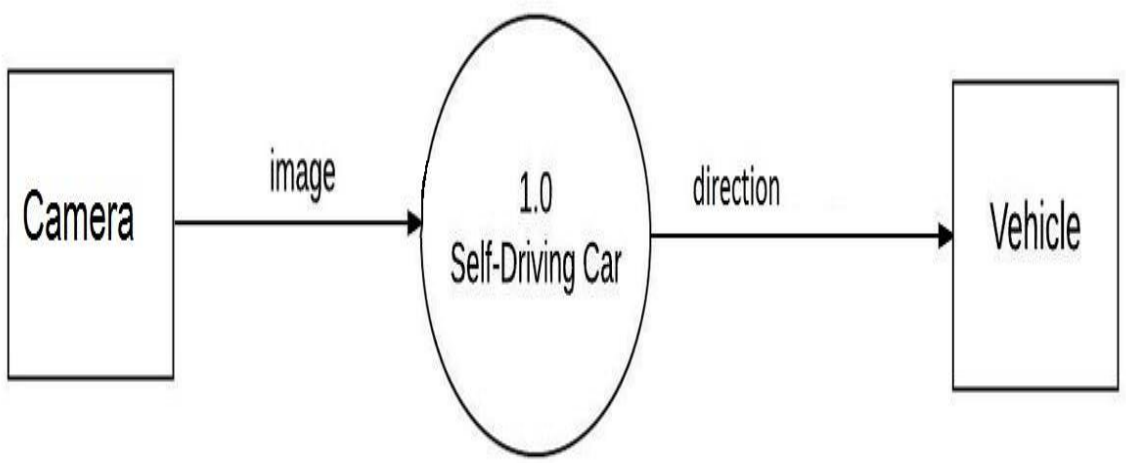


Figure 3. 1 Context Level

3.1.2 Level 1 DFD

The Fig 3.2 highlights the main functionality of the system. In this case the pre-processed images from the camera are passed on to the train model from where the labelled images are used for the prediction by the vehicle in steering.

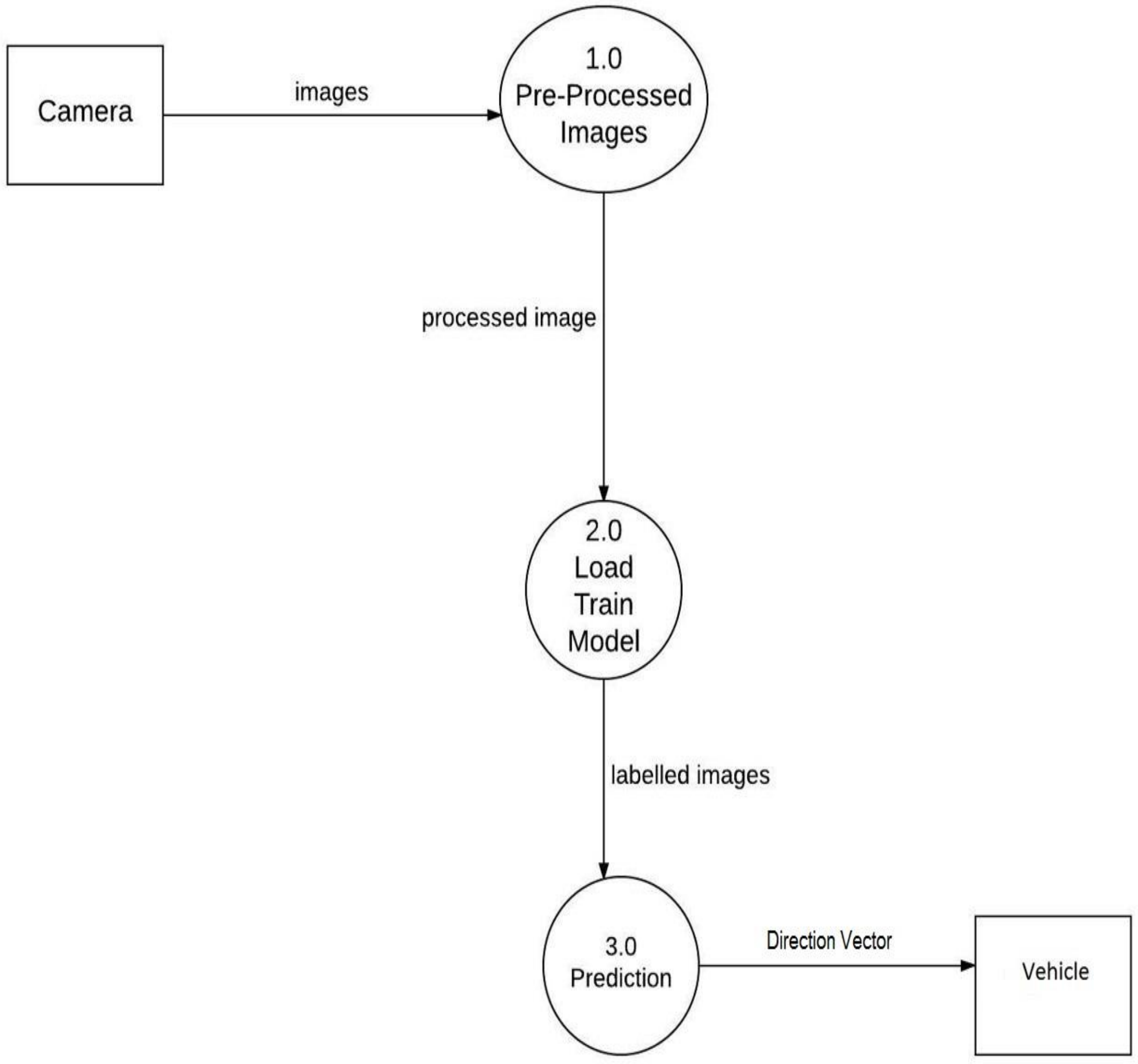


Figure 3. 2 Level 1 DFD (Testing)

3.1.3 Context Flow Diagram

The Figure 3.3 is the level 0 DFD for training with a single process names Self-Driving Car along with three external entities user, camera and vehicle. Train is the data store for storing data for further use.

