Reinforcement Learning Based Autonomous Vehicle for Exploration and Exploitation of Undiscovered Track

by

Razin Bin Issa
16101214
Md. Saferi Rahman
16101011
Modhumonty Das
16101204
Monika Barua
16101262

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
December 2019

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Razin Bin Issa 16101214	Md. Saferi Rahman 1601011
Modhumonty Das 16101204	Monika Barua 16101262

Approval

The thesis titled "Reinforcement Learning based Autonomous Vehicle for Exploration and Exploitation of Undiscovered Track" submitted by

1. Razin Bin Issa (16101214)
2. Md. Saferi Rahman (16101011)
3. Modhumonty Das (16101204)
4. Monika Barua (16101262)

of Fall, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on 24th December.

Examining Committee:

Supervisor:

Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:

Md. Khalilur Rhaman
Associate Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:

Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:

Mahbubul Alam Majumdar
Professor and Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

This research focuses on autonomous traversal of land vehicles through exploring undiscovered tracks and overcoming environmental barriers. Most of the existing systems can only operate and traverse in a distinctive mapped model especially in a known area. However, the proposed system which is trained by Deep Reinforcement Learning can learn by itself to operate autonomously in extreme conditions. The dynamic double deep Q-learning (DDQN) model enables the proposed system not to be confined only to known environments. The ambient environmental obstacles are identified through Faster R-CNN for smooth movement of the autonomous vehicle. The exploration and exploitation strategies of DDQN enables the autonomous agent to learn proper decisions for various dynamic environments and tracks. The proposed model is tested in a gaming environment. It shows the overall effectiveness in traversing of autonomous land vehicles in comparison to the existing models. The goal is to integrate Deep Reinforcement learning and Faster R-CNN to make the system effective to traverse through undiscovered paths by detecting obstacles.

Keywords: Reinforcement Learning; Faster R-CNN; Double Deep Q Learning; Markov Decision Process; Autonomous Vehicle; Object Classifier

Acknowledgement

First of all, we would like to thank Almighty Allah, as we could work on this thesis which has been a great learning experience for us. By the grace of Allah, we were able to put our best efforts and successfully complete it on time.
Secondly, we would like to convey our gratitude to our supervisor Dr. Md. Golam Rabiul Alam and co-supervisor Dr. Md. Khalilur Rhaman for their guidance and handful contribution throughout the whole phase of our thesis work. From the very beginning to the end of the work they have provided us with all kinds of help and inspired us to move forward to our goal.
Thirdly, the reviewers of The 34th International Conference on Information Networking (ICOIN 2020), whose valuable reviews helped us to improve our work.
Last but not the least, we are also very thankful to our parents, friends, and well-wishers who have supported us throughout our research. We would also like to acknowledge the assistance that we received from a number of resources over the Internet especially from related researches.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ϵ Epsilon

CNN Convolutional Neural Network

$COCO$ Common Objects in Context

$CUDA$ Compute Unified Device Architecture (Nvidia)

$DDQN$ Double Deep Q-Network

DQN Deep Q-Network

GTA Grand Theief Auto

MDP Markov Decision Process

MS Microsoft

PEA Parameter estimator algorithm

$R-CNN$ Region Based Convolutional Neural Network

$ReLU$ Rectified Linear Unit

ROI Region of Interest

RPN Region Proposal Network

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Background

Reinforcement Learning [10] involves taking decision to perform suitable action by maximizing reward in a specific circumstance. Various software and machines employ it and it then finds the most suitable decision which should be taken in a particular circumstance [10]. A Reinforcement Learning based agent learns by interacting with the environment. It can decide how to perform a given task on its own from a training dataset. However, it must learn from its experience in the absence of a dataset.

In this modern age, autonomous vehicles are considered one of the most integral parts of intelligent transportation system. An autonomous vehicle first takes perception from the environment, makes a decision, plans it and then controls the vehicle [38]. Thus, it is one of the most emerging technologies that exists today. The brain of an autonomous vehicle is the decision-making module. Integrating reinforcement learning in an autonomous vehicle will help the vehicle perform in any environment through exploitation and exploration. Thus, the vehicle or autonomous agent will be able to take its own decision to traverse and drive in undiscovered tracks.

The conventional ways of autonomous vehicles are limited within certain maps. However, integrating autonomous vehicle and reinforcement learning will make an agent take decisions more efficiently and control it in that manner as the vehicle will be able to explore undiscovered tracks [1]. The proposed research suggests, traversing of an autonomous vehicle which tries to find its own path by detecting and identifying obstacles along the way. It takes data and information of rough and rocky surface and obstacles through sensor data. The agent feeds the data in our proposed algorithm for taking decisions in future based on the fed conditions. The sensor that is mainly used to implement the proposal is Camera.

The agent explores the environment using Double Deep Q-Learning. It estimates more accurate values [35]. The process of the agent always picking the highest q-value for exploration is called the epsilon greedy strategy. The three terms that are used in this strategy are state, reward and action. In order to performing method, the algorithm uses Bellman Equation.

Furthermore, Faster R-CNN [20] will be applied at first for the prototype as the agent and vehicle tries to track and detect objects while traversing. Faster R-CNN, at this time, is one of the most prominent algorithms for object detection. In Faster-RCNN, region proposal and object detections are done using convolutional network which make this algorithm faster in terms of objection detection. By merging the mentioned processes of path finding and object tracking, developing an agent is aimed so that it can find its own path by avoiding obstacles on its way. This algorithms is tested on a gaming environment.

1.2 Research Problem

Traditional ways of driving automobiles and vehicles have made road safety an issue throughout the world. In fact, road traffic accidents reflect as the eight leading cause of death in the world and more than 1.35 million lives are taken and 50 million are injured every year because of it [54]. Further, more than half of road traffic deaths of the world are amongst cyclists, motorcyclists and pedestrians who are often neglected in road traffic system designs in many countries [46]. Without proper measure taken, road traffic deaths by 2030, are anticipated to become the fifth leading reason of death [45]. Hence, it is an alarming issue which does not receive proper attention.

Roads shared by cars, buses, trucks, motorcycles often support economic and social development in many countries [9]. As vehicles are responsible for deaths, failure to find the actual prevention for this may lead to more casualties in the future. Self-driving cars over traditional vehicles become necessary in this manner for better road management.

The technology of autonomous vehicles has been evolving since its invention. As road traffic death remains a problem, The United States Department of Transportation (USDOT) predicts that autonomous cars will reduce traffic deaths by 90% [50]. A reduction of 90% would save 30,000 lives per year. Moreover, effective advent of autonomous vehicles will cause reduction of harmful emissions by 60%, according to Ohio University's Future of Driving Report [51]. The Ohio University study also supports the report that autonomous cars will provide a reduction in fuel economy by between 4% and 10% [19]. Autonomous Vehicles will also reduce commute time by 40% [29]. Further, according to a report by US Energy Information Administration (EIA), reduction in traffic accidents will also reduce traffic congestion as these traffic incidents are the reason behind 25% of congestion [53].

Therefore, with the emerging technology of autonomous vehicles lives and environment both can be saved. This research focuses on effective traversal of autonomous vehicles. Through detection and classification of objects and integrating Reinforcement Learning, autonomous navigation of self-driven cars is ensured.

1.3 Research Objectives

The aim of this research is to establish traversal of an autonomous vehicle in undiscovered tracks through object detection. It uses the combination of Reinforcement Learning and Faster R-CNN. Reinforcement Learning is applied for traversal along with Faster R-CNN intended for object classification and detection. The research objectives are as follows:

- To initiate application of Reinforcement Learning for autonomous vehicle navigation.
- To develop Faster R-CNN with the aim of object classification and detection.
- To deeply understand Reinforcement Learning Model, Double Deep Q-Learning Network (DDQN) and Markov Decision Process (MDP).
- To develop a model for autonomous traversal of vehicles by integrating Reinforcement Learning and Faster R-CNN.
- To evaluate the outcome and accuracy of the model.
- To provide recommendations on improving the model.

1.4 Scope and Limitation

The object classifier through Faster R-CNN has scope to improve its performance. The execution can be improved by lowering the average number of misclassification and by increasing the average number of accuracy level [33]. Moreover, training with larger number of datasets can help this research attain better accuracy in future. Confusion matrix will indicate the percentage of accuracy that has been acquired through these processes. Hence, the iteration score can be increased by training object classes to overcome the misclassification problem. This research has been only used for autonomous cars for now. Further, it can be implemented in other autonomous agents.

The research model does not take rear view camera and condition into consideration which is a limitation to this research. It also struggles to take decisions while there are other vehicles trying to overtake. Further, implementation of this model in Bangladesh road perspective is difficult as of now.

1.5 Document Outline

The structure of rest of the paper is as follows. Literature review and related works that have already been recognized are described in Section 2. Reinforcement Learning, Adam Optimizer, Xavier Initializer, Challenges of Reinforcement Learning and Regional Convolution Neural Network (R-CNN) have been described in different subsections in this part. In Section 3, methodology of this research has been discussed which includes the procedure of making the object classifier through Faster R-CNN and input distributional agent for autonomous driving through Double Deep

Q Network (DDQN). Section 4 describes the implementation and result analysis part where different procedures has been discussed for implementing this research in GTA V game environment along with various testing results, accuracy of reinforcement learning and Faster R-CNN in the autonomous vehicle and comparison between different algorithms of Deep Reinforcement Learning. Section 5 concludes the paper which summarizes the whole research in some subsections along with contribution and future plan.

Chapter 2

Literature Review and Related Work

This chapter talks about Reinforcement Learning model and its elements. It also includes the comparison between supervised learning and reinforcement learning. It presents Deep Q- Learning network using which we have introduced an autonomous vehicle system. Another algorithm which we have integrated here for image classification is described in this chapter. So, this chapter also describes about Faster R-CNN and shows how Faster R-CNN is much effective than R-CNN and Fast R-CNN.

2.1 Reinforcement Learning

2.1.1 Reinforcement Learning Definition

Reinforcement Learning is a machine learning branch which works by gaining experiences through communicating with the worldly environment and evaluating feedback to develop a system's performance to make behavioral decisions [30]. It improves the system's performance through trial and error experience with dynamic environment. Qualitative and quantitative frameworks to understand and adapt decision-making are provided by reinforcement learning through rewards and punishment [12].

The roots of Reinforcement Learning are in psychology, but while getting into the details of it, the differences can be seen [7]. In psychology, it refers to occurrence of an event in relation to a response which increases the probability of the response occurring again in that situation [2]. On the other hand, in engineering and artificial intelligence, Reinforcement Learning refers to learning tasks and algorithms based on the principle of reinforcement [7]. Hence, reinforcement learning involves learning to take decisions, mapping situations to actions and maximizing reward signals [16].

2.1.2 Elements of Reinforcement Learning

Some common terms that are used in Reinforcement Learning [47] such as:

- **Agent:** The entity that executes actions in an environment to receive reward.
- **Environment (e):** The surrounding that an agent encounters.

- **Reward (R):** The return that an agent receives when it executes a particular action.
- **State (s):** The present condition that is represented by the environment.
- **Policy (π):** The approach which is applied by the agent for deciding the following action depending on the present state.
- **Value (V):** The long-term return with discount in comparison to the short-term reward.
- **Value Function:** The total amount of reward which is the state value.
- **Model of the environment:** The behavior of the environment.
- **Model based methods:** The method for resolving Reinforcement Learning problems that applies model-based ways.
- **Q value or action value (Q):** Q value is almost similar to value. The sole difference between the two is that it takes another variable as a present action.

2.1.3 Reinforcement Learning versus Supervised Learning

The following Table 2.1 [47] shows the differences between Reinforcement Learning and Supervised Learning.

Parameters	Reinforcement Learning	Supervised Learning
Decision Making	Sequential decision making	Decision making based on input given initially
Working Policy	Works by communicating with the environment	Works from previous examples or sample data
Dependency	Dependent learning decision. Hence, labels are provided in all of those decisions	Independent decisions. Hence, labels are provided for individual decision
Best Supported	Best supported in AI where human communication is usual	Mostly supported in an interactive software system or application

Table 2.1: Reinforcement Learning vs Supervised Learning

2.1.4 Exploitation and Exploration

As Reinforcement Learning refers to making an agent learn its way, finding an optimal path is thus, important. An agent learns to take decisions by receiving rewards for its actions [11]. For learning to take optimal decisions, an agent must explore the same environment many times. Hence, a balance of exploitation and explorations is needed to get the agent learn to find better goals every time [11]. According to Coggan [11], exploitation is what the agent already knows about the

worldly environment and what it knows of as the best results. On the other hand, exploration is to discover new conditions and features of the world and finding better goal path than what the agent knows of already. Better goal will not be found without exploring and the agent will choose the first goal always which will not be optimal decision making. However, Coggan [11] further mentions, an agent will not stick to a particular path if it explores much and its knowledge will not be exploited for that. Hence, a balancing between exploration and exploitation is a must for effective decision-making.

2.1.5 Reinforcement Learning Model

Usually, an agent in a Reinforcement Learning model communicates with the environment through perception and action [6]. It inputs indication from the environment and agent takes actions based on decisions which is generated as output. Actions, which has been taken by the agent can alter the condition of the environment. An agent communicates with values of this state transition with a scalar reinforcement signal. Hence, the model is composed of:

- A distinct set of environment sets
- A distinct set of agent actions
- A set of scalar reinforcement signals which are usually real numbers or 0,1

Some important learning models in reinforcement learning are:

- Q-Learning
- Markov Decision Process (MDP)
- Deep Q-Learning Network (DQN)
- Double Deep Q-Learning Network (DDQN)

Q-Learning

One of the Reinforcement Learning algorithms is Q-Learning that thrives to look for the finest action to decide in a provided present situation [16]. Being an off policy Reinforcement Learning algorithm, a Q-learning function trains itself from actions that are not inside the current policy. Q-learning also learns the policy that can maximize the whole reward. How a given action is useful can be represented by the quality for this case in gaining future reward.

