SAFE DRIVING OF AUTONOMOUS VEHICLES THROUGH IMPROVED DEEP REINFORCEMENT LEARNING

by

Abhishek Gupta

MSc, Intelligent Systems, De Montfort University, Leicester, United Kingdom, 2013

BE, Electronics and Telecommunication Engineering, University of Pune, India, 2011

A Thesis

presented to Ryerson University

in partial fulfilment of the

requirements for the degree of

Master of Applied Science

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2020

Author's Declaration

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Safe Driving of Autonomous Vehicles Through Improved Deep Reinforcement Learning

Master of Applied Science, 2020
Abhishek Gupta
Electrical and Computer Engineering
Ryerson University

In this thesis, we propose an environment perception framework for autonomous driving using deep reinforcement learning (DRL) that exhibits learning in autonomous vehicles under complex interactions with the environment, without being explicitly trained on driving datasets. Unlike existing techniques, our proposed technique takes the learning loss into account under deterministic as well as stochastic policy gradient. We apply DRL to object detection and safe navigation while enhancing a self-driving vehicle's ability to discern meaningful information from surrounding data. For efficient environmental perception and object detection, various Q-learning based methods have been proposed in the literature. Unlike other works, this thesis proposes a collaborative deterministic as well as stochastic policy gradient based on DRL. Our technique is a combination of variational autoencoder (VAE), deep deterministic policy gradient (DDPG), and soft actor-critic (SAC) that adequately trains a self-driving vehicle. In this work, we focus on uninterrupted and reasonably safe autonomous driving without colliding with an obstacle or steering off the track. We propose a collaborative framework that utilizes best features of VAE, DDPG, and SAC and models autonomous driving as partly stochastic and partly deterministic policy gradient problem in continuous action space, and continuous state space. To ensure that the vehicle traverses the road over a considerable period of time, we employ a reward-penalty based system where a higher negative penalty is associated with an unfavourable action and a comparatively lower positive reward is awarded for favourable actions. We also examine the variations in policy loss, value loss, reward function, and cumulative reward for 'VAE+DDPG' and 'VAE+SAC' over the learning process.

Acknowledgement

I would like to express my heartfelt gratitude to my supervisor, Prof. Alagan Anpalagan for giving me the opportunity to pursue my research. I am forever indebted to Prof. Alagan for his tireless enthusiasm, constant support, and critical insights throughout my MASc, which made this thesis possible. I must also thank him for his style of supervising that offers continued support, guidance, and constructive feedback.

I am grateful to Prof. Ling Guan for being my thesis co-supervisor and for the valuable guidance at various stages of my studies. I thank all the other members of my defense committee for giving me constructive advice to endeavor toward a more solid thesis.

I sincerely thank Dr. Ahmed Shaharyar Khwajah for extending his stimulating discussions, feedback, and thought provoking research topics. The endless conversations we have had throughout were enjoyable as well as memorable.

I must acknowledge Dr. Kandasamy Illanko for introducing me to the topic of convolutional neural networks, recurrent neural networks, and the vast scope of applications of deep learning. It would not be an exaggeration to say Prof. Aditya Abhyankar, Dean, Savitribai Phule Pune University (Faculty of Technology), India, shaped me as a researcher and kindled my interest in the ever evolving field of signal processing and communication.

It was my pleasure to be a member of WINCORE Lab and to be part of the research group. It was an honour to spend time with all the colleagues and research fellows, and I thank them for their willingness to help and the cordial environment. I would also like to extend my special thanks to the department of Electrical, Computer, and Biomedical Engineering (ECBE), Faculty of Engineering and Architectural Sciences (FEAS), and the School of Graduate Studies (SGS) at Ryerson University to provide me a chance to pursue my master's degree and all kinds of support for my studies.

In addition, I would like to thank to my family for their trust and dedication during my master's studies, and my friends from all communities for their encouragement.

Table of Contents

List of Tables

List of Figures

Notations and Symbols

$x_c(t)$, $y_c(t)$	The vehicle is represented as a circle with centre $x_c(t)$, $y_c(t)$ at time t
$x_c(t_i)$	X-coordinate of centre of vehicle position at i^{th} instance of time, t_i
$y_c(t_i)$	Y-coordinate of centre of vehicle position at i^{th} instance of time, t_i
x_o, y_o	An obstacle is represented as a square with centre $x_o(t)$, $y_o(t)$ at time t
x_1, y_1	An arbitrary point on the road
x_2, y_2	An arbitrary position of vehicle on the road
x_3, y_3	An arbitrary position of obstacle on the road
Δt	Difference between two subsequent timeframes when vehicle navigates
c_R	Y-intercept of equation defining vehicle position on the road
α	Set of points defining the road (driving environment), constrained between two straight lines
β_{veh}	Initial set of points of the vehicle when placed on the road
m	Slope of the striaght line path
c_{veh}	Y-intercept made by the vehicle while travelling in a straight line path
veh_{road}	Indicates vehicle remains on the road and does not drift off-track
β_{obs}	Other vehicles (obstacles) on the road
$\alpha_c^{t_i}$	Indicates a vehicle of radius c is present on the road α at time t_i
w	Width of the obstacle
$\min_{<V_r,S>} \| \mathbf{V_r} \|$	The Bellman value function to be minimized at a given state
v	Velocity
v_x	Velocity component in x-direction
v_y	Velocity component in y-direction
r	Reward
d	Distance from the middle of the road
p	Penalty
Ψ	Additional penalty
$\| <,> \|$	Inner product
S_t	State at time t
A_t	Action at time t
R_t	Reward at time t