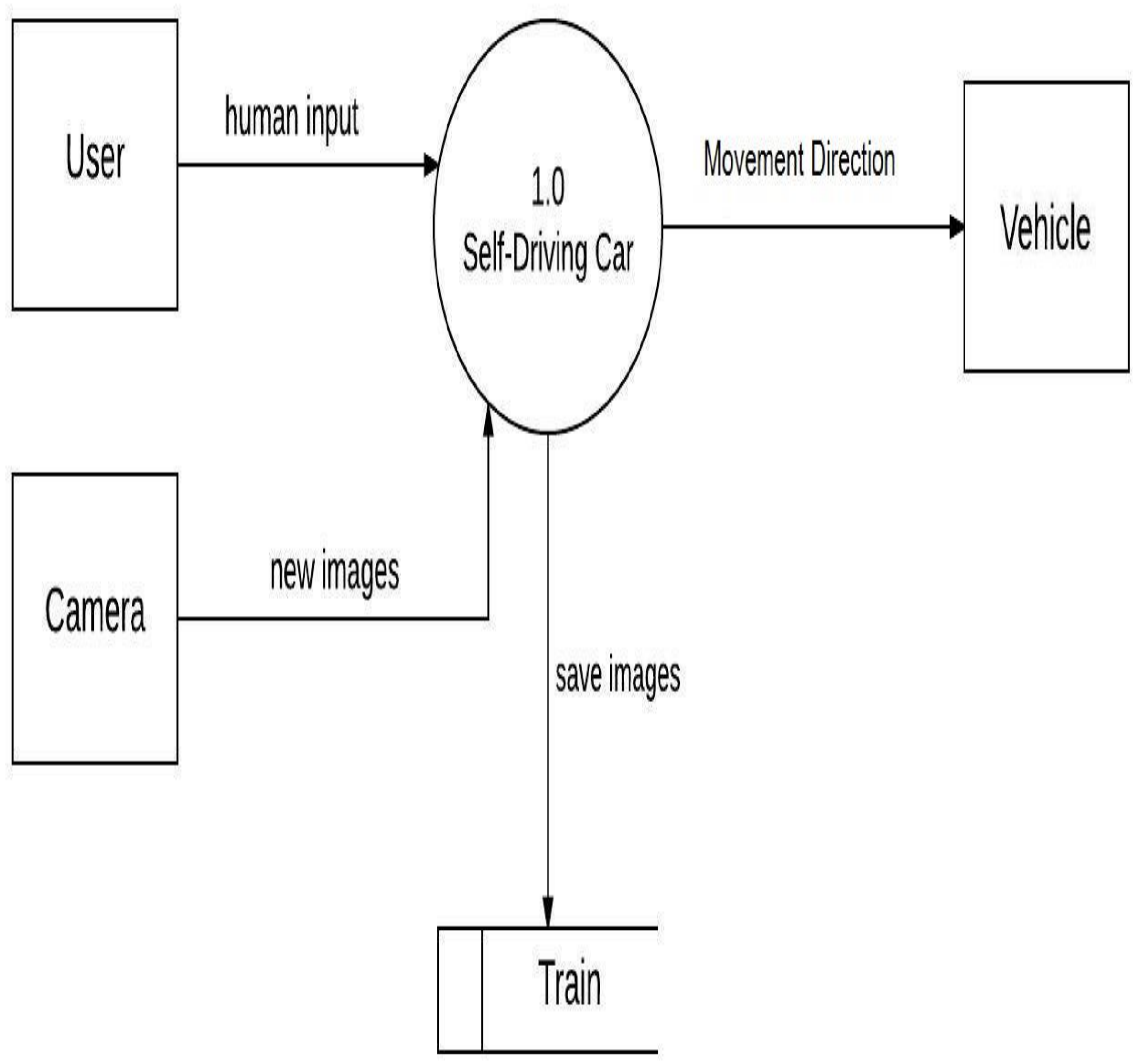


Figure 3. 3 Level 0 DFD (Training)

3.1.4 Level 1 DFD

The following figure 3.4 provides a deep explanation where the process pre-processed images contains images and direction from external entities camera and user. These processed images are sent to another process named bind image with direction from where the labelled images are sent to the third process i-e train model for the purpose of training.

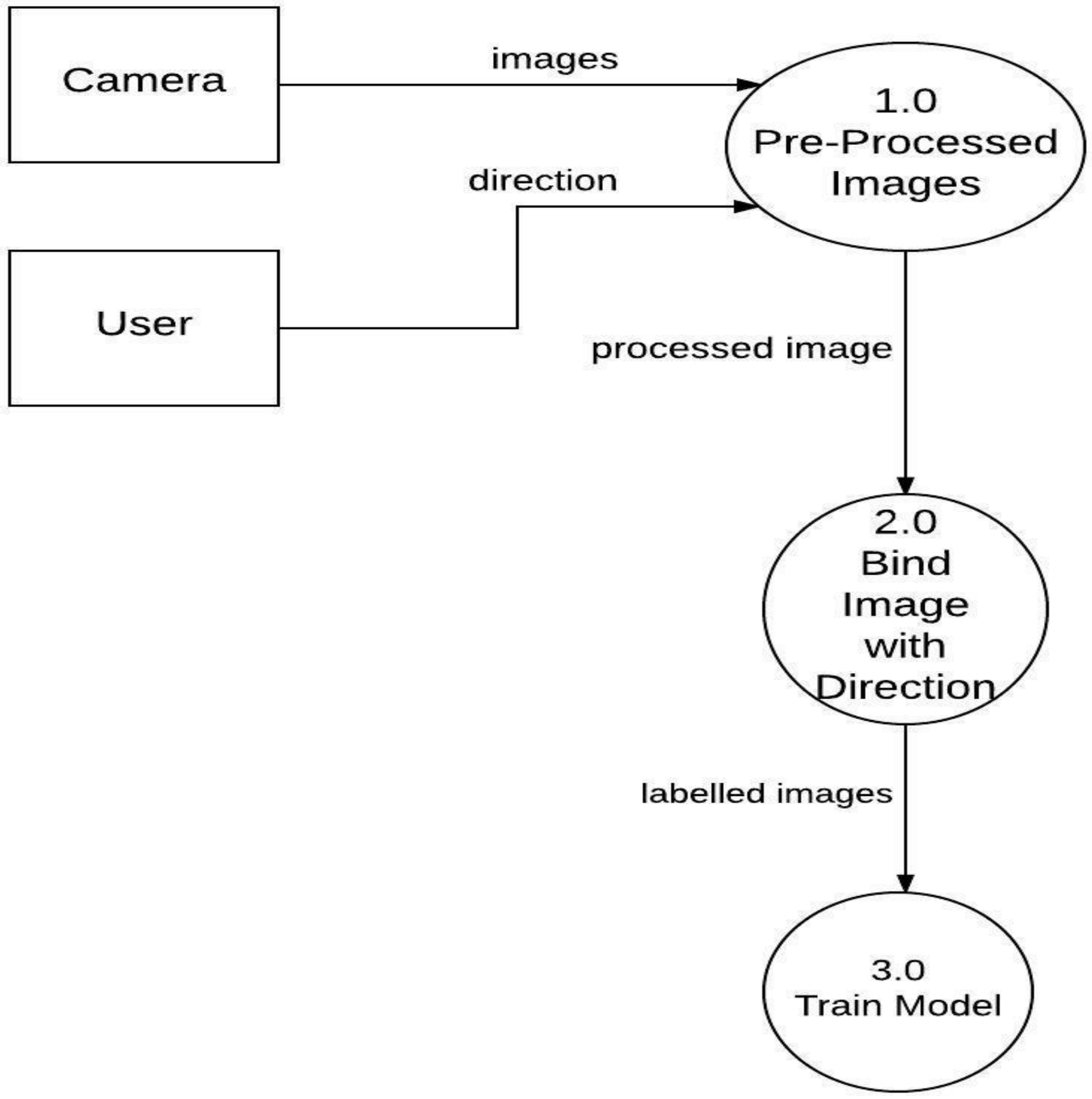


Figure 3. 4 Level 1 DFD (Training)

3.2 Activity Diagram for Testing

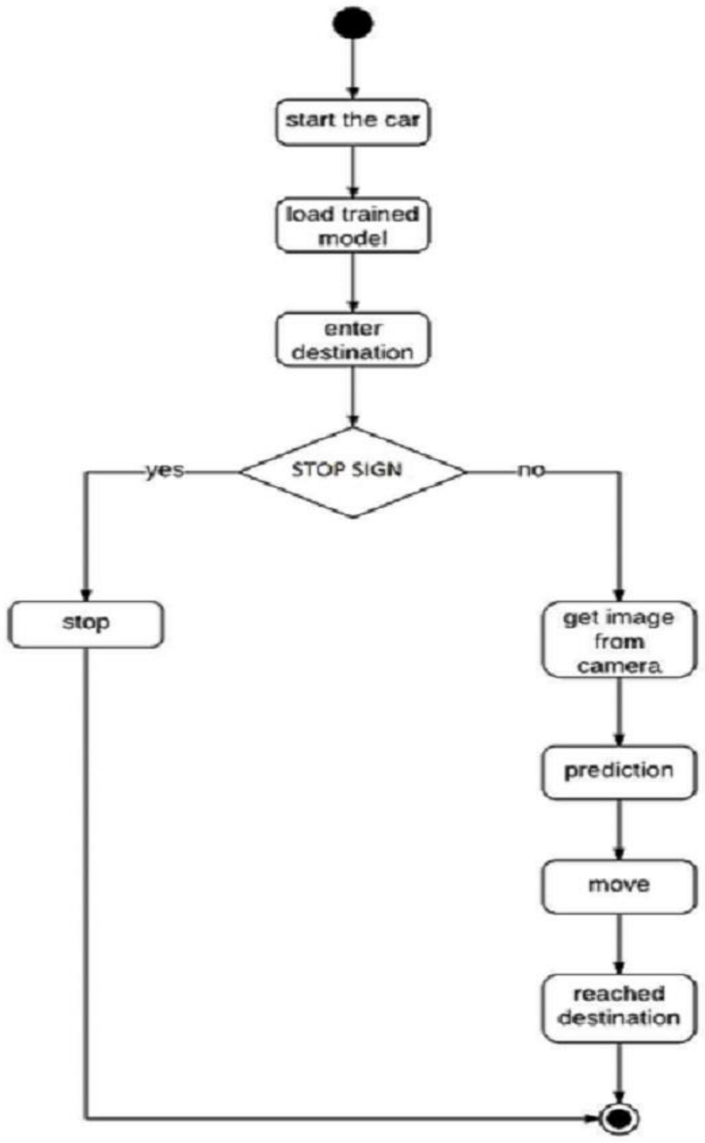


Figure 3. 5 Activity Diagram (Testing)