An agent updates Q-values in a Q table by interacting with the environment. The first one is by exploiting where the environment information is known to the agent to make decision. The second one is taking actions randomly which is called exploring. Random actions generate Q-values from which the agent chooses actions based on optimal Q-values. The key entities in Q-learning are – environment, action, reward and state.

The basic steps for Q-learning are:

- Agent taking a decision or action in a state and receiving a reward
- Agent selecting which action to take by matching Q-table with maximum value or by random (epsilon, ϵ)
- Updating Q-value

Markov Decision Process

Markov property is considered as a memory-less property of a stochastic procedure. If probability distribution of future states is dependent on present state and conditioned on both past and present state, it can be said that it has Markov property [16]. Markov Decision Process, or MDP is the Reinforcement Learning action which supports the Markov property. To understand MDP, Markov reward process has to be looked at. Markov reward process accumulates the reward through some particular sequence. Further, Markov decision process is nothing but Markov reward process along with decisions.

Deep Q-Learning Network

A neural network is used to estimate the Q-value function in Deep Q-Learning [42]. Input is state and output is the produced Q-value of all feasible actions. In Deep Q-Learning Network (DQN), all the previous experiences are gathered by the agent in memory. The following decision can be decided with the highest return of the Q-network. With DQN, an agent has a model to make predictions from instead of looking into Q table. Rather updating in the Q table, the model can be trained. It is a regression model which will output values for each possible actions which then can be called Q-values for this algorithm.

Double Deep Q-Learning Network

The problem in Q-Learning is, it requires estimation of learning from estimates. This overestimation may be troublesome as electing such highest overestimated values means electing the estimate of the highest value.

To solve this problem, Double Q Learning is initiated for solving the problem of overestimation of Q-value in basic Q-Learning [35]. The solution requires making use of two different Q-value estimator, each is utilized to upgrace the other. Unbiased Q-values of actions can be estimated using the opposite estimator as the estimators are independent [41]. Thus, maximization bias can be avoided. Double Q-Learning needs two different action-value function, Q along with Q', as estimators. If Q and Q' are noisy, the noises can be considered as uniform distribution [14].

The procedure differs from the basic Q-learning as following:

- Q function is used for electing the best measure with highest Q-value from the following states
- Q' function calculates expected Q-value by using the action selected before.
- Q function is then updated by using the expected Q-value of Q' function

Further, Double Deep Q-Learning Network [35], was inspired by double Q-Learning which uses two kinds of Deep Neural Networks. They are Deep Q Network and Target Network. The optimization stage of updating the parameters of Deep Q Network are mentioned below:

- **Deep Q Network:** selects the best action with highest Q-value of following state.
- **Target Network:** calculates the estimated Q-value with action mentioned above.
- Updates the Q-value of Deep Q Network depending on the estimated Q-value from Target Network
- Updates the variables of Target Network based on the variables of Deep Q Network per several iterations.
- Update the variables of Deep Q Network based on Adam Optimizer

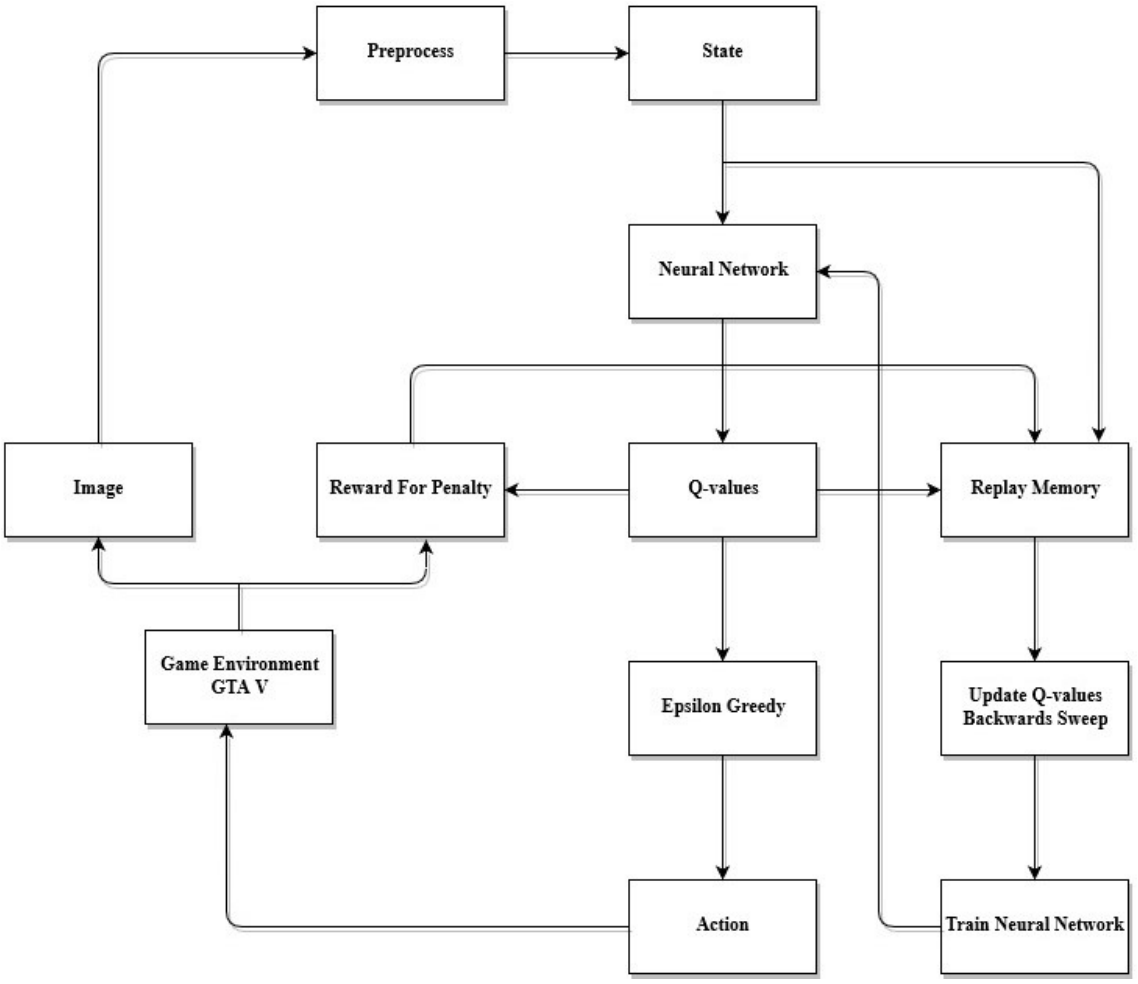

Figure 2.1: Flowchart of Proposed Reinforcement Learning Based Autonomous Vehicle Model

2.2 Adam Optimizer

Adam [23] is an adaptive algorithm which optimizes learning rate and it has been designed in a way that it can train deep neural networks. It may be considered as a amalgamation of RMSprop and Stochastic Gradient Descent with momentum. Squared gradients are used by it so that it can scale the learning rate like RMSprop. Further, it takes control of momentum by making use of moving average of the gradient as an alternative of gradient itself. Adam computes individual learning rates for diverse scenarios and utilizes approximation of the first two moments of gradient to adjust the rate of learning for individual weight of the neural network.

2.3 Xavier Initializer

Xavier initializer automatically depicts initialization scale. This depends on the neuron numbers of input and output. If initialization is done wrong, it can lead to exploding and vanishing of weights and gradients. The weights of the model might explode to infinity or vanish to 0 which make training deep neural networks very complicated. Hence, Xavier works with neural networks with five hidden layers. Xavier initialization usage ensures that the weights are not too small or big to propagate the signals precisely. Initialization of the weights from a distribution with zero mean and variance [13]:

$$\mathrm{Var}(W) = \frac{2}{n_{in} + n_{out}} \tag{2.1}$$

where n_{in} and n_{out} are respectively the number of inputs and outputs of your layer.

2.4 Challenges of Reinforcement Learning

Some challenges of Reinforcement Learning include:

- Parameters affecting the speed of learning
- Non-stationary real environments
- Weakening of results may occur due to excessive reinforcement which causes overburdening of states
- Large action space may increase complexity
- Reward design has to be involved

2.5 Region Convolution Neural Network (RCNN)

2.5.1 Convolutional Neural Network(CNN)

Object detection method involves a model or any distinct algorithm to execute the identification procedure systematically. Convolutional Neural Network (CNN) is a model constructed to identify and to classify any images data [22]. Any images of

particular regions are passed through the convolutional layers and show the ultimate detection result. This model includes several layers named The Kernel, Pooling and the Fully Connected Layer (FC Layer) through which many images or regions are trained by feeding them into the network[22].

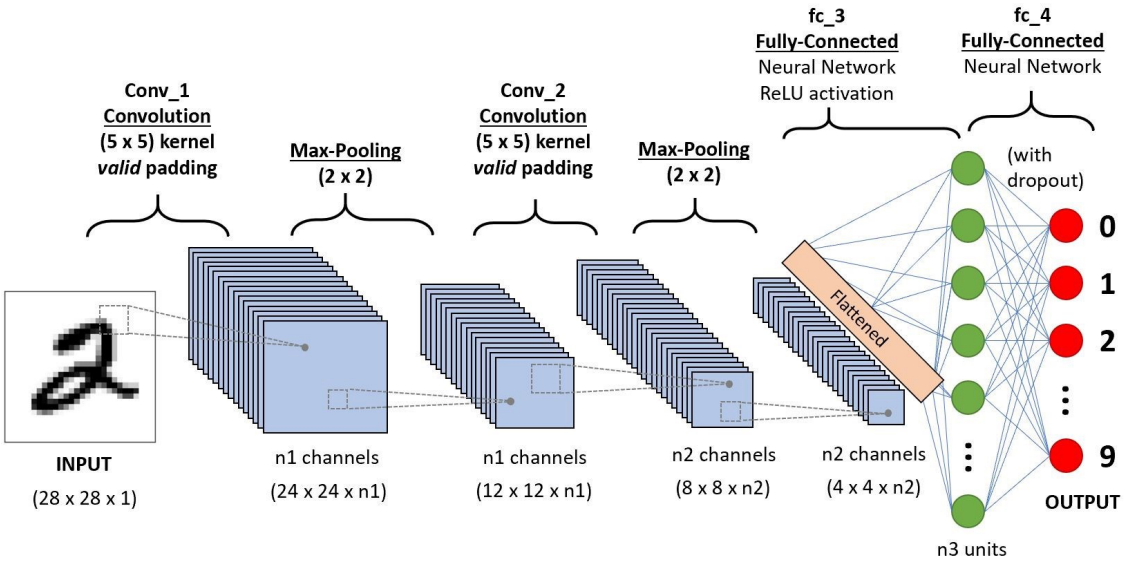

Figure 2.2: Convolutional Layers [48]

Here Figure 2.4 [48] represents the layer of convolutional networks. Any images with particular bounding boxes are passed through these layers. The first layer is the Kernel which represents the image in a matrix of 5 x 5 x 1 [48]. Then using the second layer which is Valid Padding increases the dimensionality of image. Then Pooling Layer reduces the size of the evaluated Feature. Fully-Connected layer then presents last output of the desired image [48].

2.5.2 R-CNN

Drawing bounding boxes based on the image location is the key procedure of this CNN model [18]. However, there are several ways of identifying bounding boxes. The OverFeat method is one of the popular ways of predicting boxes. As, it handles single object class thus a convolutional layer is used to identify multiple objects [18]. Multi class object requires mulitiple bounding boxes to identify the objects simultaneously. Therefore, the selction of regions can involve computational complexity [21]. So, to avoid this complexity a new model has been proposed by Ross Girshick which is called Region Convolution Neural Network (RCNN). The main powerful tool of R-CNN is its region proposal module. This method works with a fixed number of regions which is exactly 200 in numbers and those particular regions are derived using another different algorithm popularly known as selection search. Then these regions are fed into the convolutional layers and then retrieving the features of the interested regions are again fed into the Support Vector Machine (SVM) [24]. Therefore, following these steps any identifiable objects are detected using this model.

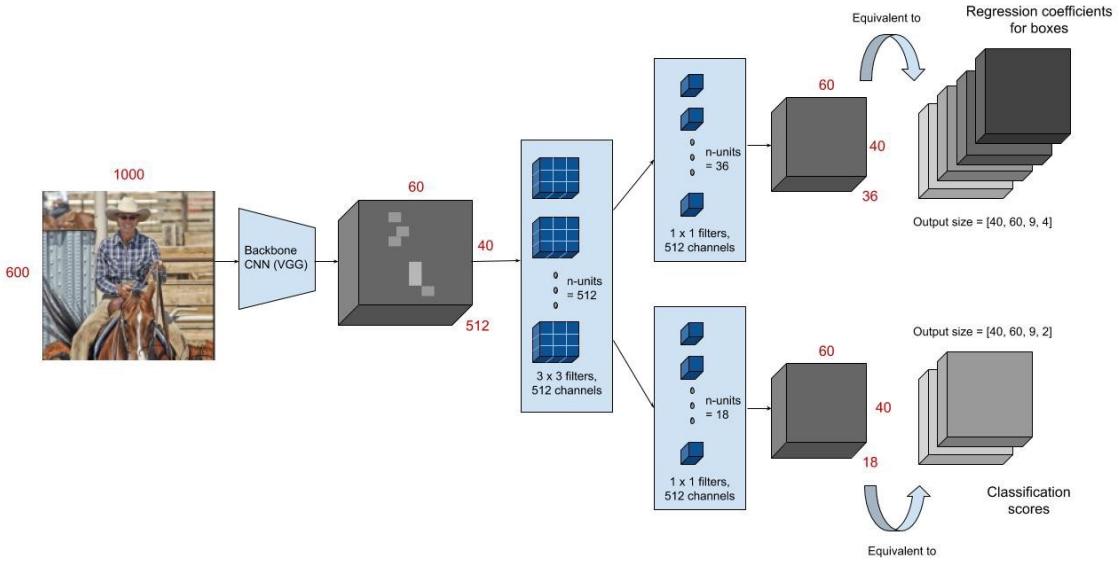

Figure 2.3: Architecture of region proposal network [49]

This Figure 2.5 [49] represents the region proposal network model which starts with input image being fed into the traditional convolutional neural network. Firstly the image is resized such that its shortest side is 600px with the longer side is 1000px [49]. Then from the output feature map network learns the location of image. During the learning procedure a 3 x 3 convolution is applied to the backbone feature map. Then this is trained through convolution layer and thus output image is retrieved [49].