γ	Discount rate (where $0 \leq \gamma \leq 1$)
G_t	Discounted return at time
$\mathcal{R}$	Set of all rewards
$\mathcal{S}^+$	Set of all states (including terminal states)
P	State transition probability matrix
S	finite set of states
A	finite set of actions
$\mathcal{A}(s)$	Set of all actions available in state s
$\mathbb{P}$	Probability distribution of the variable under consideration
$\mathbb{E}$	Expectation
ac_1	Indicates a vehicle is set to take accelerating action
ac_2	Indicates a vehicle is set to avoid accelerating action
s_1^d	Indicates that a set of actions leads to safe driving
s_2^d	Indicates that a set of actions leads to unsafe driving
$P(s_1^d\|ac_1)$	Probability that an action is unsafe if vehicle takes accelerating action
$P(s_2^d\|ac_1)$	Probability that an action is safe if vehicle takes accelerating action
$P(s_1^d\|ac_2)$	Probability that an action is unsafe if vehicle avoids accelerating action
$P(s_2^d\|ac_2)$	Probability that an action is safe if vehicle avoids accelerating action
s_t	The state of the driving environment and the vehicular agent at time t
a_t	Action taken by vehicular agent at time t
r_t	Reward (or penalty) for a specific action a_t taken by vehicular agent at time t
$P_{ss'}$	Probability of transtion from one state to next state
D	Observed data
B	Prior background knowledge
$P(s\|D, B)$	Belief about the environment
π	The learning policy of the DRL algorithm
$\pi(a\|s)$	Policy for a given state-action pair $(a\|s)$
a^*	Optimal action under a given state
$P(s', r\|s, a)$	Probability of next state s' and reward r, given current state s and current action a ($\mathbb{P}(S_{t+1} = s', R_{t+1} = r\|S_t = s, A_t = a)$)
v_π	State-value function for policy π ($v_\pi(s) \doteq \mathbb{E}[G_t\|S_t = s]$ for all $s \in \mathcal{S}$)
q_π	Action-value function for policy π ($q_\pi(s, a) \doteq \mathbb{E}[G_t\|S_t = s, A_t = a]$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)

v_*	Optimal state-value function ($v_*(s) \doteq \max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$)
q_*	Optimal action-value function ($q_*(s,a) \doteq \max_\pi q_\pi(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
V^*	Optimal value function
$Q^*(s,a)$	Optimal state-action $(s\|a)$ value function
$Q^\pi(s,a)$	State-action $(s\|a)$ value function for policy π
G_t	Return, signifies the aptness of reward at a given state
x_t	Denotes the t-th frame of the image dataset
S_t	Set of states at previous times $s_{t-n+1}, s_{t-n+2}, ..., s_t$
A_t	Set of actions at previous times $a_{t-n+1}, a_{t-n+2}, ..., a_t$
$F(X_t, S_t, A_t)$	Function representing state-action tuple for a specific environment
z	Gaussian space
ρ	Pixel space defined as a hypersphere of radius ρ
$\mathcal{L}$	Error in policy learning
$\mathcal{L}_{prior}$	Prior error due to random motion in environment
$\mathcal{L}_{DDPG}$	Error in DDPG network
$\mathcal{L}_{SAC}$	Error in SAC network
$D_{KL}(q(z\|\hat{\sigma}_{0,t})\|\|p(z))$	Kullback-Liebler divergence between the states at different timestamps
σ	Output of the VAE-SAC network after regularization
$p(z)$	Prior distribution of DDPG training data
$q(z\|\hat{\sigma}_{0,t})$	Probability distribution of the output states to calculate Kullback-Leibler divergence
$\hat{\sigma}_{0,t}$	VAE-SAC reward at t^{th} timeframe
μ	Output of the VAE-SAC network after training
ϵ	Gaussian random vector of same dimensions as μ and σ

List of Abbreviations

A3C	Asynchronous Advance-Actor Critic
AI	Artificial Intelligence
BDT	Bayesian Decision Theory
CAV	Connected and Autonomous Vehicles
CNN	Convolutional Neural Networks
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DQN	Deep Q-Networks
DRL	Deep Reinforcement Learning
GPS	Global Positioning System
ICT	Information and Communication Technology
IoV	Internet of Vehicles
ITS	Intelligent Transportation Systems
LQG	Linear-Quadratic-Gaussian
MDP	Markov Decision Process
ML	Machine Learning
NN	Neural Networks
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
SAC	Soft Actor Critic
SAE	Society of Automotive Engineers
SRL	State Representation Learning
SSD	Single Shot Multibox Detector
SVM	Support Vector Machines
TORCS	The Open Racing Car Simulator
VAE	Variational Auto Encoder

List of Appendices

Chapter 1

Introduction

Recently, autonomous driving (AD) has received significant attention from the research community as well as the industry, and it forms a critical component of the connected and autonomous vehicles (CAV) and internet of vehicles (IoV) framework [1], [2]. A long-standing goal of artificial intelligence (AI) has been to drive a vehicle in a safe manner [5]. With AI theoretically anticipated to provide sustained attention and focus than humans, yet fully autonomous driving requires a level of intelligence that surpasses the ones achieved so far by AI agents [6], [7]. The intelligent transportation systems (ITS) combined with deep learning (DL) is set to usher in an era of following technology disruptions:

- Autonomous vehicles are set to boost shared economy through car pooling and ride-sharing [8].
- Autonomous vehicles are viewed as an ultimate disruption machine, which could considerably reduce the usage of gasoline cars reducing wastage of time, space, lives, money, and energy [8].
- Increasing number of technology industries are collaborating with the transportation and automobile sector to advance the field of self-driving vehicles [6].
- Autonomous vehicles can evolve cloud based models such as car-as-a-service (CaaS),

vehicles-as-a-service (VaaS), and rides-as-a-service (RaaS) [6].

It is anticipated that urban transportation would comprise a mix of manual, semi-automatic, and fully autonomous vehicles by 2025 [1]. Current autonomous vehicles have reached SAE level-3 autonomy in simulated and restrained driving environments. However, in order to deal with the real-life driving scenarios, to safely navigate the traffic, to reduce travel delay, and to avoid congestion, novel object detection and scene perception techniques are required [2]. Environment perception and cognition to gain a comprehensive understanding of the driving scenarios are considered critical enablers for connected and autonomous vehicles. Among many applications, object detection and safe navigation around obstacles rely heavily on a vehicle's ability to discern meaningful information from the surrounding data [5]. One technique to enhance perception in autonomous vehicles is reinforcement learning that teaches machines through constant interaction with the environment, learning from past experiences, and subsequently improving upon them [6]. In the last decade, reinforcement learning combined with deep learning, known as deep reinforcement learning (DRL) has emerged as a strong artificial intelligence paradigm that has been successfully tested on various gaming applications [7].