3.3 Activity Diagram for Training

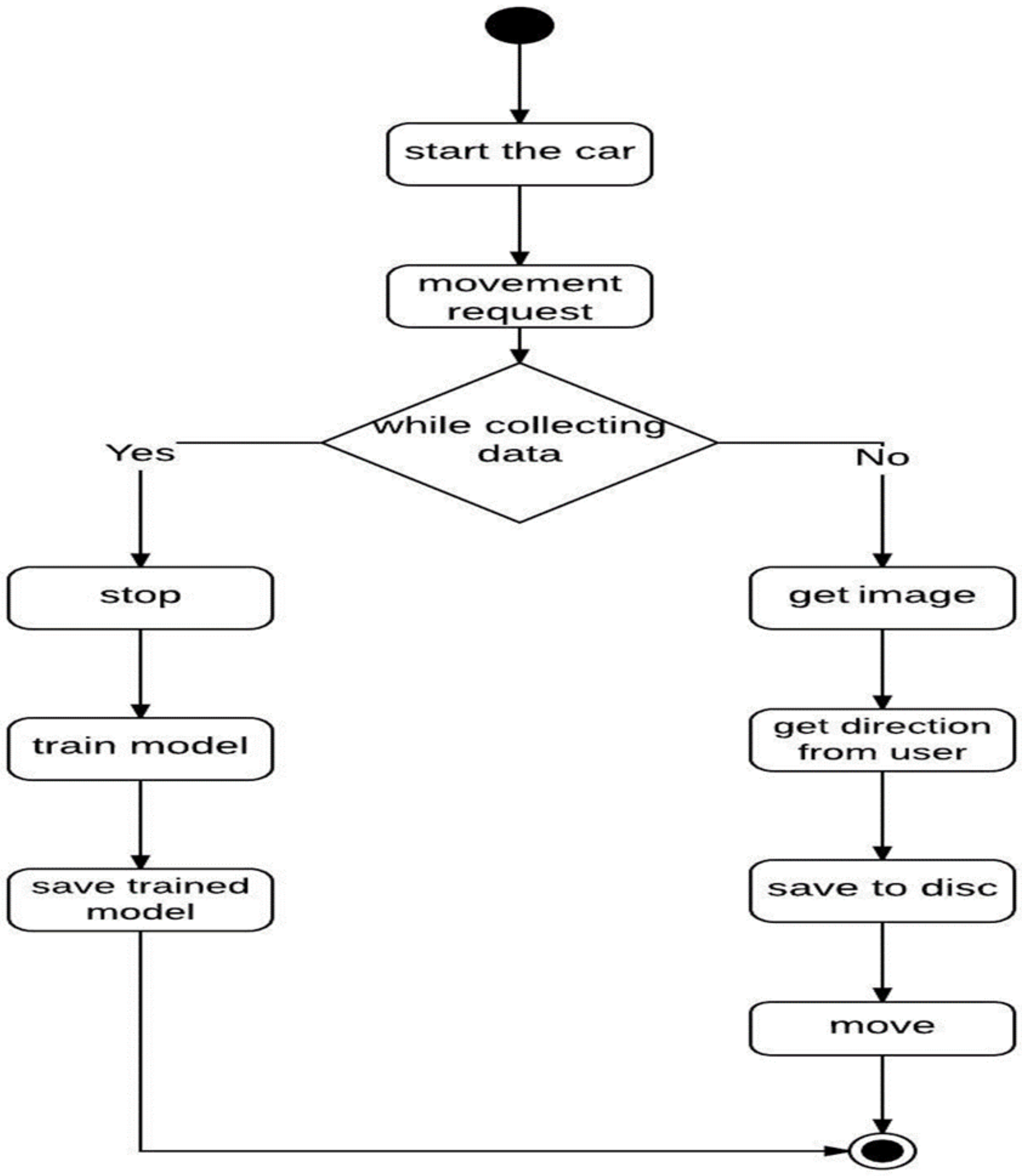


Figure 3. 6 Activity Diagram (Training)

3.4 Use Case Diagram

3.4.1 Use case Diagram for Testing

The following figure 3.5 shows different use cases that are used in our system for testing such as set destination and drive with two primary actors' user and car.

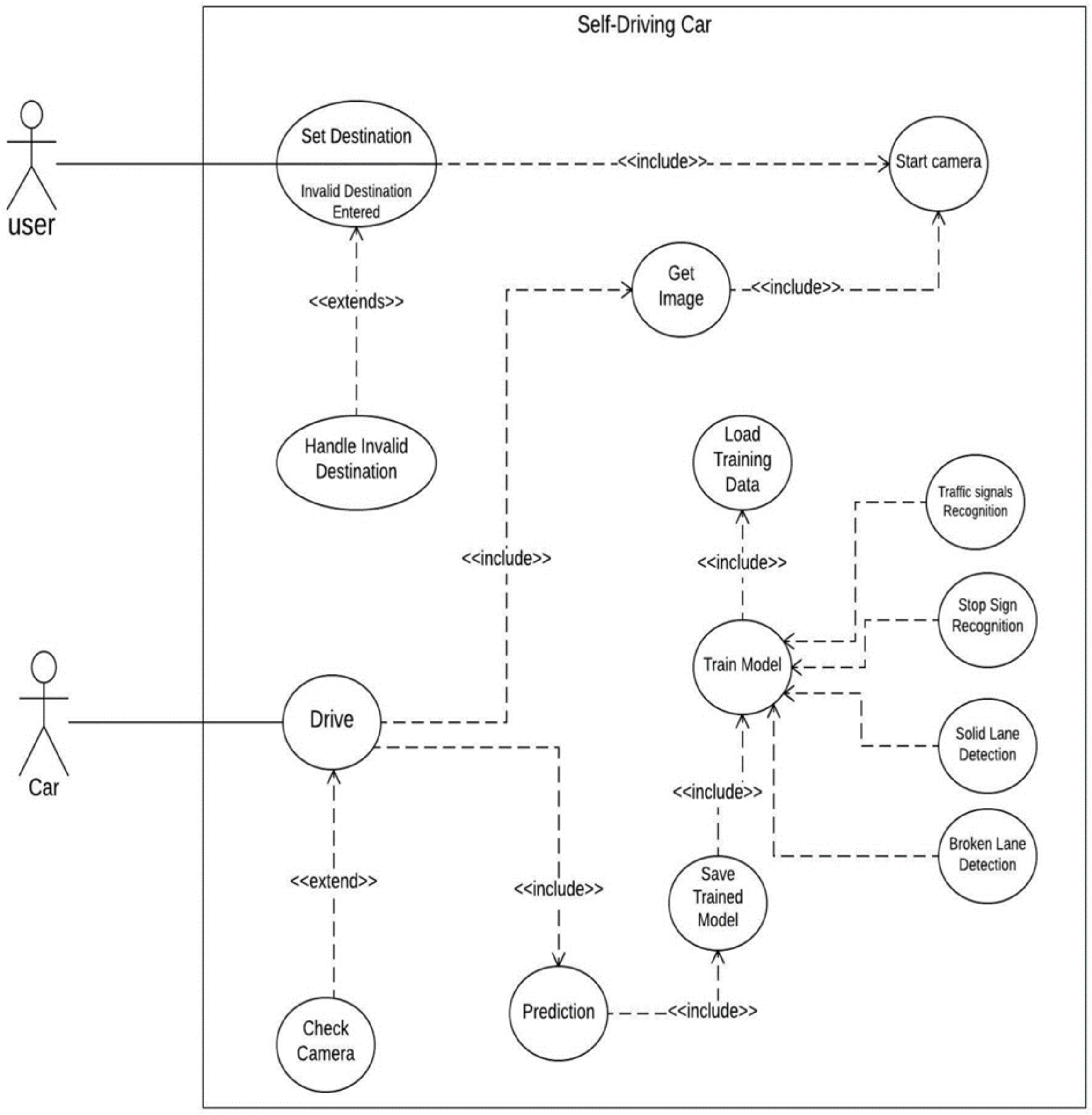


Figure 3. 7 Use case Diagram (Testing)

The figure 3.8 below consists of a use case request movement with user as a primary actor and car as the secondary actor.