There are many drawbacks associated with R-CNN [28]. It involves selective search algorithm which does not include consecutive training and learning that makes the region proposal system not adaptive with the situation. Another problem is takes a long to train the network for classifying the regions associated with the interested objects or images[24].

2.5.3 Fast R-CNN

Identifying the drawbacks of R-CNN Ross Girshick again improved this model which is popularly known as Fast R-CNN [27]. Previously, in R-CNN regions which are extracted using selective search are fed into the convolutional layers. It includes the feeding of 200 regions per image which consumes a lot of time. So, here in Fast R-CNN only images are feed into the convolutional layers rather than the regions of the interested objects. It starts with feeding the images into the CNN.

Then the identified regions of proposal are shaped using RoI pooling layer. This shaping procedure enables the efficient region-based object detection. Then the class of the proposed regions are predicted using the softmax layer [25]. Then from the offset values features of the objects are extracted. Thus it completes the object identification process. Fast R-CNN takes less time than R-CNN due to the elimination of steps which feed proposed 200 regions of each image to the network. Therefore, this model is comparatively in better position for its accuracy and faster processing capability [25].

2.5.4 Faster R-CNN

As both the algorithms do not work efficiently due to the time complexity problem Faster R-CNN was introduced by Shaoqing Ren who modified the features of previous models and algorithms. Faster R-CNN also includes convolutional network which provides a feature map [27]. In the previous algorithms the model used selective search for identifying the interested regions but here it does not use this algorithm as it consumes huge time to identify the regions associated with the interested images. Therefore, it involves another different network to predict the proposal of the regions. Then those regions identified though the networks are processed and shaped using RoI pooling layer [17]. Then it follows the procedure of Fast R-CNN to identify the object. Basically, Faster R-CNN is the product of two features. One is deep fully convolutional network using which the interested and proposed regions can be identified and then comes the Fast R-CNN detector [27]. This detector completes the steps and features that Fast R-CNN model uses.

2.6 Related Works

In recent years, neural networks have turned into the main technique for reliable object identification. In [40] most neural networks images are classified, clustered by their similarity using R-CNN, Fast R-CNN, Faster –RCNN algorithm. Thus, in object detection these algorithms are explicitly used.

CNN [34] is the abbreviation for Convolutional Neural Network. It consists of convolution, RELU, Pooling and Fully connected layer. Initial approach towards object detection is classifying some interested regions and use CNN to it. The CNN [21] in RCNN focuses on a single region at a time to minimize the interface. Here, regions are specified. So, it is also called region proposal.

However, to make object detection more effective and fast Faster R-CNN [34] is used. Other algorithms use selective search for regions but a separate network is used in Faster R-CNN to predict region proposals. By reshaping the proposed regions, the value of bounding boxes is predicted.

A work uses [39] region-based convolutional neural network for the detection of road obstacles using deep learning system. However, our proposal does not only detect obstacles, it takes decisions upon them using double deep q-learning.

Additionally, psychology includes study of learning and reinforcement which has had a well-built impact on Artificial Intelligence related work. In fact, all algorithms of reinforcement learning can be considered as reverse engineering of some psychological learning processes [5].

Although, work related to reinforcement learning has been done in the field of web-spidering for optimal sequential decision making [8], our proposal uses reinforcement learning for efficient path finding.

Another paper [4] suggests eight extensions of reinforcement learning which includes adaptive heuristic critic (AHC) learning, Q-learning, and three further extensions to both basic methods to speed up learning. This proposal mainly focuses on DDQN based learning to train the agent.

One related work [44] determines driving policy on highways using reinforcement learning. A different driving simulator from the mentioned work has been used in our implementation which is discussed later in this paper.

Longitudinal control of autonomous land vehicles has been proposed [37] by using parameterized reinforcement learning in another related work. It mainly uses PBAC algorithm which differs from the DDQN algorithm that we have used.

Thus, Reinforcement learning is said to be an interesting learning technique which only requires a performance feedback from the environment. Till date, it has been used to solve simple learning problems. Our proposal investigates reinforcement learning to be used as a decision maker for path finding.

Hence, we propose to implement a reinforcement learning based autonomous vehicle which traverse through exploration and exploitation of environments. It finds its own path by detecting and identifying objects and obstacles along the way using Faster RCNN. combination.

Chapter 3

Methodology

This chapter describes how Double Deep Q Learning (DDQN) and Faster R-CNN have been implemented in the Reinforcement Learning based Autonomous Vehicle for Exploration and Exploitation of Undiscovered Track. It includes the description of data and also describes how both the DDQN and Faster R-CNN classify and cluster any data using to develop this model. Moreover, it describes the evaluation procedure and the process of measuring accuracy in autonomous traversing.

3.1 Object Classification Through Faster R-CNN

3.1.1 Faster R-CNN

The Regional Convolutional Neural Network technique adopts the clear strategy of trimming remotely calculated box proposition out of an input image and applying neural system classifier on it. Nonetheless, this methodology can be costly because many crops are required, which leads to large overlap calculation from overlapping crops. Fast R-CNN moderated this issue by driving the entire picture through feature extractor. Crop from a middle layer allow crops to share the load of highlight extraction [27]. Although R-CNN and Fast R-CNN depends on an external proposal generator, lately it has been proven that generating box proposals using neural nets is possible. Here, it is normal that there can be few boxes on beat of each other on the picture at distinctive outline, scales and perspective proportions, which is called "anchors". Now, a model is prepared to anticipate for each anchor: (a) a discrete class expectation for each anchor, and (b) a cumulative forecast of the counterbalance, according to which the anchor has to be relocated to fit in the ground truth bounding box. In the accompanying para, minimization of a combined classification and regression loss is discussed [24], [32], [36].

The best matching ground-truth box b is first to be found for each anchor a. If we can identify a match, we consider it as a "positive anchor", and call that (a) which is a class label $y_a \epsilon \{1 \ldots k\}$. Furthermore, (b) which is a vector that will encode b with respect to anchor a (known as box encoding $\Phi(b_a;a)$). On the off chance that no such match could be found, we consider it as a "negative anchor", and set the class name to be $y_a = 0$. Considering the anchor as a, if box encoding is predicted f_{loc} (I; a; θ) and following class f_{cls} (I; a; θ), where the image will be known as I and the model parameters will be θ, then the loss for I is measured as a weighted

sum of a location-based loss and a classification loss:

$$L(a; I; \theta) = \alpha \times 1[aispositive] \times l_{loc}(\Phi(b_a; a) - f_{loc}(I; a; \theta)) + \beta \times l_{cls}(y_a, f_{cls}(I; a; \theta)) \tag{3.1}$$

Here α,β are offset values which will balance losses of localization and classification. Equation 3.1 is averaged over anchors, in order to train the model and reduced with respect to parameters θ. [36]

The determination of anchors has noteworthy outcomes for exactness and calculation as well. Previously these anchors were computed from clustering ground-truth boxes within the data-set. Nowadays, the process is handled by tiling a collection of boxes at various scales and aspect ratios, routinely over the image. The good side of a customary network of stays is that the forecasts can be composed as tiled indicators on the picture with shared parameters. This process resembles traditional sliding window method [24], [32].

3.1.2 Acquiring Data-set

Experiments were conducted on the huge data-set named Open Image V5 [43]. It's an open-source database made by Google. It consists of annotated images of 600 box-able object classes. The total number of training images in this database are 1,743,042. These images include annotated bounding boxes, object segmentation, and visual relationships, as well as the full validation (41,620 images) and test (125,436 images) sets. But for our Object Classifier we don't need all those 600 classes.

Using '$OIDv4\_ToolKit$' 7 classes from 600 were separated. Those are of Bicycle, Bus, Person, Motorcycle, Truck, Van, Car. For training criteria, 8,355 images were separated from the whole data-set, where minimum of 1000 images were of each class. And 2,360 of them were taken for testing, having a minimum of 300 images of each class. The labels of those images were then combined together in .XML format.

3.1.3 Training Object Classifier

For training our data-set and to prepare our object classifier we chose Faster R-CNN Inception V2 feature extractor on TensorFlow framework. For training and testing, a computer having NVIDIA GTX 1050 GPU, with CUDA core support has been used. The training process was consistently run for 13hours, until the total loss became persistent under 0.8 for a long time. A total of 1,58,777 steps were conducted to prepare our desired image classifier.

There are two stages is the Faster R-CNN detection process. The primary stage is called region proposal network (RPN). Here images are processed by a feature extractor, for which we have used Faster R-CNN Inception V2 feature extractor, and features are used to anticipate class diagnostic box proposals, at some selected intermediate level. The loss function for this first stage appears as Equation of Loss

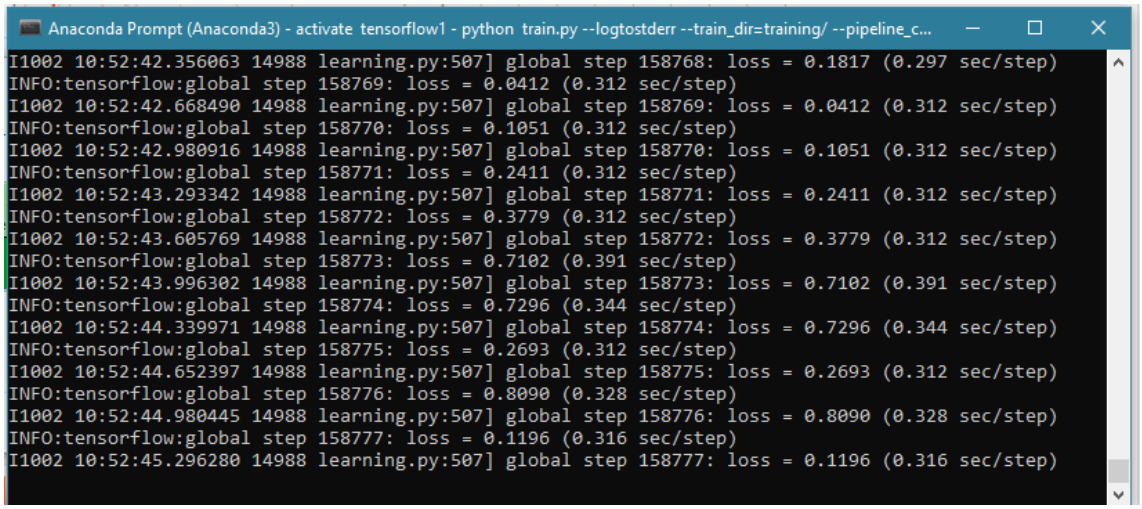

Figure 3.1: Training data model object classification

Figure 3.2: Loss Graphs of training data model: Anchor Loss - Classification

Figure 3.3: Loss Graphs of training data model: Anchor Loss - Localization

Figure 3.4: Loss Graphs of training data model: RPN Loss – Localization

Figure 3.5: Loss Graphs of training data model: RPN Loss – Localization

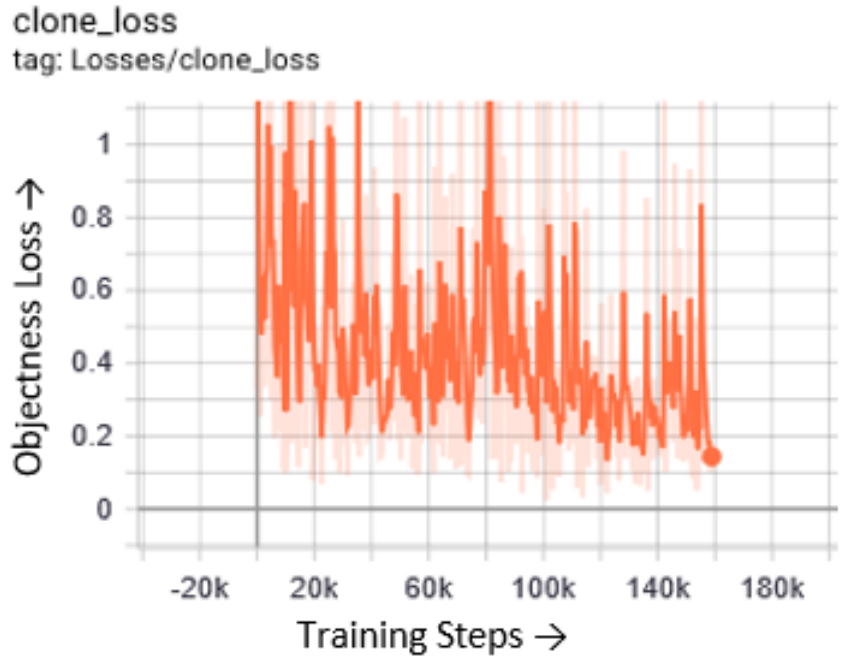

Figure 3.6: Loss Graphs of training data model: Clone loss

Figure 3.7: Total loss graph of the trained data model

using a grid of anchors tiled in space, scale and aspect ratio.