1.1 Background

Recent advances in DL have intensified research to develop autonomous agents with human-level capabilities [9]. A prevalent data-driven machine learning (ML) technique known as supervised learning allows these agents to learn from a pre-existing collection of data. However, dealing with massive amount of labelled data in real-world scenario is no longer an optimal or feasible solution [10]. The deep reinforcement learning (DRL) methodology, a combination of the existing deep learning (DL) and reinforcement learning (RL) scheme, is currently being studied as a baseline format for the self-driving vehicles [11]. Self-driving cars, also known as autonomous vehicles (AV), driverless cars, smart transportation robots

(STR) or robocars are one of the most speculated scientific inventions with a potential to change the way we commute [12]. The recent and broader implications of self-driving cars incorporate integration with novel infrastructure, smart cities, urban planning with provisions for advanced cyber-security, privacy, and insurance [13]. It is worth mentioning that while the self-driving cars have gained intense attention in the last decade, driverless transportation has been in existence for over a decade in form of driverless shuttles, driverless trains, and autonomous transportation pods [12]. Trains are a prominent example of widespread use of self-driving technology [14]. Some of such train examples include the:

- SkyTrain in Vancouver, Canada [15]
- Docklands Light Railway (DLR) in London, United Kingdom [16]
- Yurikamome in Tokyo, Japan [14]
- London Heathrow airport's ultra-pods [16]

These autonomous rail systems transport thousands of passengers on a daily basis. Authors in [16] note that the majority of passengers commuting through self-driving trains were not worried about using those trains. However, the aforementioned trains and autonomous pods operate on enclosed tracks, isolated from the public roads, and bypass the need to interact with other vehicles or pedestrians [15]. In contrast, self-driving cars are set to encounter various users, thereby resulting in complex interactions and the possibility of collision [17]. Whether people will be as accepting of self-driving cars as they appear to be of existing autonomous transport is an active area of research [15]. The current trends in DL and AI applied to ITS pave the way for fully autonomous vehicles. Traditional DL techniques such as convolutional neural networks (CNN) and recurrent neural networks (RNN), along with ML techniques such as clustering, fuzzy logic, and support vector machines (SVM) have demonstrated success to address challenges pertaining to autonomous driving [11].

1.2 Motivation and Objective

One of the earliest four-layer neural networks (NN) applied to autonomous vehicles consists of one input layer, two hidden layers, and one output layer. The NN takes relative speed, desired speed, follower speed and gap distance as inputs to predict follower acceleration [18]. A modified version of NN with one hidden layer can predict vehicle acceleration by considering instantaneous reaction-time (RT) delay [19]. Until recently, the creation of NNs was often a time-consuming task. However, with the ubiquitous availability of cloud based processors such as Google Cloud Platform (GCP) and Amazon Web Services' (AWS) Elastic Compute-2 (EC2), the complex process of fine-tuning and optimizing NN architectures have been extensively simplified [20]. This has led to a surge in research activities to achieve the quality and the speed needed to simulate, test, and run autonomus vehicles using various DL paradigms. The conventional NN-based models have undergone significant improvements with the advent of recurrent neural networks (RNN), convolutional neural networks (CNN) and deep neural networks (DNN) [11].

Recently, DRL has been widely applied to various problems, predominantly in game playing [21], [22]. The DRL paradigm has been extended to other domains such as autonomous vehicles and has opened new research avenues [23]. A model-free DRL technique known as Q-learning offers a compelling technique to further explore the problem of autonomous driving without explicitly modelling the driving environment [24]. The current study is a step forward to use a deep reinforcement learning (DRL) to model safe autonomous driving behavior. Our results have been compared to other data-driven models based on the accuracy, maximum error-free drive-time before following too closely, and maximum error-free drive-time while overtaking [25]. These state-of-the-art developments in deep learning and reinforcement learning were the principal motivation behind this work.

1.3 Contributions and Organization

The main contribution of this thesis is the application of deep reinforcement learning (DRL) to train an autonomous vehicle and to evaluate the smoothness and effectiveness of drive mechanism and control. The method uses three DRL techniques namely the variational auto encoder (VAE), deep deterministic policy gradient (DDPG), soft actor-critic (SAC) to achieve safe autonomous driving. The simulations and experimental results provide useful insights when extending the design to real world scenarios. The thesis further contributes towards understanding the advantages, shortcomings, and pitfalls of modeling autonomous driving problem as a Markov decision process (MDP) problem and a partially observable MDP (POMDP) problem. The thesis explores the role of DRL to complement the autonomous driving problem modeled using an MDP/POMDP. The results provide a significant groundwork for considering solutions to some autonomous driving problems using standalone DRL frameworks.

The main contributions of this thesis can be summarized as follows:

- Proposing a DRL based solution using VAE, SAC, and DDPG to solve the autonomous driving problem formulated as a MDP. We propose a DRL algorithm that combines the feature extraction capabilities of unsupervised CNNs with the fast and powerful batch RL approach of DDPG, VAE, and SAC.

- Analysis of the autonomous driving behavior, verification of the proposed DRL scheme, and performance comparison of VAE+DDPG and VAE+SAC approaches. This is done by the implementation of VAE, DDPG, and SAC based autonomous driving in DonKey simulator, that closely approximates real-world driving conditions.

The remainder of the thesis is organized as follows: Chapter 2 summarizes the existing developments in self-driving vehicles from perception and object detection point of view, and the role of DRL in achieving improvements over existing methodologies. We compare some applicable solution strategies and discuss the application of deep deterministic policy

gradient (DDPG), soft actor-critic (SAC), and variational autoencoder (VAE) to achieve safe autonomous driving.

In Chapter 3, the problem of safe autonomous driving using DRL is presented and formulated for a specific scenario. The problem formulation is followed by the discussion of solution strategy. This chapter also defines the loss function, proposed penalty, and value entropy. The experimental setup with DonKey simulator is discussed in Chapter 4, outlining the design requirements, available resources, and constraints. This chapter shows the detailed results with a focus on the main contribution of this thesis, the improved learning with the use of VAE, DDPG, and SAC algoritims. The simulation results and performance comparison of all the proposed methodologies are presented in this chapter. In Chapter 5, we conclude the contributions of this thesis and discuss the future research topics.

Chapter 2

Literature Review of Deep Reinforcement Learning for Autonomous Vehicles

2.1 Introduction

Engineers, computer scientists, and applied mathematicians all agree that if the last two decades belonged to the internet and communication revolution, the current decade and the coming decade would belong to the artificial intelligence (AI) revolution [26]. A sub-set of AI, known as machine learning (ML), and a further sub-set of ML known as deep learning (DL) have opened vast avenues of unprecedented research. Supervised learning does not scale computationally well when applied to autonomous driving [27], [28]. Reinforcement learning (RL) is unsupervised, and following the selection of the relevant features, the appropriate sequence of actions is based on a relevant reward function [23]. Incorporating recurrent neural networks (RNN) for information integration enables autonomous vehicles to handle partially observable scenarios; however, it adds to computational overhead on the embedded hardware to focus on relevant information [29]. Table 2.1 outlines key features of partially-

automated and fully autonomous vehicles as defined by the society of automotive engineers (SAE) [5].