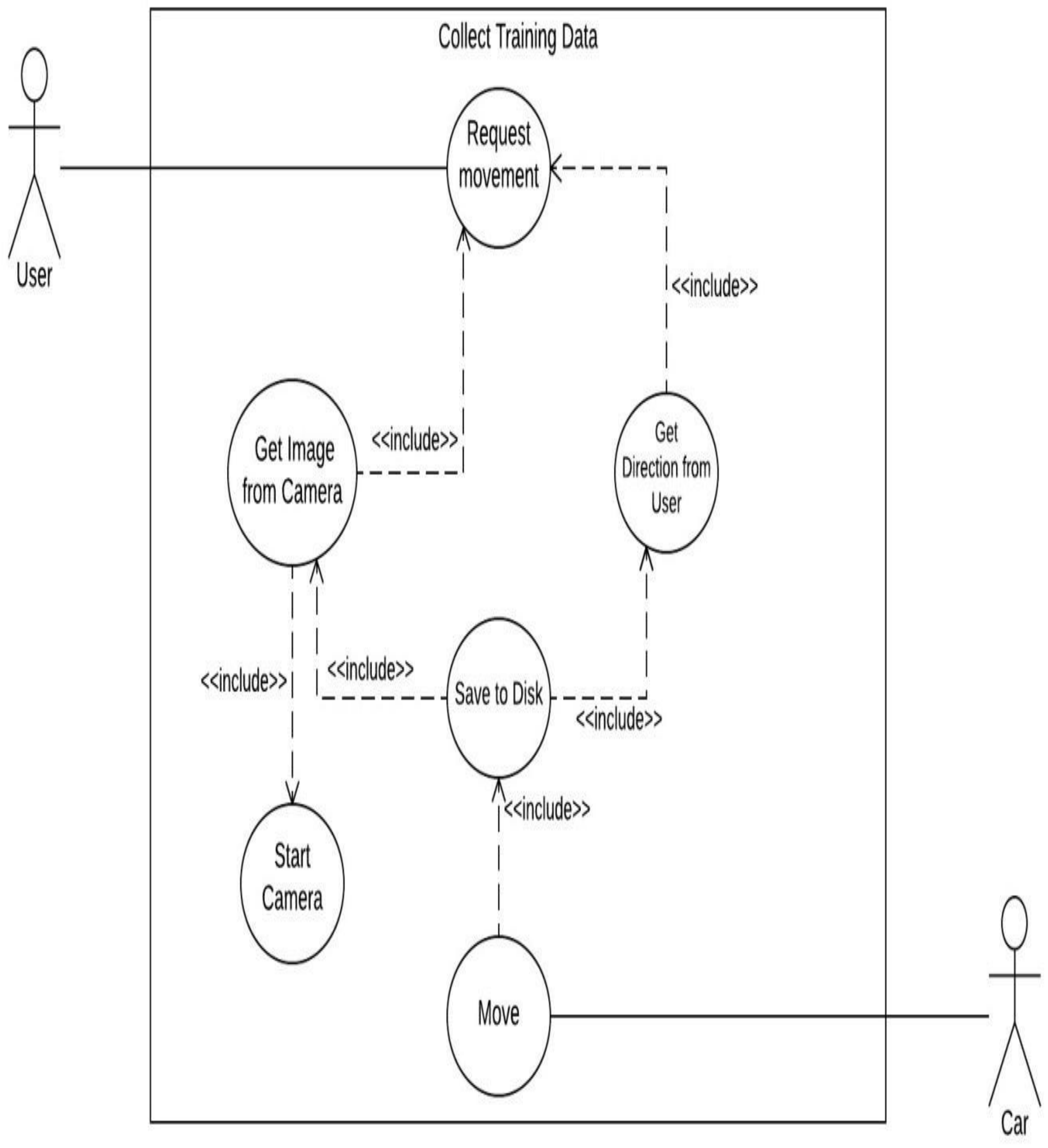


Figure 3. 8 Use case Diagram (Training)

3.5 USECASE NAME: SET DESTINATION

Name: Set Destination.

Scope: Self-Driving Car.

Level: Primary level.

Primary Actor: User.

Stakeholders Interest: User – Needs to enter destination**.**

Preconditions: User has to enter the desired destination after entering the car.

Post conditions: After the user enters the desired destination, the system administrator starts the camera.

Main Success Scenario: The user reaches the desired destination without any distraction in no time.

Technology and Data Variation list: Camera.

Frequency of Occurrence: Occurs frequently.

Miscellaneous: In case of wrong destination entered the system causes delay.

3.6 USECASE NAME: START THE CAMERA

Name: Start the Camera.

Scope: Self Driving Car.

Level: Primary level.

Primary Actor: User.

Stakeholder Interests: -User: wants to be transported without any driver.

Preconditions: User starts the car, and sets the destination, thus turning on the camera. The camera takes images of the path and predicts the next move based on the stored data from the disc.

Post conditions: After turning the camera on images of the path are being taken which are compared to the saved trained model.

Main success scenario: The movement of the car depends upon the images taken by the camera and so the car moves without any problem based upon the prediction from the saved trained model.

Extensions: There might be a condition where the camera may not start because of poor connections.

Special Requirements:

1. The touchscreen UI on a lcd panel. Text should be visible from the distance of at least half meter.
2. Start-up response within 30 seconds 90% of the time.

Technology and Data variation list: Raspberry pie, Camera.

Frequency of occurrence: Occurs frequently.

Miscellaneous: There are no such issues here.

3.7 USECASE NAME: Request Movement

Brief: User requests movement.

Name: Request Movement.

Scope: Self Driving Car.

Level: Primary level.

Primary Actor: User.

Preconditions: User requests for the movement of the car.

Post conditions: User enters destination and so the camera starts automatically. The movement of the car depends upon the directions from the user as well as the stored images from the disc.

Main success scenario: The car moves by getting directions from the user as well as the images from the camera.

Extensions: There might be a situation where the camera may not start or the user gets distracted while giving directions**.**

Technology and data variation list: Camera, disc.

Frequency of occurrence: The car is trained to move on a special designed path and is tested continuously.

Miscellaneous: No such issues**.**

3.8 USECASE NAME: Drive

Name: Drive.

Scope: Self-Driving Car.

Level: Primary level.

Secondary Actor: System.

Preconditions: The car drives based on the availability of the camera.

Post conditions: The car drives by predicting the training data as well the images stored in the disc**.**

Technology and data variation list: Camera, disc.

Frequency of occurrence: Occurs frequently.

Miscellaneous: No such issues.

Chapter#4

System Architecture

4 System Architecture

This chapter defines the structure and behavior of the system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system

4.1 Convolutional Neural Network:

In machine learning, A Convolutional Neural Networks belongs to the class of Deep and feed-forward Artificial Neural Networks consisting of nodes or neurons that have learnable weights and biases. CNNs has successfully been applied in areas such as image recognition and classification. In case of ConvNet primary goal of convolution is to extract features from the input image. One of the main advantages of CNNs is the fact that proximity is strongly correlated with similarity. That means if two pixels are near one another in a given image, they are more likely to be related than two pixels that are further apart. Convolution solves the issue of connecting every pixel to every single neuron in conventional Neural Networks by simply killing a lot of these less important connections. Thus, each neuron is only responsible for processing a certain part of an image, similar to cortical neurons in human brain.

There are four basic operations in the Convnets containing one or more of each of the following layers

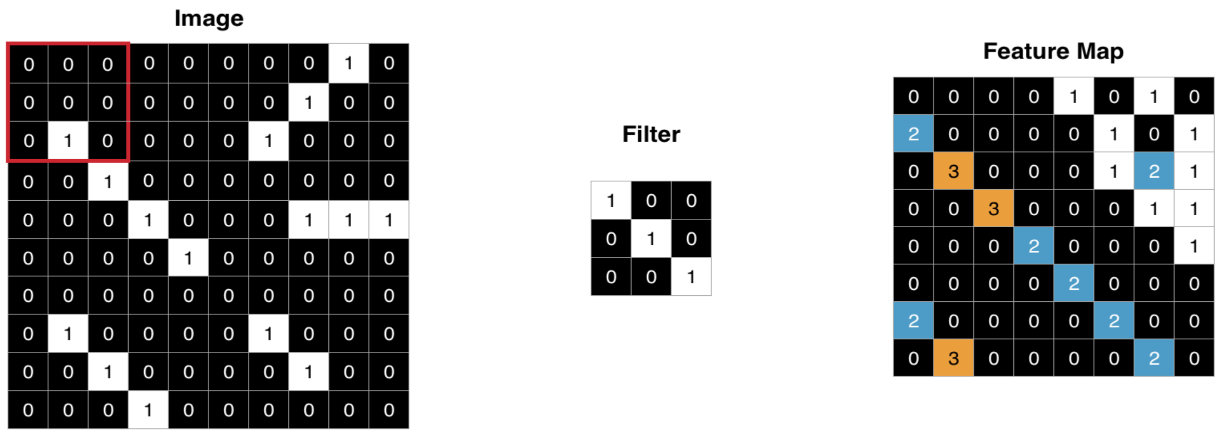
- Convolution Layer
- Non Linearity (ReLU) Layer
- Pooling Layer
- Fully Connected Layer

4.1.1 Convolution:

ConvNets are originated from the idea of "convolution" operator. The key feature of Convolution in instance of a ConvNet is to extract features from the input image.