In the subsequent stage, the box proposals which were predicted in the first stage, are utilized at cropping features from the same common feature map which are subsequently used to the rest of the feature extractor so as to predict a class and class-specific box refinement for each proposal. The loss function for this second stage box classifier likewise appears as the Equation of Loss using the proposals generated from the RPN as anchors. Here, Figure 3.7 shows the normalized form of total loss graph.

3.2 Distributional Agent for Autonomous Driving

3.2.1 Double Deep Q Network (DDQN)

Double Deep Q Network (DDQN) algorithm was proposed by H. V. Hasselt [35]. This calculation utilizes the concept of Double Q-learning and is an extension of H. V. Hasselt's previous proposal [14] and it is applied to DQN. These Q-learning based algorithms have overestimation problems which are caused by estimation errors. Overoptimistic fee estimation and performance dilapidation happen as a result of overestimation. However, the strategy for DDQN does not just diminish the overoptimistic esteem estimation, yet additionally gives better execution than DQN on a few game conditions. Selection and evaluation process are isolated by DDQN while it gets to target an incentive with two Q-functions. The required conditions of DQN and DDQN are appeared underneath:

$$y^{DQN} = R_t + \gamma \max_{A_{t+1}} Q\left(S_{t+1}, A_{t+1}; \theta^{-}\right) \tag{3.2}$$

$$y^{DDQN} = R_t + \gamma Q\left(S_{t+1}, \operatorname*{argmax}_{A_{t+1}} Q\left(S_{t+1}, A_{t+1}; \theta\right); \theta^{-}\right) \tag{3.3}$$

3.2.2 Markov Decision Process for Path Distribution

Markov Decision Process articulates the acquiring path direction for autonomous driving in this research. The agent decides his own action in every step and immediately a reward is received for that action. Markov Decision Process is narrated by the tuple S, P, A, R, γ which has already been stated before. For our problem, a brief summary of MDP is stated below:

- s ϵ S is the finite state space which contains a gray scale image from camera of the agent.
- P is the transition function where P $(s' \| s\ a\ )$: S $\times$ A $\times$ S $\rightarrow [0, 1]$
- a ϵ A is finite action space which works for an agent.
- R(s, a) : S $\times$ A $\rightarrow$ R where R is reward function
- γ defines the discount factor where $\gamma \rightarrow$ [0,1] for delayed reward

MDP states' s ϵ S can be used for the high dimensional observations by using deep neural networks. Figure 3.8 represents the perception of the surrounding coverage by using 3 cameras mounted at the front.

The autonomous driving agent has 5 distinct actions. The finite action space A consists of forward, left, right, stop and deceleration. 5 kph is added or subtracted from the current agent speed for forward and deceleration. The agent speed is confined in the range of 30 kph to 80 kph. The agent automatically adjusts the speed for vehicles in a certain distance so that it maintains a safe distance from the front vehicle. The 'stop' activity happens instantly when the vehicle in front all of a sudden brakes or any other vehicle cuts in abruptly before our agent vehicle.

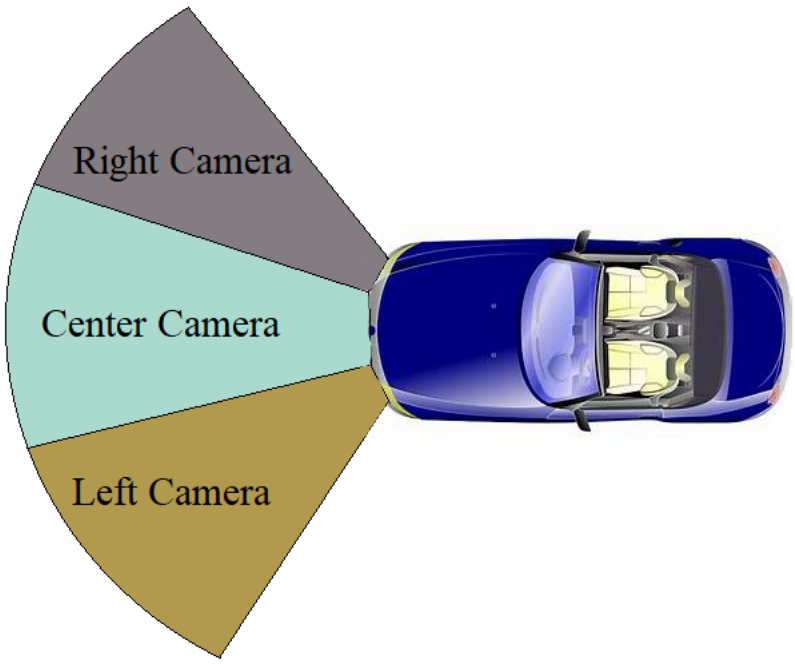

Figure 3.8: Camera Coverage of Perception.

3.2.3 Data Preprocessing

The images are cropped so that the model will not be trained with the sky and the car's front parts. Those are resized to 160x320 (3 YUV channels) as per NVIDIA model. Those images are normalized (image data divided by 127.5 and subtracted 1.0). As stated in the Model Architecture section, this is to avoid saturation and make gradients work better.

3.2.4 Model Architecture Design

The main target of the agent, π(a]s) is mapping the perception state, S and making the following move on action space, A. The entirety of the activity will be led in a stochastic driving condition. However, to accomplish this mapping the model needs to satisfy to distinct conditions: (1) extract and capture significant highlights from 3 camera images,(2) it should take account of the inherent randomness of the environment for choosing any particular action.

Here, to satisfy the first condition; spatio-rational information retrieving from camera sensor need to be sensed by the network. This process is conducted using Convolutional Neural Network (CNN). It is popular for extracting spatial features from images. Moreover, high dimensional camera images are refined into visual feature vector using three two-dimensional convolutional layers.

Furthermore, the second condition can be fulfilled by using DDQN framework. Driving environments that are stochastic use this framework. For each action there is a return distribution which is created by the completely associated layer with the help of θ. Here, the Q(s,a) result can be estimated as the desire of quantiles,$\sum_i q_i \theta_i(s,a)$.

Additionally, the maximum Q value can be retrieved from the best action, a^* which also can be picked from accessible limited Q values of action space, A.

$$a^* = \text{argmax}_a \, Q\left(s, \text{argmax}_a \, Q(s,a)\right) \tag{3.4}$$

Proposed DDQN architecture for Reinforcement Learning based Autonomous vehicle is shown in Figure 3.9. Keras has been used to train this network.

3.2.5 Hyperparameters

The network is designed following NVIDIA model. To perform end-to-end self-driving test by NVIDIA, it has been used. Supervised image classification or regression problems can be solved in seamless procedure using deep convolutional network. NVIDIA model itself is well documented. Thus, our main focus lies on adjusting the training images for delivering the best result. However, to acquire the best result we need to make necessary adjustments for avoiding over-fitting nature and adding non linearity to make the prediction accurate. Additionally, we have added the following adjustment to the model.

- Lambda layer is introduced to normalize input images to make gradients work more smoothly and to avoid saturation.

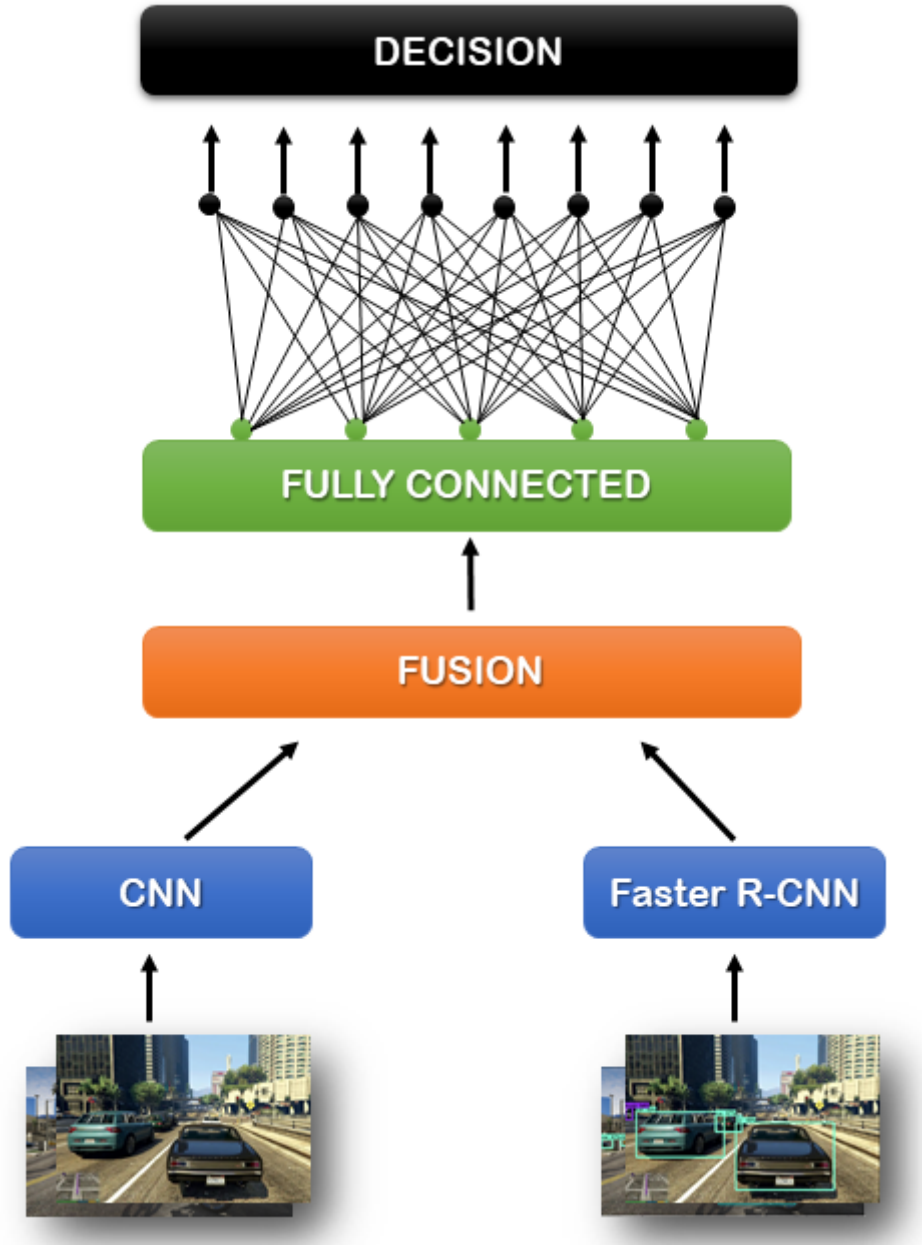

Figure 3.9: Proposed DDQN architecture for Reinforcement Learning based Autonomous vehicle

- Additional dropout layer is added after the convolution layers to avoid the over-fitting nature.
- Then we have implemented ReLU for activation function to ensure linearity.

Adam Optimizer with epsilon 0.0001 at a learning rate $1e^{-5}$ and mini-batches of size 32 is used to train the network for optimum accuracy. Here, Xavier initializer has been used to initialize network weights and all inputs are normalized into [-1,1]. For achieving the accuracy of the prediction of steering angle for each image we have used mean squared errors for estimating the loss function. The table 3.1 shows Hyperparameters of this driving policy network. We have set the value of support Q as 200. The size of the replay memory is 5000000 and γ which is the discount factor has been fixed to 0.99. ϵ-greedy policy has been used where was gradually diminished from 1.0 to 0.1 in each step and then fixed to 0.1. All of these policies have been implemented during 3000000 steps trainings.

Data	**Layer Type**	**Actuation**	**Hyperparameters of Policy Network**
Camera Data	Convolution 2D	ReLU	Patch size = (5×5) Stride = 4 No. of filters = 24 Patch size = (5×5) Stride = 4 No. of filters = 36 Patch size = (5×5) Stride = 4 No. of filters = 48 Patch size = (3×3) Stride = 1 No. of filters = 64 Patch size = (3×3) Stride = 1 No. of filters = 64
Concatenated Data	Fully Connected Layer	ReLU	No. of Units = 512

Table 3.1: Autonomous Driving Policy Network:Hyperparameters

3.2.6 Model Training

For training, we used the following augmentation technique along with Python generator to generate an unlimited number of images:

- Randomly choose right, left or center images.
- For left image, steering angle is adjusted by +0.2
- For right image, steering angle is adjusted by -0.2
- Randomly flip image left/right
- Randomly translate image horizontally with steering angle adjustment (0.002 per pixel shift)
- Randomly translate image vertically
- Randomly added shadows
- Randomly altering image brightness (lighter or darker)

Using the left/right images is useful to train the recovery driving scenario. The horizontal translation is useful for difficult curve handling.

Chapter 4

Implementation and Result Analysis

This chapter describes the implementation of the proposed model for Exploration and Exploitation of Undiscovered Track. The Object Detection part of the model was implemented and tested using MS COCO dataset. On the other hand the Reward Determination part of the model was implemented and tested using GTA V game environment [52]. This model follows three stages; input data pre-processing, combined decision making and finally the result output. Data preprocessing has again two parts: Object Detection based of Faster R-CNN and Reward Determination for the Reinforcement Learning algorithm.

4.1 Data Preprocessing

4.1.1 Object Detection

The translated image is taken from the three physical cameras of the autonomous vehicle, and run through Faster R-CNN architecture for object detection. Faster R-CNN uses Fast R-CNN as detector, which consists of Convolutional Neural Network backbone, Region of Interest pooling layer and fully connected layers for classification and bounding box regression. Firstly, a feature map for the image is generated through backbone CNN. The bounding box proposals of the Region Proposal Networks are then used to pool features from the feature map. ROI pooling is very advantageous to be used here. The features are fed into the sibling classification and regression branches.The features are passed through a softmax layer to get the classification scores. Basically these scores determine the belonging class of each box. The regression layer coefficients are used to improve the predicted bounding boxes. The Figures below show the output of object detection which was implemented on sample frames from a game environment.