Table 2.1: Brief overview of SAE levels of automation in vehicles [5].

SAE level	**Name**	**Human-centered/ Autonomous**	**Monitoring of driving environment**	**Fallback Performance of dynamic driving tasks**	**Example features**
0	No automation	Human driver is completely in-charge of all aspects of driving	Human driver	Human Driver	No system-driven capability
1	Driver Assistance	Human driver in-charge of all aspects of driving with	Human driver	Human Driver	Some minor driving modes are system capable
2	Partial Automation	Human driver in-charge of all aspects of driving	Human driver	Human Driver	System driven steering and acceleration/ deceleration
3	Conditional Automation	Vehicle assists human driver in some non-critical aspects of driving	System	Human Driver	Huamn driver expected to respond when need arises
4	High Automation	Human driver in-charge of very few aspects of driving	System	System + Human Driver	Human driver only need to intervene if unavoidable
5	Full Automation	System in-charge of all aspects of driving	System	System	Human can totally disengage from driving tasks

2.2 Recent Advances and Bottlenecks in Realizing Self-Driving Vehicles

Analysis of the behavior and strategies frequently employed by safe human drivers is essential in the development of autonomous vehicles (AV) [30]. Considerable research has gone into

extracting the driving strategies adopted by drivers and to model their behavior without human intervention in varying situations [30]. In the last decade, ML algorithms such as fuzzy logic control, predictive and adaptive control, hybrid dynamical models and Markov chain models have been used to model the driving behavior. The predictive control based theories successfully predicted and reproduced human driving behavior under constrained environments [31]. However, the method relied heavily on data collection and representation as prior knowledge about the rule base was essential. The accuracy of Markov chain models depends on the prior knowledge about the state transition probability between each state pairs [32]. In realistic scenarios, the diverse driving conditions to be encountered are unlikely to be entirely known in advance. With the emergence of and rapid breakthroughs in deep learning (DL), attempts to learn driving behavior directly from the driving data without prior knowledge of the driving conditions have gained prominence in recent years [33], [14].

2.2.1 Advantages of self-driving cars

Information and communication technology (ICT) continues to find applications in transportation to reduce collisions, reduce pollution, ameliorate mobility issues, modernize public transportation, and share resources, materials and space [12]. According to a study, there are 1.3 million deaths globally every year due to drunk, drugged, distracted, or drowsy driving [34]. Autonomous AI systems that can save lives by eliminating these human follies offer the following advantages that motivate research in self-driving cars:

- For users, the advantages are reduced stress, faster commutes, reduced travel times, enhanced user productivity, optimum fuel consumption, reduced carbon emissions. These cars can be programmed to drive defensively, stay clear of blind spots, and follow speed limits [12].

- Self-driving cars would lead to enhanced roadway capacity, reduced road casualties and number of on-road driving related accidents [12].

- Reduced accidents are expected to be beneficial for children and the elderly, encouraging people to feel comfortable and amiable towards self-driving cars [34].

- Self-driving electric cars would introduce a greener mode of transport, leading to less greenhouse and noise pollution, along with increased mobility for the elderly and disabled people [34].

- In the current driving landscape, cars are parked for a long time. With self-driving cars, parking lots can be converted to parks [8].

- Self-driving cars would be equipped to improve scheduling and routing, and provide best routes to improve travel times, while also lowering the cost of moving from one point to another [6].

- Self-driving cars are envisioned to eliminate drunk driving issues, eliminate issues related to distracted driving, texting and other cell phone use, less braking and accelerating, and less gridlock on highways [13].

- Although self-driving cars would reduce or even eliminate car ownership, they would expand shared access, keep transportation personalized, efficient and reliable [20].

2.2.2 Probable disadvantages and drawbacks of self-driving cars

Cars are one of the most widespread and readily available modes of transportation. While technology has developed safer cars, driving is still a dangerous undertaking [30]. Self-driving cars formulate a scenario where a few lines of source codes, coupled with AI decide the life of a human being [6]. Some disadvantages of self-driving cars are as follows:

- Considering the number of people employed in the transportation sector, the foremost catastrophic consequence of self-driving cars would be elimination of jobs in the transportation industry [12].

- The acceptance of self-driving technology at philosophical, ethical and technological levels is a fundamental research problem in psychology and cognitive science. It is argued that in case autonomous vehicles and AI systems malfunction, a person would not die or suffer injuries if they themselves were in control of the system [34].

- Although the role of AI in our society is consistently evolving, an AI system making critical decisions needs to respect societal values and conform to social norms to gain accpetance [8].

- Driving at intersections without traffic lights, malfunctioning traffic lights, uncontrolled intersections, busy intersections, regions with humans in close proximity are a challenge for self-driving cars [6].

- As self-driving cars use global positioning system (GPS) for localization, they are deemed unsuitable to drive in non-mapped areas [13].

- The scope of car's connectivity, the car being online at all times, makes it susceptible to hacking. The safety and convenience offered by self-driving cars might compromise privacy of passengers as their moves will be tracked and logged unawares [20].

2.2.3 Data-driven autonomous driving models

The availability of high-fidelity traffic data and consequential data-driven approaches model an autonomous vehicles' car-following behavior directly from human-behavior data [1]. The autonomous driving tasks can be broadly classified into following two steps [35]:

(a) Environment Perception and

(b) Cognition

The cognition stage can be subdivided into following two categories [36], [37], [35]:

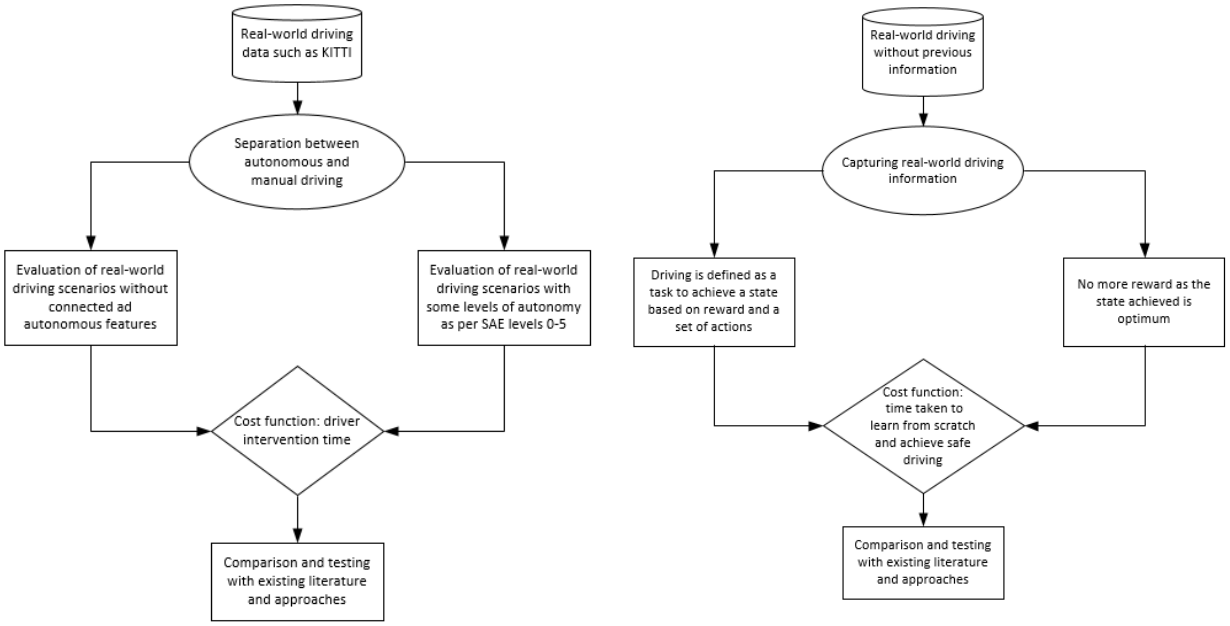
(i) Prediction and

(ii) Planning

Autonomous vehicles interpret the sensor data, identify objects, and monitor the surroundings over a period of time for effective path prediction and planning [38].

2.2.4 Empirical decision-making system for autonomous vehicles

Cognition in autonomous vehicles is primarily a vision-based task that consists of environment recognition, entity identification such as pedestrian detection, traffic sign recognition, and detection of static objects [19]. Image segmentation algorithms based on low-level feature extraction appended with shallow trainable architectures such as support vector machines (SVM) have been widely adopted in computer vision [39]. Figure 2.1 compares data-driven autonomous driving approach with DRL based autonomous driving approach.



(a) Data-driven approach to autonomous driving

(b) Deep reinforcement learning based approach to autononomus driving

Figure 2.1: A comparison of data-driven and deep reinforcement learning based approaches to autonomous driving [1], [2].

Information integration over continuous-time is a key feature offered by DRL where successive optimum decisions for vehicle manouevres are derived from the present state of the

vehicle [11]. Localization and mapping, occlusion of objects, and dynamics of the environment allow the autonomous vehicle to predict future states and actions that facilitate tracking tasks based on features extracted and tracked over time [35]. Advanced neural network (NN) based architectures such as convolutional neural networks (CNN) and deep neural networks (DNN) have dominated computer-vision since the arrival of AlexNet [40]. Rather than completely relying on manually defined features, these NNs also learn increasingly complex features relevant to the task using function approximation [41]. These architectures were enhanced through expressivity and robust training to generalize and learn informative object representations in the autonomous driving domain. End-to-end learning for visual odometry using CNNs has led to results comparable to the state-of-the-art methods used for localization and mapping [28]. Deep learning has gained increased attention as their design requires minimal prior knowledge and the models can be fine-tuned to scale to different environments [10]. These models have been enhanced using recurrent neural networks (RNN) that memorize long-term dependencies and tackle autonomous driving as partially observable Markov decision processes (POMDP) [42]. This is a significant improvement over traditional methods such as Bayesian decision process based on Markov assumption [43]. POMDPs formulate the autonomous vehicle control problem as an optimization task, and rely on assumptions to optimize an objective [42]. The RL seems to be promising for planning and control aspects and scales to very complex environments and unexpected scenarios [44].

Recently, researchers have tried to divide the autonomous vehicle problem into sub-problems for categories such as object-detection, scene-segmentation, visual odometry and combine the results together [45]. However, the sub-problems might be more complicated than the autonomous driving task itself. For example, object detection using single shot multibox detector (SSD) in driving environment is redundant, as human drivers do not detect and classifiy all the objects; rather, they classify the most relevant objects [46], [35]. Moreover, the motion of the vehicle introduces a Doppler shift which causes dynamic contraction of the visual zone [47], [48]. Furthermore, the solutions to the isolated sub-problems

could be optimum, fine-tuned and well-solved, but might not integrate so as to result in a cohrerent solution. DRL addresses these issues by introducing a reward signal that correlates current driving and future planning to arrive at optimum driving [49]. The reward is positive for a correct/favourable action, and negative for incorrect/unfavourable/disastrous action [50]. In our driving scenario, a positive reward is associated with stable driving/not crashing and the reward diminishes for an unsafe maneuver. As the reward involves driving manouvres in a real-time driving environment, it is impractical to train a DRL system on a real vehicle [49], [51]. This limitation has been countered by training DRL autonomous vehicles using video games and other simulation engines [7]. Fig. 2.2 depicts the autonomous driving problem under consideration, solved using DRL. Based on a regulalrly updated reward function, the vehicuar agent learns to drive on a straight line path, without deviating from the trajectory [52]. The environment specifies the relevant driving conditions, such as the visible road, turns, and presence/absence of obstacles.

Considerable attempts have been made to solve autonomous-agent / multi-agent problem using value function approximation methods such as gradient descent, linear function approximation, and incremental prediction algorithm [42], [53]. However, in batch reinforcement learning, gradient descent is not sample efficient [42]. In order to arrive at the best-fitting value function, the least-squares algorithms calculate a parameter vector and minimize the sum-squared error between the obtained and target values. Deep Q-networks (DQN) use experience replay and fixed Q-targets and repeatedly re-evaluate experience replay with different target values [36].

2.3 Deep Reinforcement Learning

As a combination of DL and RL, deep reinforcement learning (DRL) does not require prior training on labelled data as in supervised learning, and tries to learn an optimal strategy from the environment through repeated corrective actions [11]. Based on deep Q-networks (DQN), considerable advances and breakthroughs have been achieved in the domain of gaming [42].