Convolutional Neural Networks contain multiple types of layers through which all data is fed. These layers are arranged in a hierarchical manner and can include convolution layers, pooling layers, fully connected layers, and a loss layer. In Convolutional Neural Network features are detected by convolving a small window of weights called filter or kernel.
Filters acts as feature detectors from the original input image. Moreover filters are used to find where in an image details such as horizontal lines, curves, colors, etc.

The filter below finds right-sloping diagonal lines.



0 x 1 + 0 x 0 + 0 x 0 + 0 x 0 + 0 x 1 + 0 x 0 + 0 x 0 + 1 x 0 + 0 x 1 = 0

Figure 4. 1 Convolution

4.1.2 Non-Linearity (ReLU):

ReLU short for Rectified Linear Unit, layer commonly follows the convolution layer. A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. ReLUs' machinery is more like a real neuron in your body. ReLUs result in much faster training for large networks

$$f(x) = max(x, 0)$$

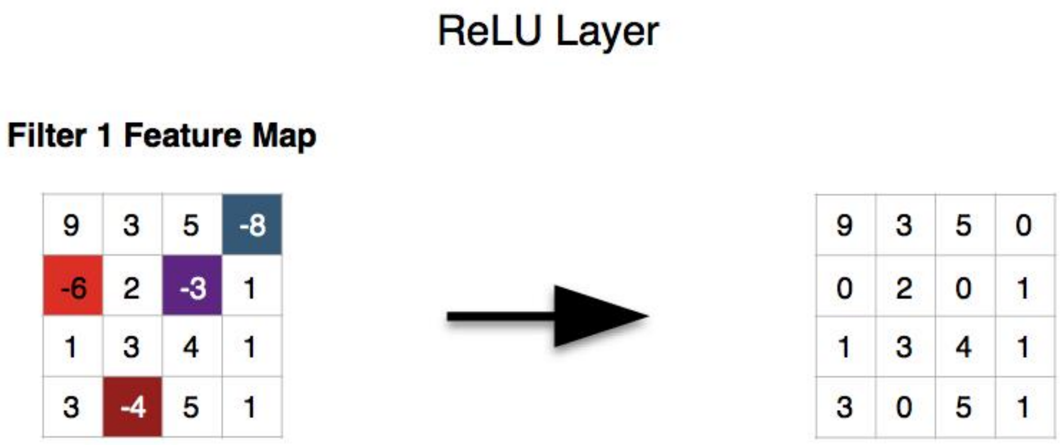


Figure 4. 2 ReLU

There are various non-linear activation functions such as sigmoid or tanh that can also be utilized in place of ReLU, but ReLU generally works better in practice. The purpose of ReLU is to introduce non-linearity in our ConvNet, the ReLU function is applied to each point in the feature map. The result is a feature map without negative values.

4.1.3 Pooling Layer :

Additional significant concept of CNNs is pooling, there are numerous non-linear functions that can be used for pooling among which max pooling is the utmost common, in the illustration below image, uses a 2×2 pixel window and a filter of 2. After taking the largest value in the window, convolve the window over by 2 pixels, and repeat

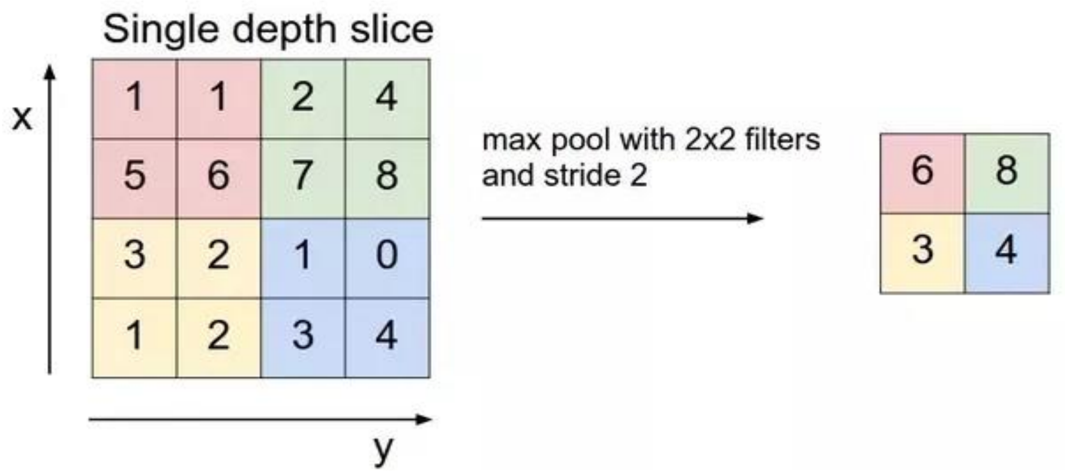
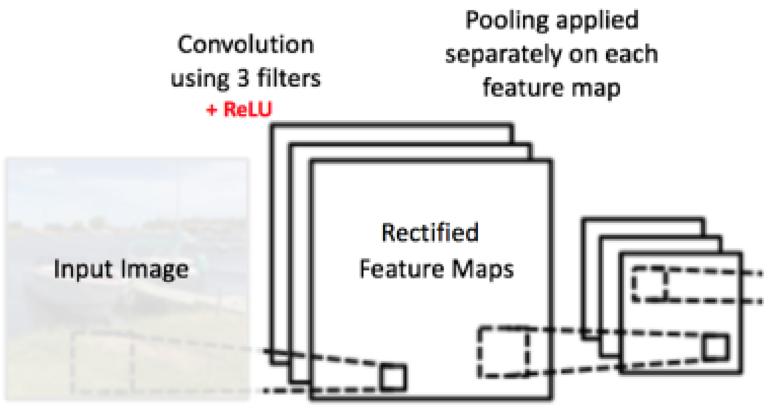


Figure 4. 3 Max Pooling

The function of Pooling is to reduce the spatial size of the input representation. In particular, pooling

- Max pooling makes the feature dimension smaller and more manageable
- Pooling also reduces the size of the feature maps, making computation easier in later layers, results in controlling overfitting
- The max pooling layer makes the network less sensitive to small changes in image

Figure 4. 4 Pooling


Convolution using 3 filters
+ ReLU
Pooling applied separately on each feature map
Input Image
Rectified Feature Maps

4.1.4 The Fully-Connected Layer:

The fully-connected layer is where the final "decision" takes place. The output from the Convolutional layers represents high-level features in the data. While that output could be flattened and connected to the output layer, adding a fully-connected layer is a (usually) cheap way of learning non-linear combinations of these features. Essentially the Convolutional layers are providing a meaningful, low-dimensional, and somewhat invariant feature space, and the fully-connected layer is learning a (possibly non-linear) function in that space. For instance, classifying images of road's boundaries leads to four probable outputs as shown in Figure 4.5 below