4.1.2 Reward Determination

Double Deep Q Learning is working based on reward value which comes from the reward function of this algorithm. The main part of this reward function is epsilon

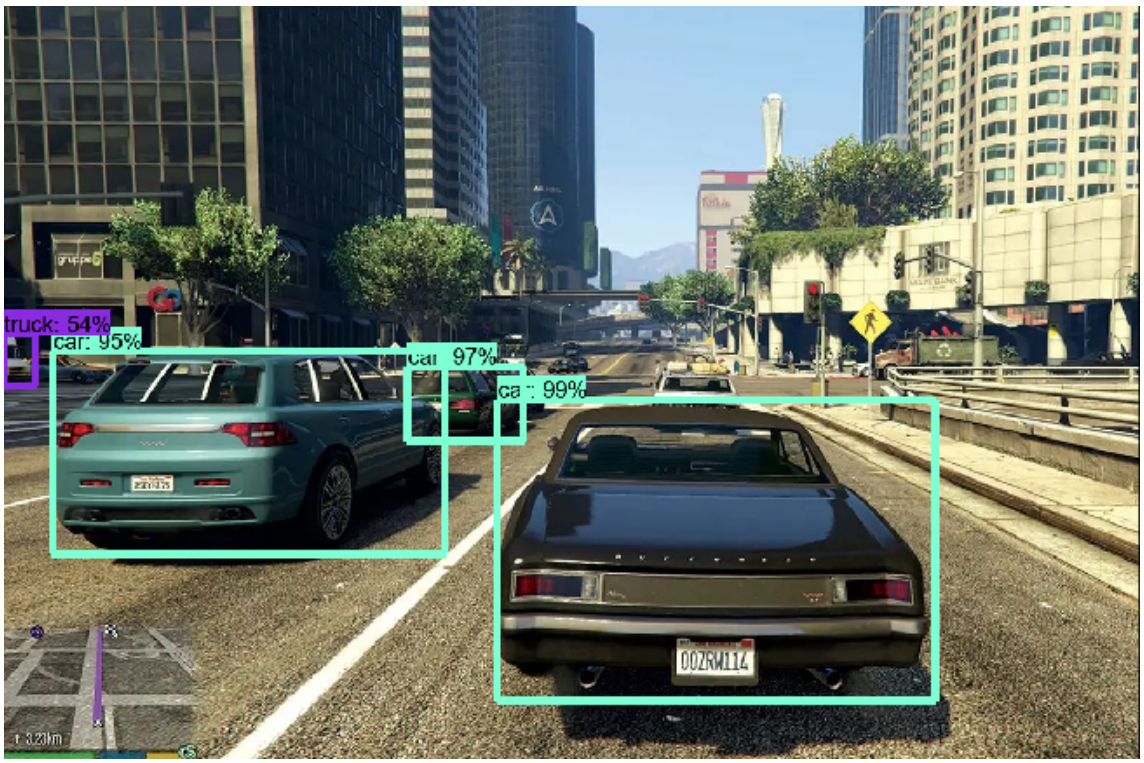

Figure 4.1: Image Classification through Faster R-CNN: Car [52]

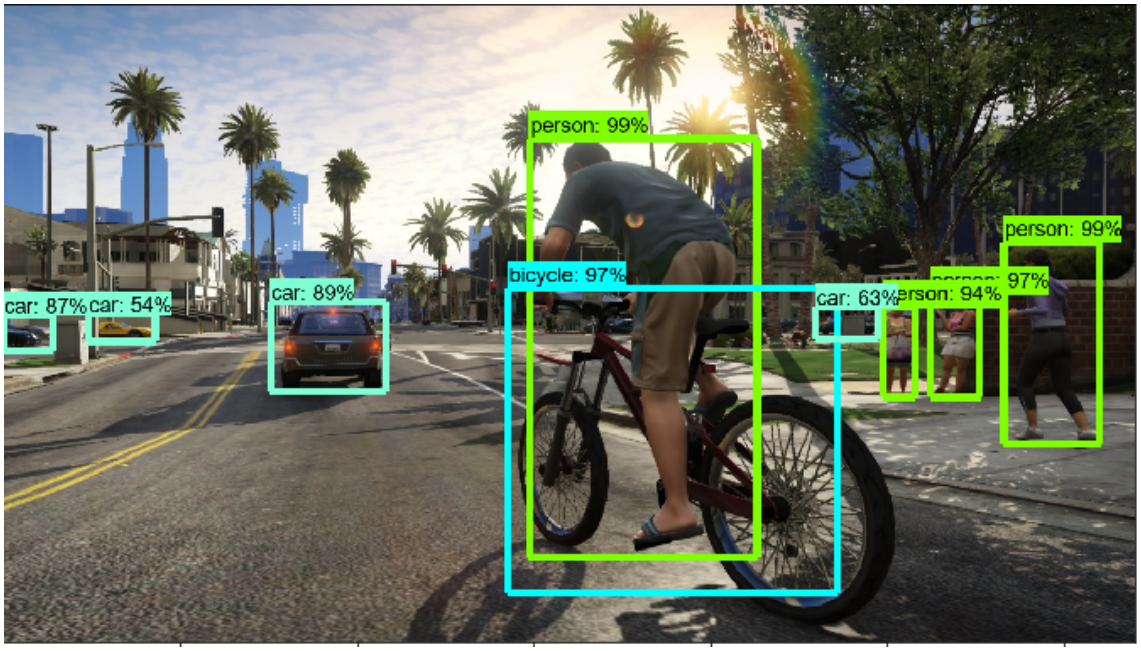

Figure 4.2: Image Classification through Faster R-CNN: Object [52]

Figure 4.3: Image Classification through Faster R-CNN: Pedestrian [52]

value determination. Those epsilon values comes from data preprocessing part. In the training period we are generating a drivinglog.csv file from input data. Figure 4.4 shows the driving log file. In this file first 3 columns are defining the cropped image files of 3 cameras which has been already described in the methodology part. Following 3 columns shows the steering angle from different images. After that the last columns generates the epsilon value for each case which helps to construct the reward values.

4.2 Combined Decision Making

This traversing algorithm involves reward method which is considered as an inevitable part of Deep Q Learning to make this decision-making process more accurate. Moreover, we are delivering the reward to our agent based on the Q values of every training data. Following figures show the characteristics of DDQN hyperparameters by which Figure 4.5 defines the reward values, Figure 4.6 defines the value loss which comes from the value loss function of double deep q learning and the total loss function from the values of value loss function shows in Figure 4.7. Following figures are generated from the output section of the algorithm that we have implemented for the uninterrupted traversing of our agent which shows the average reward of the training dataset.

The following graphs are the representation of training data collected from GTA V game environment. The agent has been tested in every possible environment which is similar to real life scenario (e.g. day, night, rainy, foggy, crowded etc.). Figure 4.8 represents the graph of raw training data. On the contrary, Figure 4.9 represents the graph of 160000 normalized training data.

Center Image	Left Image	Right Image	Steer Angle	Throttle	Speed_n	Epsilon
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	1	0	23.41998
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	1	0	24.00864
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.823999	0	24.89198
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.617178	0	25.24657
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.406294	0	25.49292
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.512264	0	25.71033
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	-0.25	0.798631	0	26.22812
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	-0.45	1	0	26.80527
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	-0.65	1	0	27.3697
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	-0.85	1	0	27.84374
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.796008	0	28.38897
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.573994	0	28.71699
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.361925	0	28.90331
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.153049	0	28.9042
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	-0.1	0.403547	0	28.96272
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	-0.3	0.647913	0	29.1839
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.420725	0	29.44802
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0.175192	0	29.4813
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0	0	29.31334
D:\Self Driving (	D:\Self Drivi	D:\Self Drivin	0	0	0	29.08018