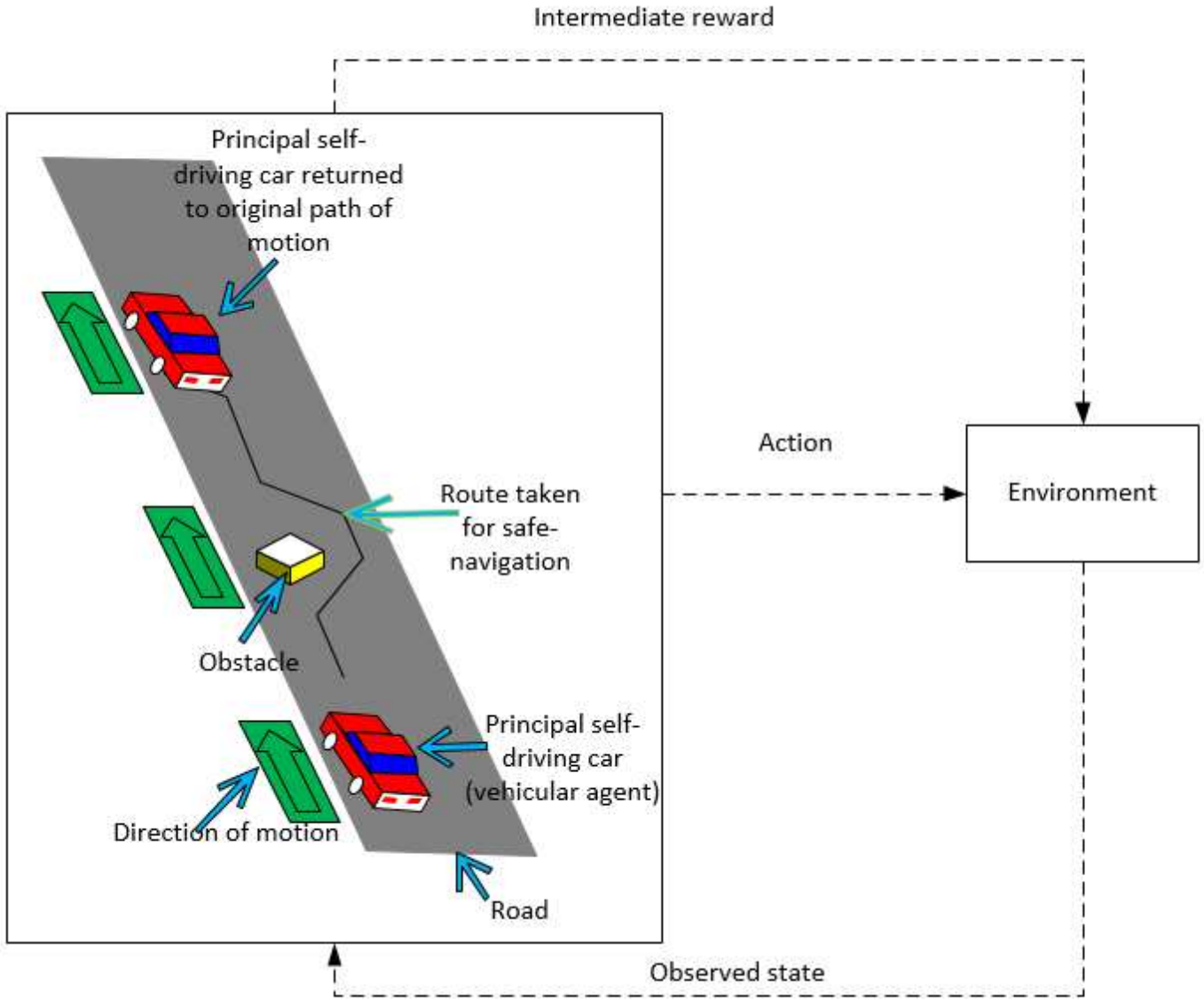


Figure 2.2: DRL for self-driving cars.

Many gaming applications utilize discrete asynchronous advance-actor critic (A3C) to train the agent [53]. However, it is impractical to switch steering, adjust speed, or take braking actions in binary levels comprising of two discrete values [49]. Realistically, the outcomes depending on continuous actions are more applicable to driving scenarios [12].

Reinforcement Learning (RL) is a type of ML algorithm in which agents take actions in an environment to maximize the cumulative reward. RL is gaining increasing acceptance in autonomous driving [23]. However, it generates unstable or divergent results, especially during nonlinear function approximation on image recognition and video processing using loss function, which is usually the Q-value [54]. The instability arises due to correlation in the sequence of observations [12]. Furthermore, the Q-value instability leads to significant variations in the policy for every small change in the Q-value [54]. The conventional schemes exploit the experience replay for solving such instability issues in deep Q-learning (DQL) [23]. Applying regularization and generalization between similar state inputs still leads to inefficient results while solving the following issues:

- The randomness of the experience replay is likely to lead to high errors [54], and
- These systematic errors might cause instability and divergence of learning [54].

To address these issues, neural network based techniques have been implemented in literature, that tune the adaptive size of experience replay to construct target values based on Q-learning algorithm [18], [19], [32], [33], and [36]. Although these techniques lead to exploration with better performance, the trade-off between the Q-value instability and divergence is not upper bounded and tends to vary asymmetrically [54].

DRL provides the ability to output continuous action using deep deterministic policy gradients (DDPG) and soft actor-critic (SAC) for driving behavior [55]. Both DDPG and SAC can be further enhanced when instead of being trained on raw input, they are trained on outputs obtained through variational autoencoders (VAE) with pre-defined loss functions [11]. For instance, the steering actions in a vehicle fluctuate a lot when the agent is trying to

maintain its position in a lane or while making a turn [49]. In this thesis, we investigate the applicability of VAE, DDPG, and SAC for autonomous driving. Moreover, we show that our approach using VAE, SAC, and DDPG reduces the learning time and leads to longer episodes of uninterrupted driving. As the existing inference frameworks pose challenges for learning optimal policies, the application of probabilistic models to RL leads to optimization through variational inference [53]. In self-driving vehicles, the absence of mode capturing behavior and approximation of pseudo-likelihood methods pose difficulties in learning deterministic policies [42]. The maximum-entropy RL based approaches and Bayesian probabilistic inference frameworks utilize a parameterized action-value function to estimate future states of an underlying Markov decision process (MDP) [11]. The mode-seeking approcah suffers from divergence, undermining the ability to learn deterministic optimal polices naturally. Action-state based inference along with the ability to optimize value-functions and policies in separate, iterative steps has led to variational expectation-maximization in actor-critic algorithms [56]. Our experiments demonstrate that actor-critic algorithms in conjunction with VAE outperform the state-of-the-art methods based solely on soft-actor critic or deterministic value-functions in self-driving domain. Autonomous driving problems modelled as MDPs introduce severe challenges due to critical scenarios arising from multiple objects in the vicinity of the vehicle [28]. In the existing literature, the driving environment is usually Rayleigh distributed [57]. Actor-critic methods have achieved incredible performance on RL problems such as games, but they have been prone to instability due to frequent interaction between the actor and critic during learning [21]. An inaccurate step taken at one stage might adversely affect the subsequent steps destabilizing the learning. To avoid such issues, rewards were introduced to regularize the learning objective of the actor by penalizing the error of the critic [58]. This improves stability, as large steps in the actor update are prevented when the critic is inaccurate [53].