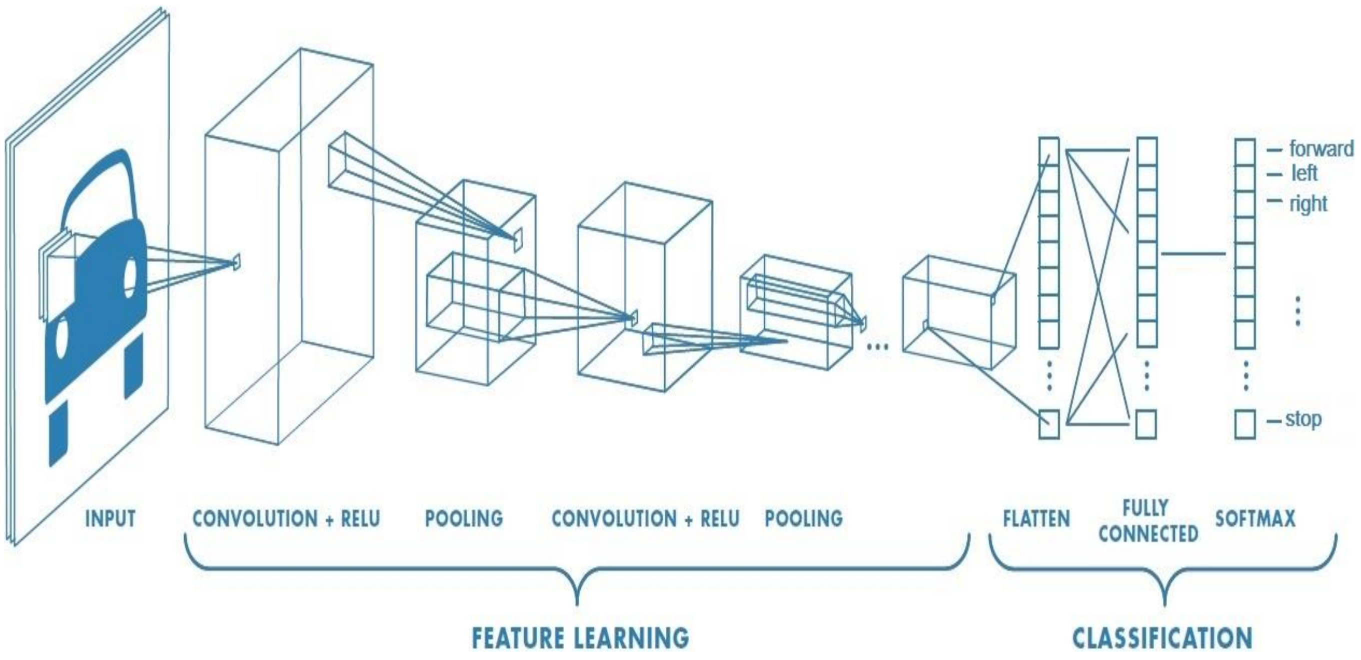


Figure 4. 5 Classification

4.2 Training Process:

The entire training process of the Convolution Network is be summarized as below:

- **Step1:** Determine hyper parameter which are number and size of filters in each Convolutional layer

- **Step2:** Feed the images into network from our training dataset the network which goes through the process of forward propagation i.e.,(convolution, ReLU and pooling operations beside backpropagation in the Fully Connected layer)
    - Output probabilities for the road image above are [0.2, 0.5, 0.3]

- **Step3:** Compute the mean squared error at the output layer

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y_i} - Y_i)^2$$

- **Step4:** Use Backpropagation learn from mistakes, weights are readjusted in proportion to calculate error repeat until error is less than a predefined threshold value.
    - Compute the classification from the Neural Network, and compare to the known value. This gives you an error at the output node. Backpropagation is trying to do a gradient descent on the error surface of the Neural Network, adjusting the weights with dynamic programming techniques to keep the computations tractable

- **Step5:** Repeat steps 2-4 with all images in the training set

4.3 Backpropagation:

Backpropagation is like “learning from mistakes“. In supervised learning, the training set is labeled, the supervisor corrects the Neural Network whenever it makes mistakes.

Idea behind BP algorithm is quite simple, output of Neural Networks is evaluated against desired output. If results are not satisfactory, connection (weights) between layers are modified and process is repeated again and again until error is small enough.

4.4 ConvNet Architecture:

Designing CNN architectures for optimal output is a tough and many a times an empirical job, however there are specific tricks and techniques which are used to design the network.

- First is the Convolutional layers (in this case using a 3x3 kernel)

- A rectified linear unit (RELU) on the Convolutional layer output)

- A pooling layer where the image is halved in every dimension.

As we go down the contracting path, the number of feature channels is doubled every time the input goes through a Convolutional layer, but the size is reduced every time it goes through a pooling layer. The expanding path is more or less symmetrical, with up sampling layers that increase (double) the size of the input, followed by a typical Convolutional layer and a rectified linear unit (RELU). While we go up the expanding path, the number of feature channels is reduced in the Convolutional layers, until you end up with a single matrix (array) that corresponds to the output image.

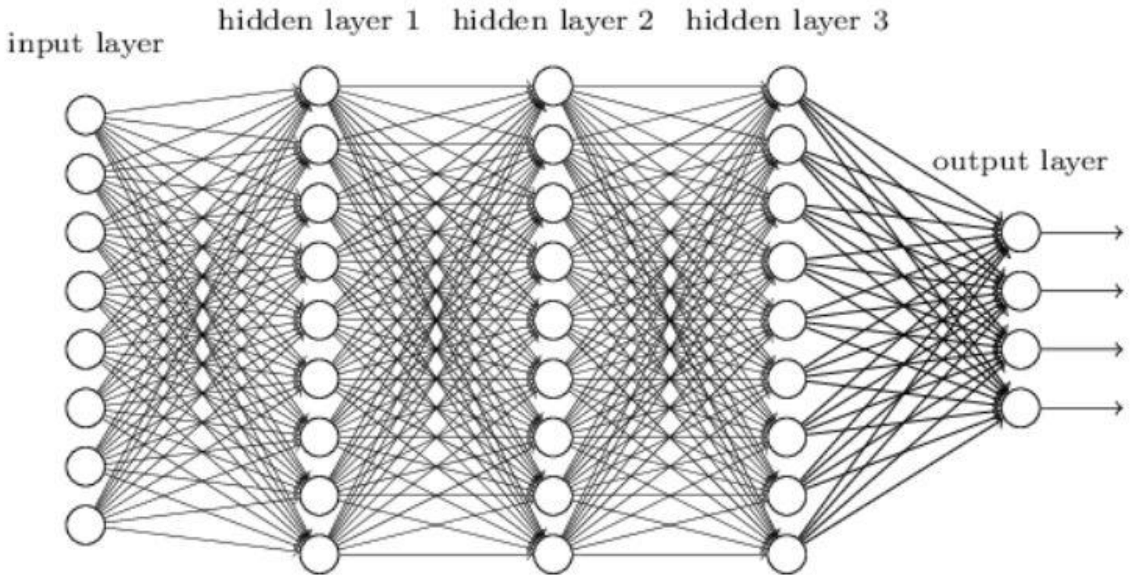


Figure 4. 6 Network Architecture

After the three Convolutional layers which are responsible of feature engineering, these feature maps are then fed to three fully connected layers which perform all of the complex computation leading to an output control value. As it's a fully connected layer. Every neuron from the last max-pooling layer is connected to every layer of the fully-connected layer the fully connected layers are designed to function as a controller for steering.

The basic working principle is that at the end of the network we have inserted fully-connected layer with N output channels, where N is the number of classes. Then we proceed to apply global average pooling, which will produce a 1D hot vector of N elements (independent of input feature map width/height), and we have applied the softmax function to that vector to produce output class probabilities.