Figure 4.4: Training data model object classification

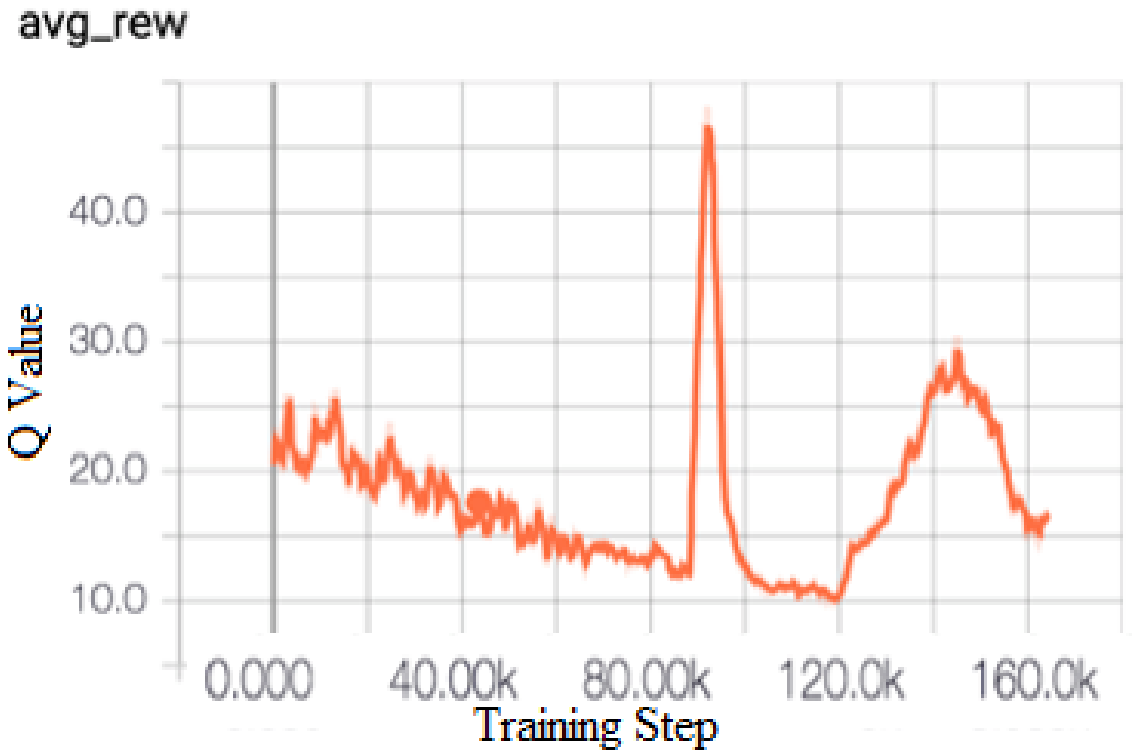

Figure 4.5: Characteristics of DDQN hyper-parameters: Average Reward

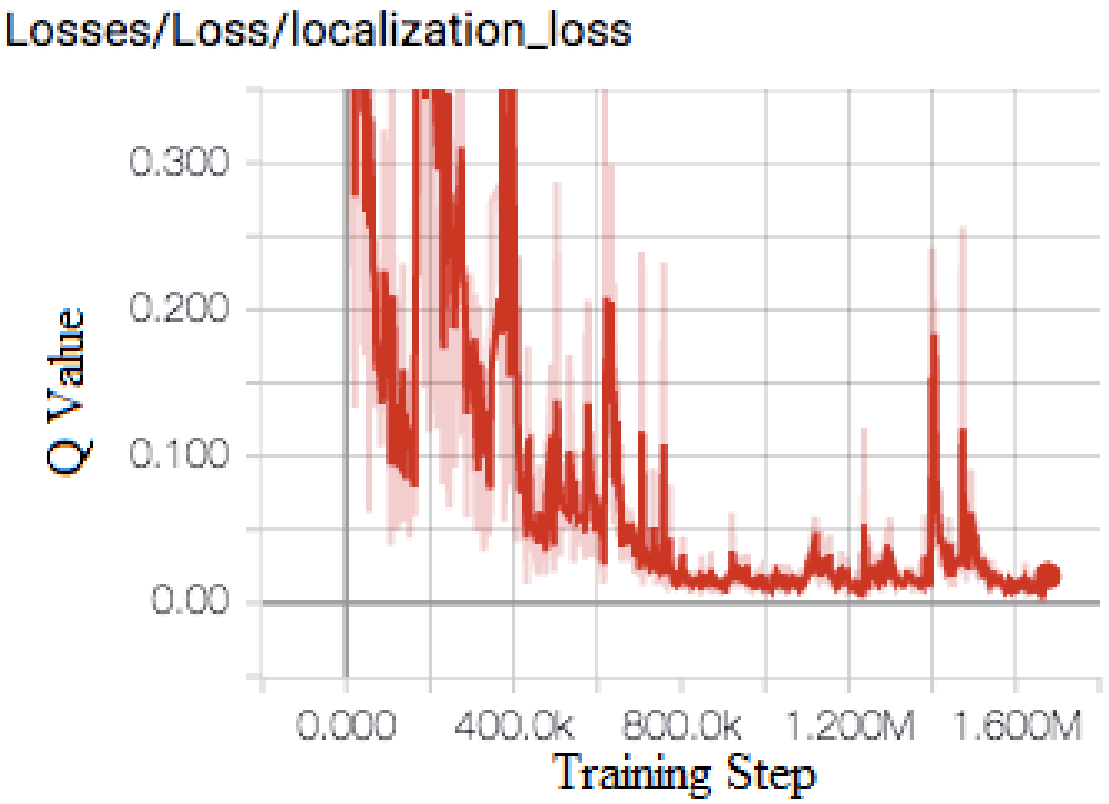

Figure 4.6: Characteristics of DDQN hyper-parameters: Value Loss

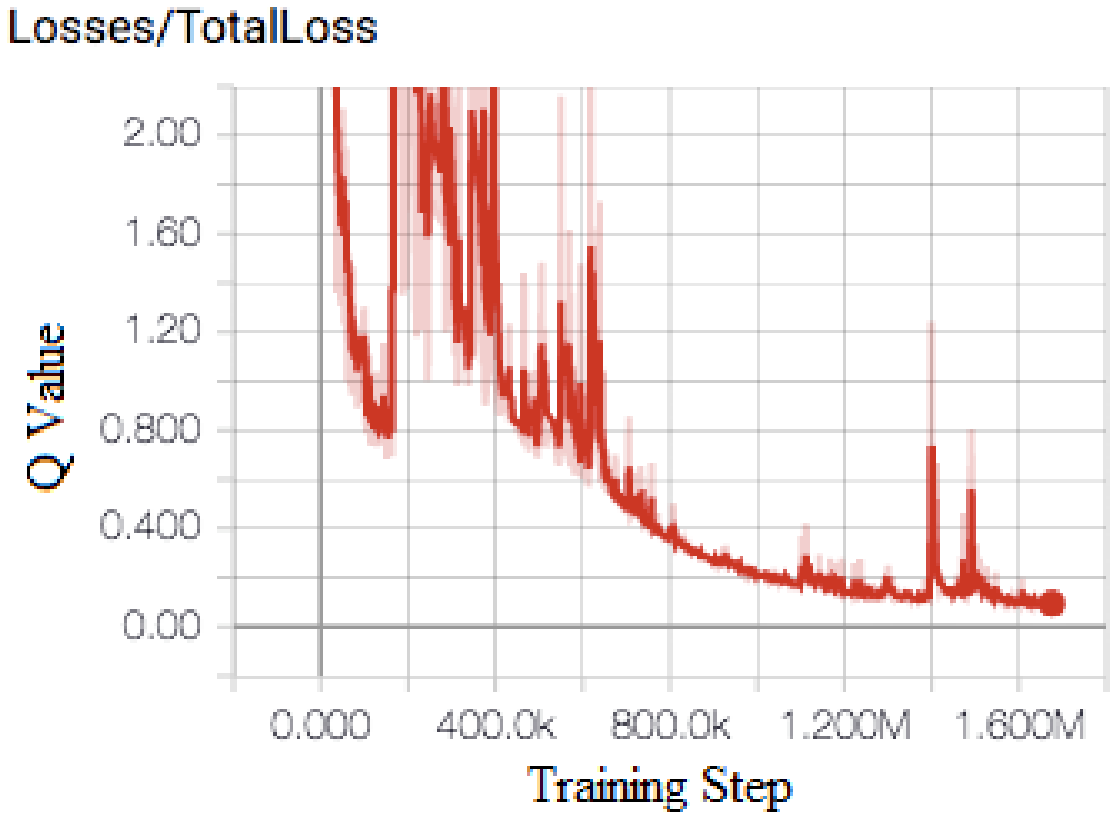

Figure 4.7: Characteristics of DDQN hyper-parameters: Total Loss

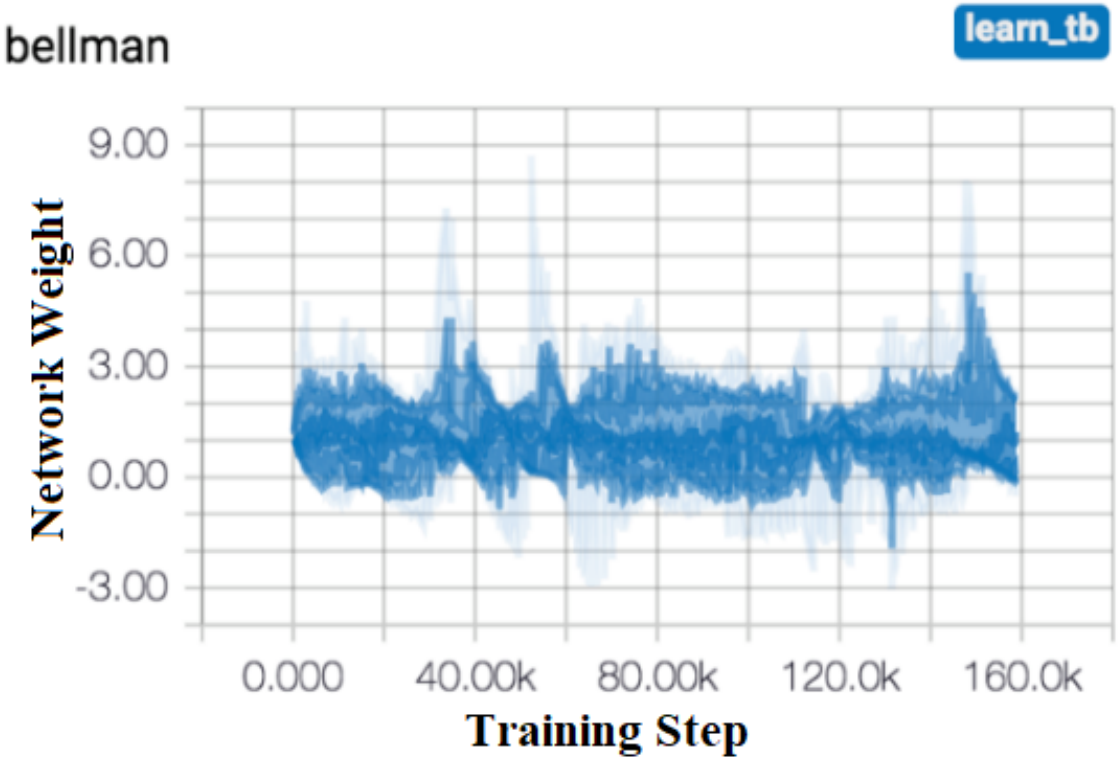

Figure 4.8: Training data values before normalization

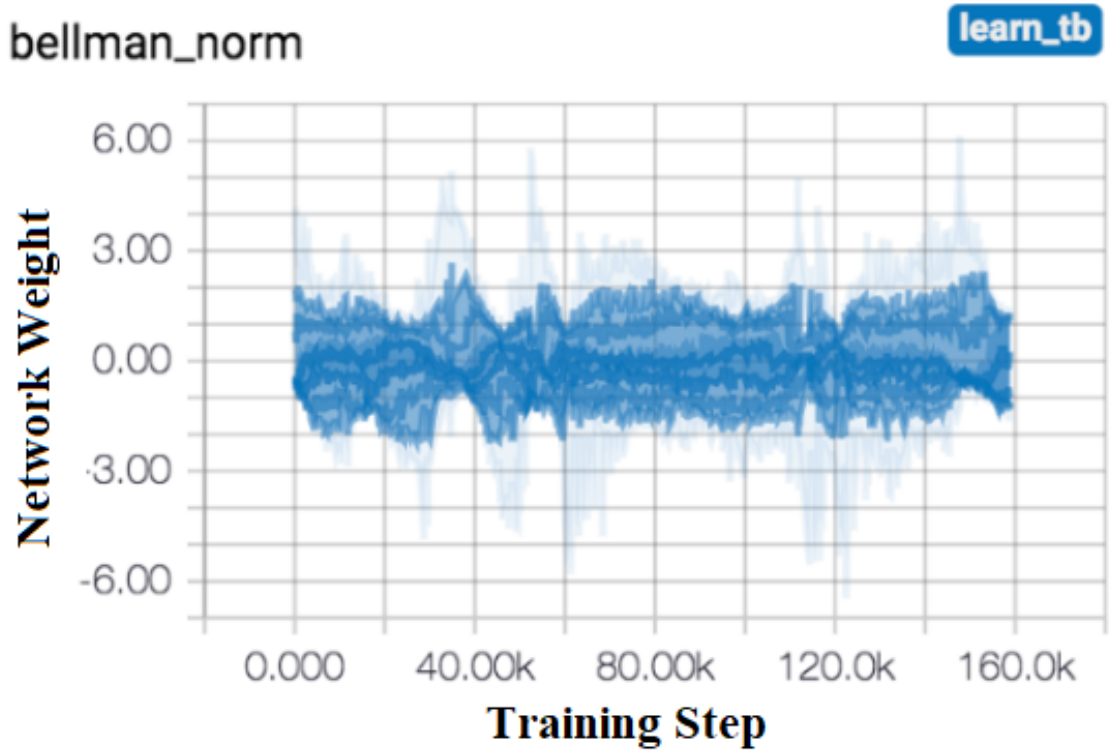

Figure 4.9: Training data values after normalization

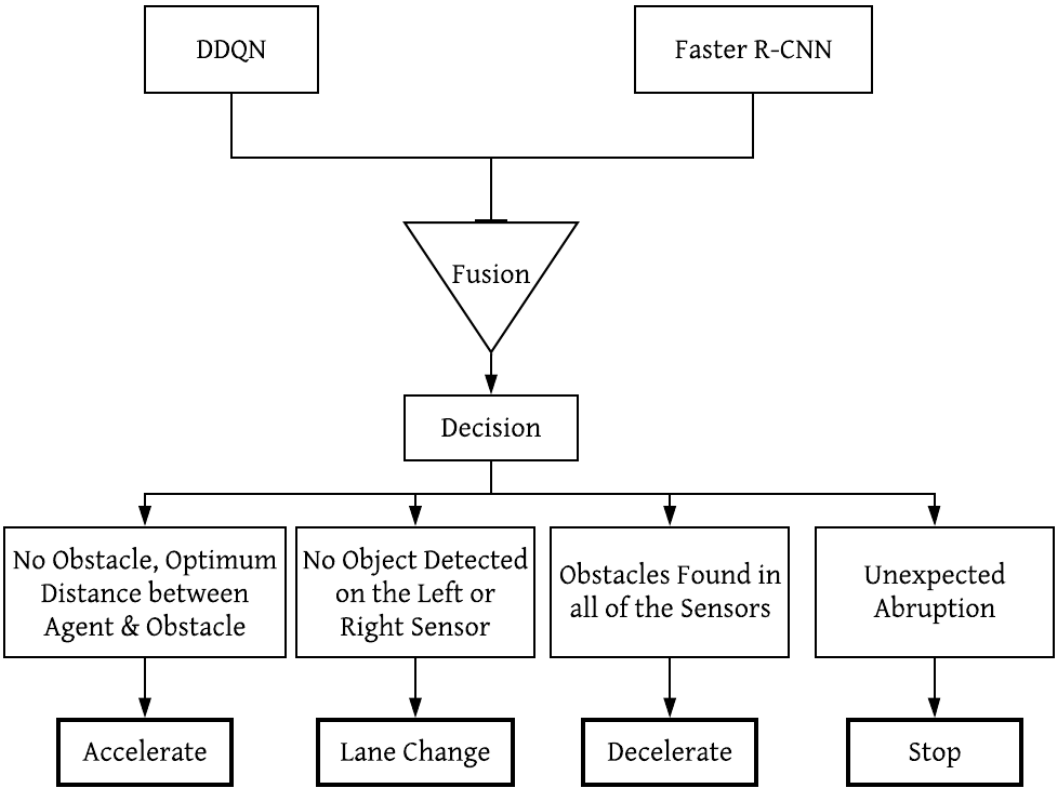

Figure 4.10: Combined Decision Making Process

In this flow diagram (Figure 4.10), the decision making procedure is represented which our agent follows. After getting output result from the DDQN and Faster R-CNN, they are fused and according to that an agent takes the decision. Agent takes the mentioned decisions or actions which satisfy the following condition. Here, four decisions or actions are considered for different scenarios. So, agent's response and decision making are categorized into accelerate, lane change, decelerate and stop modes. If agent finds no obstacle in a certain distance, it will be in acceleration mode. If no obstacle is found in the right or left side of the agent then it will be able to change its lane. It will be in deceleration mode if obstacles are found on both sides of agent and lastly it will immediately stop its exploration for any unexpected abruption.

Agent take these decisions based on the distance calculations of surroundings and the condition which satisfies the action. Moreover, to make the decision making process error free and smooth agent follows a formula for calculating the braking distance precisely. Here is the following calculation used for calculating the braking distance:

$$\text{Distance } = \frac{1}{2} \times \frac{V_{\text{init}}^2}{\text{Gravity}} \tag{4.1}$$

Speed	Reaction Distance	Braking Distance	Total Stopping Distance
40 km/h	1700cm	900cm	2600cm
50 km/h	2100cm	1400cm	3500cm
60 km/h	2500cm	2000cm	4500cm
70 km/h	2900cm	2700cm	5600cm
80 km/h	3300cm	3600cm	6900cm

Table 4.1: Braking Distance Calculation (Dry Road)

Table 4.1 shows the calculation of our braking distance measurement for our agent in dry road. Following the mentioned measurements our agent will compare the braking distance with other surrounding cars. Parameter estimator algorithm (PEA) [26] for calculating the distance between our agent and other cars has been implemented. Figure 4.11 shows the braking situation in terms of distance calculation in GTA V Environment [52].

Figure 4.11: Braking Situation in terms of Distance Calculation in GTA V Environment [52]

4.3 Result

For evaluating the constructed object classifier some tests were run on the proposed model. To understand the accuracy in terms of False positive and True negative for each class, a normalized confusion matrix is constructed, which is shown in Figure 4.12. Here the target values are constituted by the horizontal rows which is the prediction of the model - the ground-truth. The predicted values are also constructed by the vertical columns which is the actual prediction of the model.

Furthermore, on a scale of 0.0 to 1.0 the precision and recall for each class is computed below gradually according to Bicycle, Bus, Person, Motorcycle, Truck, Van,

Car. While recall expresses the potential to find all applicable times in a dataset, precision expresses the share of the data factors our model says was relevant that actually was relevant.The overall result shows that the image classification model is quite strong for detecting objects that will come in the way of our autonomous vehicle.