2.3.1 Deep deterministic policy gradient (DDPG)

Fundamentally, deep deterministic policy gradient algorithm is an actor-critic based policy gradient algorithm that estimates a deterministic target policy and continuously improves the learning process [11]. It applies gradient descent to the policy with minibatch data sampled from a pool of possible actions [54]. Furthermore, to improve the stability of the algorithm, DDPG uses target networks for both the critic and the actor [11]. The current state acts as an input to the actor network and the output is a real value representing an action selected from a continuous action space. The critic network outputs the estimated Q-value of the current state and the action selected by the actor [9], [59]. For autonomous driving, DDPG was preferred in this thesis over DQN, as DQN is more suitable for applications with discrete action spaces.

2.3.2 Variational auto encoder (VAE)

In machine learning and deep learning, the process of reducing the number of features that describe the data is called dimensionality reduction [11]. This data reduction leads to low-dimensional data consisting of relatively fewer features than in the original dataset [54]. Although there exist a number of methods for dimensionality reduction, a popular framework consists of encoder-decoder, where an encoder produces a set of new features from the old features and the decoder reverses the encoding process [9]. This framework for dimensionality reduction makes use of data compression where the encoder compresses the data from the original environment space to the encoded space, whereas the decoder decompresses it [60]. If the compression is lossy and a part of the information is lost during the encoding process, it cannot be recovered during decoding [59].

In DRL, a novel technique for compression-decompresion is variational auto encoder (VAE), which is a combination of two neural networks [11]. Its input is a set of data that represent features, its output is a compressed representation, and it has weights and biases [9]. For example, let us consider an input image of a hand-written number having a size 28 $\times$

28 pixel. The encoder encodes the 784-dimensional data into a latent representation space much less than 784 dimensions [11]. Using deep learning, the encoder is trained to learn an efficient compression of the data into lower-dimensional space, without incurring loss of information or suitable features. The VAE encoder outputs parameters to an output space with Gaussian probability density and samples from this distribution to get noisy values of the dimensionally reduced features [11], [54]. The decoder comprises of another neural network that outputs the selected features mapped to the probability distribution of the data [60]. For the handwritten digit example, if the image is gray scale with each pixel represented as 0 or 1, then the probability distribution of a single pixel follows Bernoulli distribution [11], [43]. The decoder input is then the latent representation of the digit and it outputs 784 Bernoulli parameters, each corresponding to one of the 784 pixels in the image [10]. To ensure minimum loss of information, usually a large number of imput images are fed as input so the enconder selects the best possible features from each input image, reducing the loss probability [45], [59].

Although VAE are computationally slower in comparison to traditional encoders, yet VAE leads to more robust feature selection. These robust features may be critical deciding factors as an autonomous vehicle tries to determine the best action for a specific state [43], [59]. Moreover, a traditional autoencoder is an unsupervised learning technique, primarily used to compress and decompress an original input. This simplifies graphical representation or visualization of the input data, and also removes noise in the data [11]. A VAE, rather than compressing and decompressing an input dataset, explores the underlying probability distribution in the dataset to model the distribution of latent variables, leading to enhanced feature selection, in a manner akin to state representation learning (SRL) [59], [61].

2.3.3 Soft actor-critic (SAC)

The aim of DRL is to learn an environment in a short duration and then to generalize to conditions. The conditions may be unseen during training in non-simulation environments

[54]. Some of the most successful RL algorithms in recent years such as trust region policy optimization (TRPO), proximal policy optimization (PPO) and asynchronous actor-critic agents (A3C) suffer from sample inefficiency [27], [28]. This is due to the fact that these techniques enable learning in an on-policy manner and require new samples after each policy update [54]. Q-learning based off-policy methods such as DDPG learn efficiently from past samples using experience replay buffers [11]. However, such methods are sensitive to hyper-parameters and require a lot of tuning to converge [11]. Soft actor-critic (SAC) provides an alternative method to speed-up the convergence [9], [11], [59].

As SAC is defined for RL tasks involving continuous actions, it is applicable to autonomous driving problem [11]. Along with maximizing the lifetime rewards, SAC also aims to maximize the entropy of the policy. The term entropy is a measure of the unpredictability of a random variable [11]. If a random variable always takes a single value then it has zero entropy as it is not unpredictable at all. If a random variable can be any real number with equal probability then it is highly unpredictable and has very high entropy [60]. A high entropy in policy encourages exploration, and assigns equal probabilities to actions that have same or nearly equal Q-values [56]. It ensures that exploration does not collapse into repeatedly selecting a particular action leading to inconsistency in the approximated Q-function by assigning a high probability to any one action out of the possible set of actions [42].

Table 2.2 highlights some advantages offered by DDPG, VAE, and SAC to model autonomous vehicles.

Table 2.2: Requirements of autonomous vehicles and the scope of DDPG, VAE, SAC: Connection of theory to application.

Requirements of autonomous vehicles	**DDPG**	**VAE**	**SAC**
Effective learning rate	✓	✓	✓
Extension to other driving environments	✓	✓	✓
Tolerance to abrupt variations in driving conditions	✓	✓	✓
Support benchmarking for effective comparison	✓	✓	✓
Provides parametric probability distribution of drive terrain	×	✓	✓
Discrete action space	×	✓	×
Continuous action space	✓	✓	✓

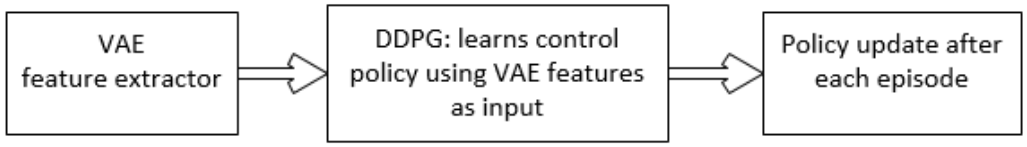
2.3.4 Integration of DDPG, SAC, and VAE

Combining the aforementioned approaches makes it possible to derive the benefits and strengths of VAE, DDPG, and SAC. The objective function then becomes to minimize the weighted sum of different loss functions [56]. This thesis implements a feature extractor variational auto encoder (VAE), to compress the images captured during driving to a lower dimensional space [62]. The weights that provide gradients with similar magnitudes indicate that each feature has been kept relevant. The first step of extracting the relevant information from raw data is done by VAE by compressing the search space. This also accelerates the training in later stages by learning the control policy from the lower dimensional image space [63], [61].