Chapter#5

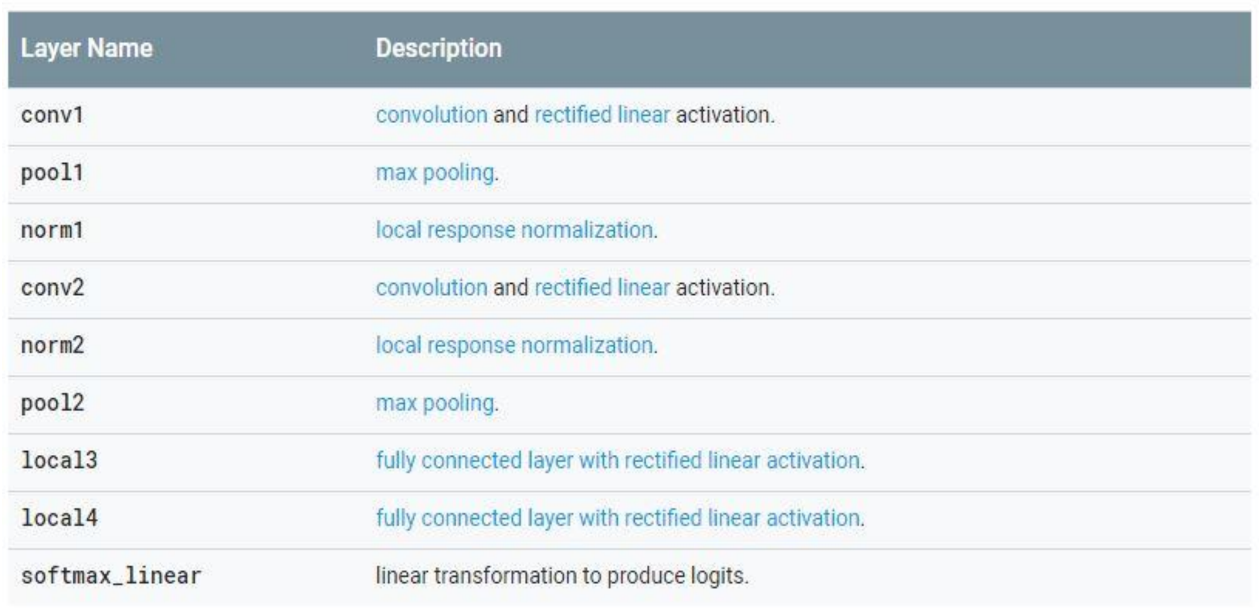
Tests and Results

5 Tests and Results:

In software development, we need to check functionality in the developing product to make sure whether it fulfills requirement or not. In this chapter we talk about system Test Plan and Test cases. This analysis is used to know about the functionality of the system that is it prepared in the expected way that we will obtain or not.

5.1 Model Architecture:

The model of our CNN is multi-layered architecture comprising of three convolutions, nonlinearities and pooling layers. These Convolutional layers are then followed by two fully connected layers containing 64 and 34 neuron per layer respectively, leading into a soft-max classifier. This model accomplishes the maximum performance of around 95% accuracy within less than 15 hours of training time on a NVIDIA gtx 1080 GPU.



Layer Name	Description
conv1	convolution and rectified linear activation.
pool1	max pooling.
norm1	local response normalization.
conv2	convolution and rectified linear activation.
norm2	local response normalization.
pool2	max pooling.
local3	fully connected layer with rectified linear activation.
local4	fully connected layer with rectified linear activation.
softmax_linear	linear transformation to produce logits.

Figure 5.1 Execution order

5.1.1 Model Training:

The process for training a Neural Network to do multiple classification is multinomial logistic regression problem, difference between multinomial and univariate logistic regression is that univariate logistic regression uses logistic function aka sigmoid which takes a real numeric value and limits it between range of 0 to 1 . The softmax can be used for any number of classes. It's also used for hundreds and thousands of classes, for example in object recognition problems where there are hundreds of different possible objects.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

5.2 Precision and Recall:

Precision represents the actual number of True Positives classifications divided by the number of True Positives and False Positives classifications. Recall and precision measure the quality of your result. To understand them let's first define the types of results. A document in your returned list can either be

5.2.1 Classified correctly

- A true positive (TP): a document which is relevant (positive) that was indeed returned (true)
- A true negative (TN): a document which is not relevant (negative) that was indeed NOT returned (true)

5.2.2 Misclassified

- A false positive (FP): a document which is not relevant but was returned
- A false negative (FN): a document which is relevant but was not returned

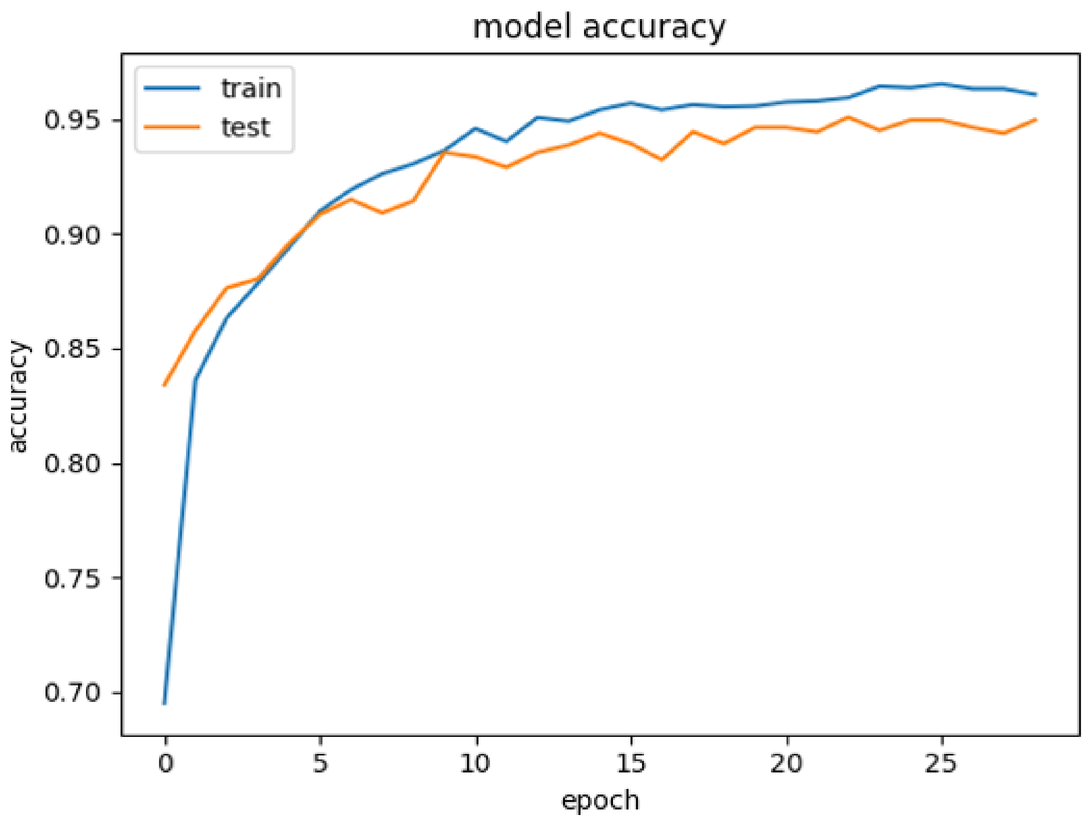


Figure 5.2 Accuracy

5.3 Loss Functions:

One of the most important part of Artificial Neural Network is loss function, used to measure the inconsistency between predicted value (y hat) and actual label (y).

5.3.1 Mean squared Error:

Mean squared error is the most common loss function in machine learning, I believe it is the most intuitive loss function for every machine learning beginner.

The formula is:

$$MSE := \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

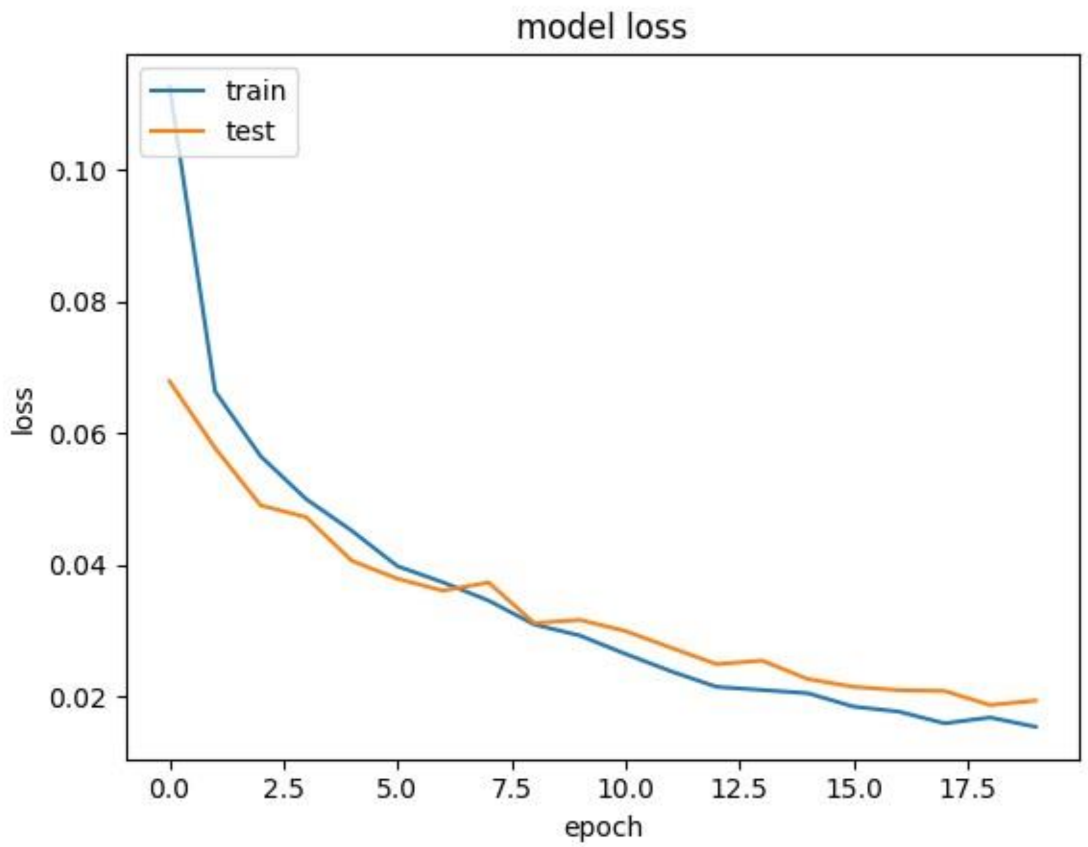


Figure 5.3 Model Loss

5.4 Confusion Matrix:

The confusion matrix is a way of tabularizing the number of misclassifications, i.e., the number of predicted classes, which finally be in a particular wrong classification bucket based on the true classes.

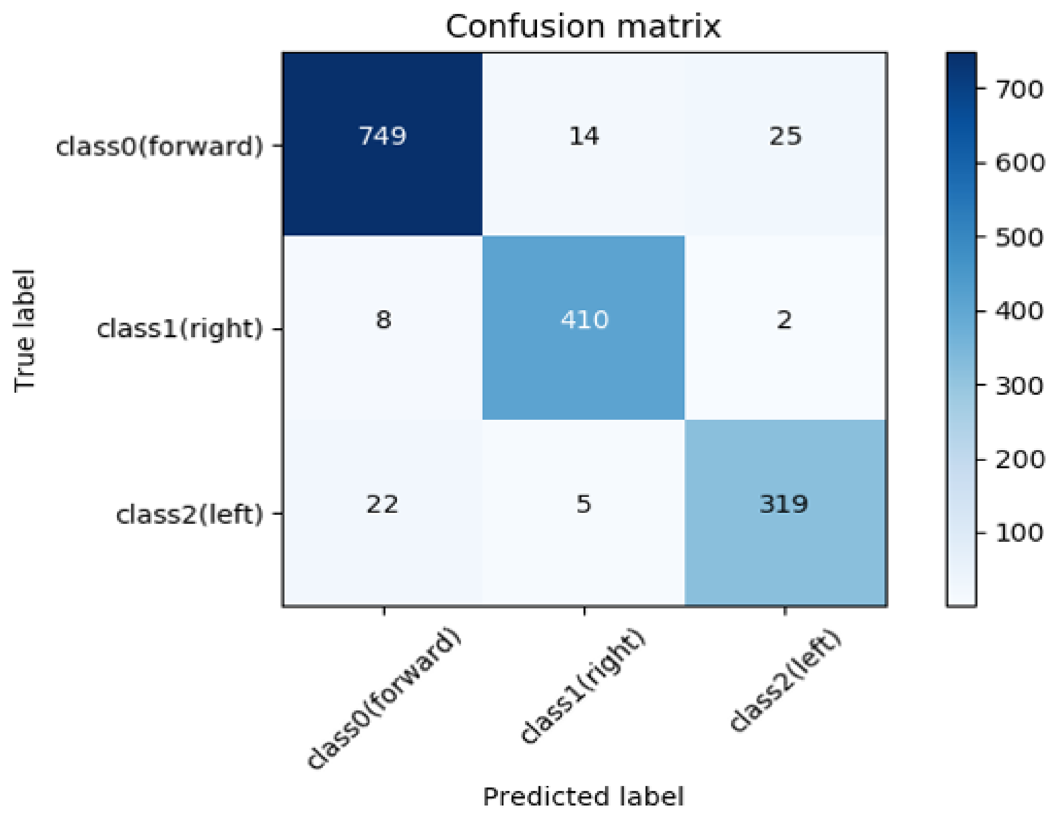


Figure 5.4 Confusion Matrix

Definition of the Terms:

This allows us to see that:

- The elements on the diagonal represents the number of correct classifications for each class.
- The off-diagonal elements provides the misclassifications.
- The total number of classifications for each class in both y_true and y_pred, from the "All" subtotals.

Chapter#6

Conclusion

6 Conclusion

In this thesis, a method to make a self-driving robot car is presented. The different hardware components and their assembly are clearly described. We have proved that practically CNNs can learn the entire task of lane and road following without being explicitly programmed for marking detection and control. A small amount of training data comprising of 9,654 images captured from less than 15 hours of manual driving was sufficient to train the network to operate in various tracks and lighting conditions.

Prediction on the testing samples returns an accuracy of 95% in actual driving situation, predictions are generated about 15 times a second (streaming rate roughly 15 frames/s).

Once the Convolution Neural Network has learnt form the examples to detect the boundaries of solid and broken lane-markings on a road without being explicitly trained to detect outlines. Moreover, additional work is required to improve the robustness and accuracy of the network, to define a metric to evaluate the robustness, and to improve visualization of Convolutional layer and network-internal processing steps.

Chapter #7

References

7.1 REFERENCES

[1] [n. d.]. Baidu Apollo. https://github.com/ApolloAuto/apollo. ([n. d.]).

[2] [n. d.]. Tesla Autopilot. https://www.tesla.com/autopilot. ([n. d.]).

[3] 2013. Add Dramatic Rain to a Photo in Photoshop. https://design.tutsplus.com/ tutorials/add-dramatic-rain-to-a-photo-in-photoshop--psd-29536. (2013).

[4] 2013. How to create mist: Photoshop effects for atmospheric landscapes. http://www.techradar.co m/how-to/photography-video-capture/cameras/ how-to-create-mist-photoshop-effects-for-atmospheric-landscapes-1320997. (2013).

[5] 2014. The OpenCV Reference Manual (2.4.9.0 ed.).

[6] 2014. This Is How Bad Self-Driving Cars Suck In The Rain. http://jalopnik.com/ this-is-how-bad-self-driving-cars-suck-in-the-rain-1666268433. (2014).

[7] 2015. Affine Transformation. https://www.mathworks.com/discovery/ affine-transformation.html. (2015).

[8] 2015. Affine Transformations. http://docs.opencv.org/3.1.0/d4/d61/tutorial_warp_ affine.html.(2015)

[9] 2015. Open Source Computer Vision Library. https://github.com/itseez/opencv. (2015).

[10] 2016. Chauffeur model. https://github.com/udacity/self-driving-car/tree/master/ steering-models/community-models/chauffeur. (2016).

[11] 2016. comma.ai’s steering model. https://github.com/commaai/research/blob/ master/train_steerin g_model.py. (2016).

[12] 2016. Epoch model. https://github.com/udacity/self-driving-car/tree/master/ steering-models/community-models/cg23. (2016).

[13] 2016. Google Auto Waymo Disengagement Report for Autonomous Driving. https://www.dmv.ca .gov/portal/wcm/connect/ 946b3502-c959-4e3b-b119-91319c27788f/GoogleAutoWaymo_disengage_ report_2016.pdf?MOD=AJPERES. (2016).

[14] 2016. Google's Self-
Driving Car Caused Its First Crash. https://www.wired.com/ 2016/02/googles-self-driving-car-may-caused-first-crash/. (2016).

[15] 2016. Rambo model. https://github.com/udacity/self-driving-car/tree/master/ steering-models/community-models/rambo. (2016).

[16] 2016. Udacity self driving car challenge 2. https://github.com/udacity/ self-driving-car/tree/master/challenges/challenge-2. (2016).

[17] 2016. Udacity self driving car challenge 2 dataset. https://github.com/udacity/ self-driving-car/tree/master/datasets/CH2. (2016).

[18] 2016. Who's responsible when an autonomous car crashes? http://money.cnn.com/ 2016/07/07/tec hnology/tesla-liability-risk/index.html. (2016).

[19] 2017. Autonomous Vehicles Enacted Legislation. http://www.ncsl.org/research/ transportation/aut onomous-vehicles-self-driving-vehicles-enacted-legislation. aspx. (2017).

[20] 2017. Inside Waymo's Secret World for Training Self-