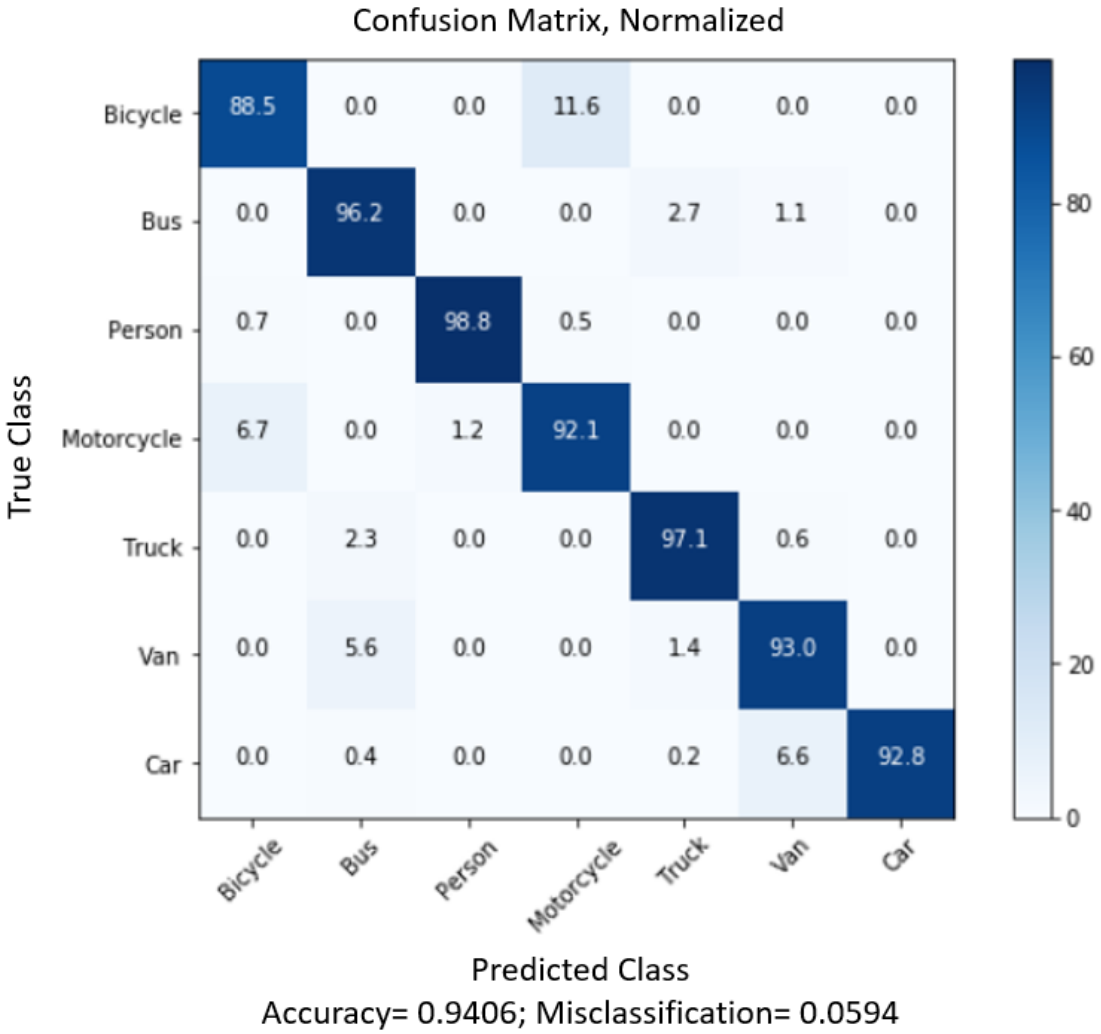

Figure 4.12: Normalized Confusion Matrix for the object classifications

For implementing the Reinforcement Learning based Autonomous Vehicle, DDQN and Faster R-CNN have been used. At the end, the effectiveness of the proposed algorithms is measured. The proposed algorithm is differentiated from other similar algorithm such as- Deep Q learning (Multi input) and Deep Q Learning(image) which basically works on only image processing.

These algorithms are compared in terms of lane changes of the autonomous agent in GTA V game environment. The following figure shows the lane changes result of those algorithms which shows that, this proposed algorithm is better than those algorithms.

Class	Precision	Recall
Bicycle	0.89	0.94
Bus	0.97	0.93
Person	1.0	0.99
Motorcycle	0.93	0.89
Truck	0.98	0.97
Van	0.94	0.93
Car	0.94	1.0

Table 4.2: Precision and Recall values of the classes gradually

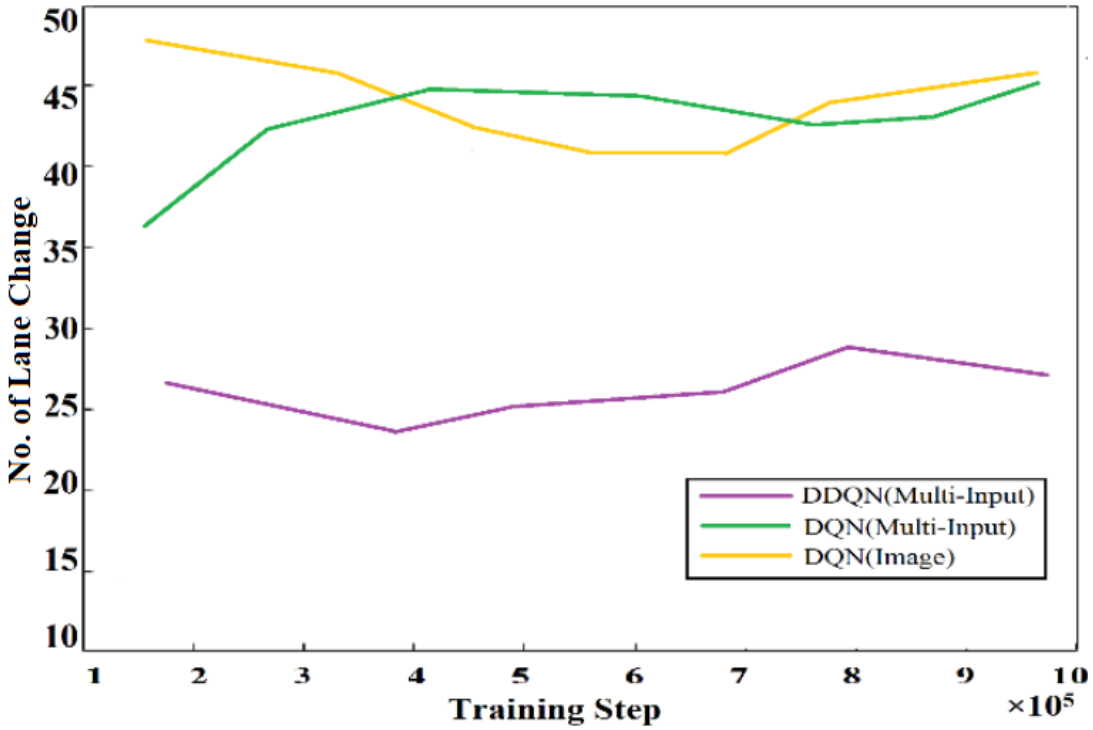

Figure 4.13: Lane Changing comparison of different algorithms of Reinforcement Learning

Reinforcement Learning gives reward based on Q-value which is generated from Q function. That is why, algorithms have been compared in terms of Q-values. Following figure shows the result of Q-values of different algorithms by which it can be said that DDQN is better than other deep Q algorithms for autonomous vehicle. The graph in Figure 4.13, Figure 4.14 and Figure 4.15 are generated from the output section of the algorithm that have been implemented for the uninterrupted traversing of the agent. This traversing algorithm involves reward method which is considered as an inevitable part of deep Q learning to make this decision making process more accurate. Moreover, the reward is delivered to the agent based on the Q-values of every training data.

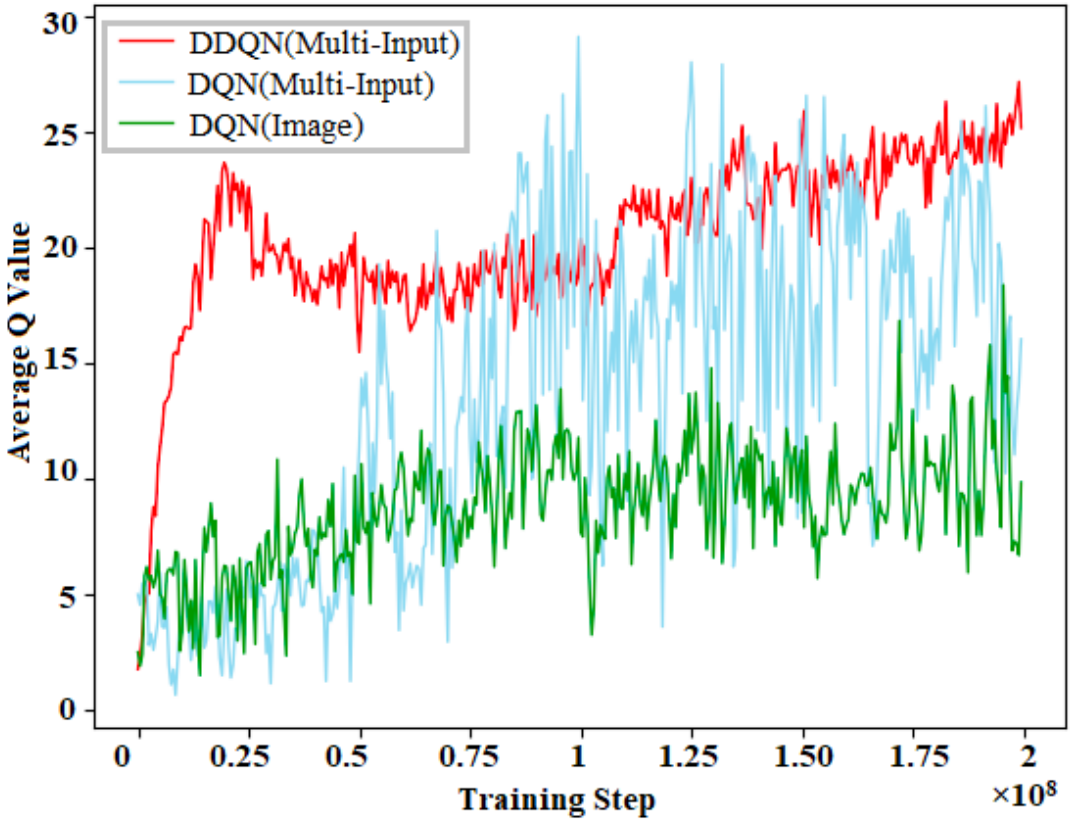

Figure 4.14: Characteristics of DDQN hyper-parameters: Average Q Value

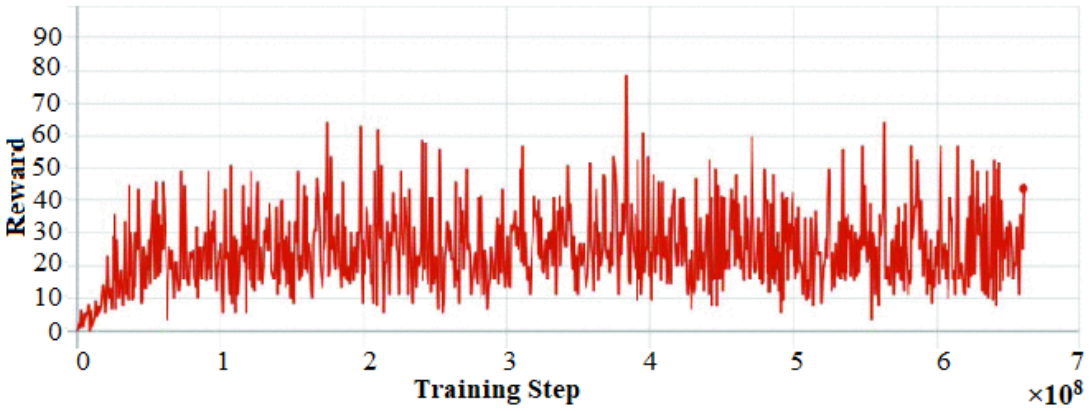

Figure 4.15: Characteristics of DDQN hyper-parameters: Average Reward Value

4.4 Discussion

The integration of Faster R-CNN along with DDQN shows the maximum optimum result in the field of autonomous exploration. The dataset that is used here to train the image classifier shows the accuracy of 94.06%. This accuracy level clearly represents the effectiveness of the classifier to identify any object. Here, the key objective of the image classifier is to identify any object during any unexpected situation the vehicle faces and after the classification of the object then the decision is taken. Otherwise, the whole exploration process continues with the help of the decision that agent makes from the Q- values generating from the DDQN model. Both the system works simultaneously and it makes the exploration process smooth. The only implementation of DDQN model limits the situation handling process of the agent. Therefore, the focus is more on the integration of both models to secure the exploration process without any interruption. As, exploration requires uninterrupted lane changing technique, Markov Decision Process is used to execute this line changing policy. Thus the accuracy level of the agent is obtained. The whole algorithm has been tested on GTA V game environment. Hence, it is very similar to the real world.

Chapter 5

Conclusion

5.1 Research Overview

In this research, DDQN (Double Deep Q Learning) algorithm has been integrated for autonomous exploration of agent. The main objective of DDQN is to perform any action without depending on the model of any environment. Alongside, it enables agents to take optimum decisions and perform required action to produce most effective results. DDQN is executed through reward system policy which handles any stochastic transition without any additional adaptation. In this proposed system it enables the vehicle to take rational action according to the perceived decision. Moreover, the whole system is built on the platform where multi-input architecture has been integrated.

Q-learning is a well accepted learning algorithm which was introduced by Chris Watkins [3]. This algorithm is widely used to solve Markov Decision Process where it shows poor performance by overestimating action values. Therefore, another modified and upgraded algorithm is introduced which is Double Deep Q- learning. This algorithm mainly operates through different policy from Q-learning which handles the overestimation problem that makes learning policy even more fugitive. So, the whole exploration of this automated vehicle is connected with the action values and reward policy which it retrieves from this algorithm and takes optimum decision.

As this research proposal aims to make the exploration process accurate and efficient Faster R-CNN is also integrated. Here, it makes the decision making system more reliable during any unexpected or uneven situation. As it is one of the most efficient object classifiers it takes certain objects from datasets for testing and training to execute the object detection process in a proper way. Acquiring values from Faster R-CNN; reward function can be manipulated which accelerates the efficiency in decision making process and smoothen autonomous exploration of vehicle.

Moreover, another advancement of this autonomous vehicle is, it is not confined to any particular environment. This vehicle does not follow the mapped model for its exploration. So, its exploration range is not limited which makes this system more significant as it is different from other existing autonomous vehicles.

5.2 Research Challenges

DDQN(Double Deep Q-learning) is widely used as it is one of the most optimal algorithms for constructing any automation system [31]. This is also referred as semi-supervised learning model. As the whole system is dependent on the Q-values generated by this algorithm; ensuring its accuracy and solving the overestimation problem is a real struggle in this case. As, overestimation problem can be misleading and its consequences can lead this system to perform oddly so ensuring its absence is must required [15].

This automation vehicle system is model-free learning based. As a result, this does not rely on any model to execute the exploration process. So, if the full focus of exploration process is given to the vehicle it can perform oddly during any unexpected situation like high beams, speed breaker, dead end roads. It even sometimes performs oddly if suddenly any passerby comes across. During the time of initial training for inexperienced cases, it has been examined in this research that it has made many wrong decisions. Another struggle it faces when it has to consider the vehicles which are in parallel sides during the decision making process. However, to overcome this struggle it need to rely on an object classifier and Lidar.

Along with DDQN to enable the agent to take action in an error free way and to make vehicle able to face any unexpected situation Faster R-CNN is also integrated here. The main objective of this classifier is to detect any object accurately and make decision from it. However, sometimes the images that system retrieves through this algorithm are hard to detect and locate its position. Due to discontinuous data it does not work as accurately as it is expected. Sometimes low resolution of images is also the reason behind error in object detection. Additionally, dataset have been used from Google Open Image which has images for 600 object classes. Due to this huge number of classes only specified 7 classes have been extracted to train. Before using Cuda processor it took huge time to train these images approximately took 5hrs for conducting 6k iteration. Due to this many of the objects remained unidentified. Later using Cuda processor iteration numbers have been made larger than before.

5.3 Experimentation and Results

Deep neural networks have been explicitly developed for autonomous vehicle control. It is the technique following which this system is implemented to traverse smoothly in a stochastic environment where the action will be taken according to the Q-values. The implementation of DQN starts with mapping the perception state and then conducting the possible action according to this. Therefore, Fig.3 shows the extraction of the images from camera and then action state is derived using the DQN framework.

Another part of this research is object classification conducting by Faster R-CNN. In this phase of object classification a large dataset has been used where 7 distinct object classes have been separated and images of these object classes have been trained on Tensorflow framework till the total loss has minimized. Total 2360 im-

ages have been trained here where each class have minimum of 300 images using '$OIDv4\_ToolKit$'. Later, using Cuda core processor these images were trained for 13 hours till the total loss is minimized to 0.8. The accuracy level of these object classes are measured through confusion matrix Fig. 5. After several hours training the accuracy level shows its value 0.9406 and misclass is 0.0594. This Fig. 5 shows the percentages of predicted classes and true classes extracted from the dataset in the initial of object classification phase.

Following both techniques that Fig. 4 represents where images from both frameworks are merged and then system using Markov Decision Process makes decision about traversing policy. Here in Fig. 7 shows that the lane changing process measuring from the reward policy with the help of MDP. Here, the reward policy based on the Q-values system receives from the framework. Fig. 8(b) shows the average Q-values and 8(a) shows the average reward. The whole algorithm has been tested on GTA V game environment so it is very similar with the real world.

5.4 Contribution and Impact

The main target of this research is the uninterrupted exploration of autonomous vehicle. This exploration process is ensured to be error free by integrating two different algorithms. Here, DDQN is used for autonomous exploration and also it can make decision measuring its Q-value and can take action according to it. However, if we look at any existing autonomous vehicle system they use a policy map to produce any action states . It sometimes produces the result which is different from any real action. Many existing system only use tradition DQN which leads to system producing unstable action. The main reason behind this problem is overestimation. As, we have implemented DDQN the value of Q-function are considered to be more appropriate than tradition DQN algorithm. Moreover, our research has also focused on optimal path finding process using Markov Decision Process. Therefore, this autonomous vehicle can perform autonomous exploration with ease and can tackle the problem like lane changing and shortage path finding.

Moreover, many existing system of autonomous vehicle follow general reinforcement learning where value is predicted and action is determined according to this which limits its performance and ability to tackle the randomness of the environment. If the system does not comply with the real situation then it does not serve the purpose of autonomous exploration either. Therefore, this research is aimed to cope up with stochastic environment.

Additionally, where most of the existing autonomous vehicle is fully constructed on the frame of neural networks; Faster R-CNN has also been merged along with DDQN to detect any object discovered on the road. Training this system with particular object classes enable this vehicle to identify any object accurately which makes it possible to take any decision in favor of the instructions we are feeding into it.

To summarize, this research contributes to the advancement of technology and uncovers areas which have not been explored and considered before. The contributions of this study are as follows:

- Implementing Double Deep Q-Learning (DDQN) for autonomous traversal.
- Integrating Faster R-CNN for image classification.

5.5 Recommendation and Future Work

The key recommendation for this proposed system is handling the struggle it faces having other vehicles alongside the road. Another one will be increasing the accuracy level during the time of any randomness and taking preferable action complying with the reality. However, the object classifier also has great scopes for improving its performance by lowering the average number of misclassification and by increasing the average number of accuracy level. By configuring confusion matrix; it will simply indicate how much percentage of accuracy has been acquired through these processes. Therefore, we can increase the iteration score while training object classes to overcome the misclassification problem.

Bibliography

[1] P. P. Reddy, *Autonomous car: Deployment of reinforcement learning in various autonomous driving applications*, EasyChair Preprint no. 1305, EasyChair, 2019.

[2] G. A. Kimble, "Hilgard and marquis'" conditioning and learning."", 1961.

[3] C. Watkins, "Learning form delayed rewards", *Ph. D. thesis, King's College, University of Cambridge*, 1989.

[4] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching", *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[5] R. S. Sutton, "Introduction: The challenge of reinforcement learning", in *Reinforcement Learning*, Springer, 1992, pp. 1–3.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey", *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[7] A. G. Barto, "Chapter 2 - reinforcement learning", in *Neural Systems for Control*, O. Omidvar and D. L. Elliott, Eds., San Diego: Academic Press, 1997, pp. 7–30, ISBN: 978-0-12-526430-3. DOI: https://doi.org/10.1016/B978-012526430-3/50003-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780125264303500039.

[8] J. Rennie, A. McCallum, *et al.*, "Using reinforcement learning to spider the web efficiently", in *ICML*, vol. 99, 1999, pp. 335–343.

[9] Jacobs G, Aeron-Thomas A, Astrop A, *Crowthorne, united kingdom: Transport research laboratory*, Estimating Global Road Fatalities, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.174.5207&rep=rep1&type=pdfexternal%20icon.

[10] A. Greenwald, K. Hall, and R. Serrano, "Correlated q-learning", in *ICML*, vol. 3, 2003, pp. 242–249.

[11] M. Coggan, "Exploration and exploitation in reinforcement learning", *Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University*, 2004.

[12] P. Dayan and Y. Niv, "Reinforcement learning: The good, the bad and the ugly", *Current opinion in neurobiology*, vol. 18, pp. 185–96, Sep. 2008. DOI: 10.1016/j.conb.2008.08.003.

[13] X. Glorot and Y. Bengio., "Understanding the difficulty of training deep feedforward neural networks", 2010.

[14] H. V. Hasselt, "Double q-learning", in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[15] H. P. van Hasselt, *Insights in reinforcement learning*. Hado van Hasselt, 2011.

[16] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction", 2011.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks", *arXiv preprint arXiv:1312.6229*, 2013.

[19] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[20] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2147–2154.

[21] ——, "Scalable object detection using deep neural networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2147–2154.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[23] D. P. Kingma and J. Ba., "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[24] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, "Scalable, high-quality object detection", *arXiv preprint arXiv:1412.1441*, 2014.

[25] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3992–4000.

[26] C.-G. J. Gerardo, A.-L. Jesús, and O.-M. Ricardo, "Modeling the turning speed and car following behaviors of autonomous vehicles in a virtual world", *Ingenieria, Investigación y Tecnologia*, vol. 16, no. 3, pp. 391–405, 2015.

[27] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[28] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?", *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 814–830, 2015.

[29] J. Leech, G. Whelan, M. Bhaiji, M. Hawes, and K. Scharring, "Connected and autonomous vehicles–the uk economic opportunity", *KPGM*, 2015.

[30] M. L. Littman, "Reinforcement learning improves behaviour from evaluative feedback", *Nature 521*, no. 7553, pp. 445–451, 2015.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, p. 529, 2015.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.

[33] Q. Fan, L. Brown, and J. Smith, "A closer look at faster r-cnn for vehicle detection", in *2016 IEEE intelligent vehicles symposium (IV)*, IEEE, 2016, pp. 124–129.

[34] T. Ujiie, M. Hiromoto, and T. Sato, "Approximated prediction strategy for reducing power consumption of convolutional neural network processor", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 52–58.

[35] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning", in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[36] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[37] Z. Huang, X. Xu, H. He, J. Tan, and Z. Sun, "Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 4, pp. 730–741, 2017.

[38] S. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. Eng, D. Rus, and M. Ang, "Perception, planning, control, and coordination for autonomous vehicles", *Machines*, vol. 5, no. 1, p. 6, 2017.

[39] G. Prabhakar, B. Kailath, S. Natarajan, and R. Kumar, "Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving", in *2017 IEEE Region 10 Symposium (TENSYMP)*, IEEE, 2017, pp. 1–6.

[40] M.-C. Roh and J.-y. Lee, "Refining faster-rcnn for accurate object detection", in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, IEEE, 2017, pp. 514–517.

[41] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods", *arXiv preprint arXiv:1802.09477*, 2018.

[42] e. a. Hester Todd, "Deep q-learning from demonstrations", *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[43] R. Benenson, S. Popov, and V. Ferrari, "Large-scale interactive object segmentation with human annotators", in *CVPR*, 2019.

[44] K. Min, H. Kim, and K. Huh, "Deep distributional reinforcement learning based high level driving policy determination", *IEEE Transactions on Intelligent Vehicles*, 2019.

[45] Asirt.org, *Road safety facts*, [Online; accessed December 24, 2019]. [Online]. Available: https://www.asirt.org/safe-travel/road-safety-facts/.

[46] Center for Health Statistics and Information Ministry of Healthy, Beijing, China, *Vital registration data*, [Online; accessed December 24, 2019].

[47] Guru99, *Reinforcement learning vs. supervised learning*, [Online; accessed December 10, 2019]. [Online]. Available: https://www.guru99.com/reinforcement-learning-tutorial.html.

[48] Medium, *Convolutional layers*, [Online; accessed December 10, 2019]. [Online]. Available: https://miro.medium.com/max/1644/1*uAeANQIOQPqWZnnuH-VEyw.jpeg.

[49] Medium.com, *Architecture of region proposal network*, [Online; accessed December 10, 2019]. [Online]. Available: https://miro.medium.com/max/1280/1*S_-8lv4zP3W8IVfGP6_MHw.jpeg.

[50] nhtsa.gov, *Automated driving system*, [Online; accessed December 24, 2019]. [Online]. Available: https://www.nhtsa.gov/vehicle-manufacturers/automated-driving-systems.

[51] onlinemasters.ohio.edu, *The future of driving*, [Online; accessed December 24, 2019]. [Online]. Available: https://onlinemasters.ohio.edu/blog/the-future-of-driving/.

[52] Rockstar North, *Grand theft auto v*, [Online; accessed December 24, 2019]. [Online]. Available: https://www.rockstargames.com/V/info.

[53] U.S. Energy Information Administration, *Study of the potential energy consumption impacts of connected and automated vehicles*, [Online; accessed December 24, 2019]. [Online]. Available: https://www.eia.gov/analysis/studies/transportation/automated/pdf/automated_vehicles.pdf.

[54] World Health Organization, *Global status report on road safety 2018*, [Online; accessed December 24, 2019]. [Online]. Available: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/.

Double Deep Q-Learning

Algorithm 1 : Double Q-learning

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau << 1$
for each iteration **do**
  for each environment step **do**
    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$
    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
  for each update step **do**
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute target Q value:
      $Q^*(s_t, a_t) \approx r_t + \gamma\ Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$
    Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$
    Update target network parameters:
      $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$

Adam Optimizer

Algorithm 1. Adam Optimization Algorithm

Require: Objective function $f(\theta)$, initial parameters θ_0, stepsize hyperparameter α, exponential decay rates β_1, β_2 for moment estimates, tolerance parameter $\lambda > 0$ for numerical stability, and decision rule for declaring convergence of θ_t in scheme.

procedure ADAM(f, θ_0 ; α, β_1, β_2)
  m_0, v_0, $t \leftarrow [0, 0, 0]$ # Initialize moment estimates
  # and timestep to zero
  # Begin optimization procedure
  while θ_t has not converged **do**
    $t \leftarrow t + 1$ # Update timestep
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ # Compute gradient of objective
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ # Update first moment estimate
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (g_t \odot g_t)$ # Update second moment estimate
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ # Create unbiased estimate $\widehat{m}_t$
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ # Create unbiased estimate $\widehat{v}_t$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \lambda)$ # Update objective parameters
  return θ_t # Return final parameters