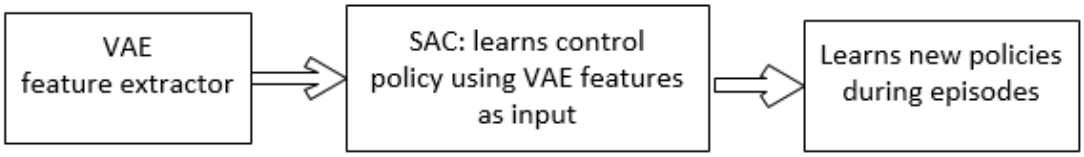
The second step after the features have been extracted is to use a DRL algorithm. This thesis investigates DDPG and SAC algorithms. The DDPG policy gradient algorithm learns a control policy using VAE features as input and the policy is updated after each episode [9]. A distinguishing feature offered by DDPG is the reply buffer, which is memory to store the interactions with the environment. These interactions can be played when needed at a later time, so that the self-driving car can update the policy without explicitly interacting with the environment in real-time again [50].

In our experiments, the vehicle is trained to maximize the distance travelled before it steers off the track. The episode ends as soon as the vehicle steers off the road. The episode termination also prevents the vehicular agent from exploring regions that do not contribute at all to effectively learn the driving task [64]. If a VAE feature extractor is trained after each episode, the distribution of features is not stationary. As the features change over time, this introduces instabilities in the learning policy [65]. Moreover, on low power CPU machines, traning a VAE after each episode is time consuming and a slow process. To address these issues, in this work, a VAE is pre-trained and a fixed set of features are collected beforehand [62]. Next, these features are provided as input to DDPG to learn and update the policy. Also, to speed up the process, we trained feature extractors using

Google Colab notebook. Lastly, the DDPG algorithm is known to be inherently unstable in cases where its performance degrades during training and fails to tune if there are multiple factors that affect the learning outcome [11], [63]. The soft actor critic (SAC) algorithm that provides much stable performances and is easier to tune in case of multiple parameters is applied and its performance comparison with DDPG is analyzed. Fig. 2.4 represents the steps executed in 'VAE+DDPG' approach and 'VAE+SAC' approach. Fig. 2.3 highlights key differences the features offered by aforementioned methods [59], [61].



(a) VAE + DDPG approach



(b) VAE + SAC approach

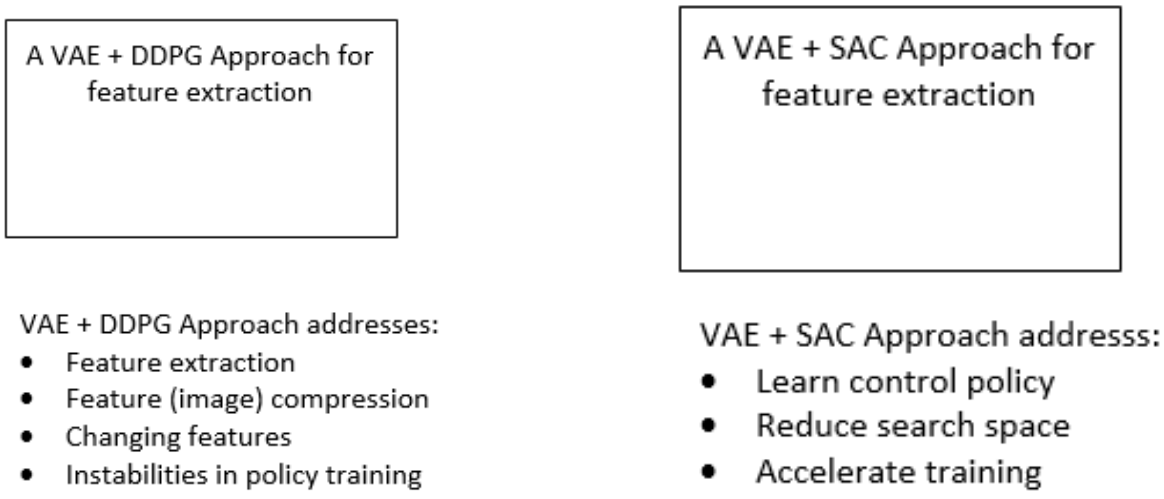
Figure 2.3: Block diagrams representing VAE + DDPG and VAE + SAC approaches.

2.3.5 Kullback-Liebler divergence between the vehicle states at different timestamps

The total policy loss is calculated as per equation 2.1.

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{DDPG} + \mathcal{L}_{SAC}. \tag{2.1}$$

The term $\mathcal{L}_{prior} = D_{KL}(q(z|\hat{\sigma}_{0,t})||p(z))$ is the Kullback-Liebler divergence between the states at different timestamps. The SAC and VAE follow the $q(z|\hat{\sigma}_{0,t})$ distribution of output states. The prior distribution for DDPG is $p(z)$, which is Gaussian in the range $\mathcal{N}(0,1)$ and is used to optimize the VAE regularizer. During the training, we have $z = \mu + \epsilon\sigma$ and at test time



(a) VAE + DDPG approach key features

(b) VAE + SAC approach key features

Figure 2.4: Comparison of VAE in conjunction with DDPG and SAC algorithms.

$z = \mu$, where μ and σ are outputs of the VAE-SAC network and ϵ is a Gaussian random vector with the same dimension as μ and σ. $\mathcal{L}_{DDPG}$ is an error calculated using the hidden activations of the l-th layer of the DDPG network [7], [42], [53].

2.4 Problem of Scene Perception and Decision Making in Autonomous Driving

The following questions pertaining to scene perception, environment cognition, and decision making are investigated in this thesis:

1. To generate realistic and safe autonomous driving using DRL and related techniques [11].

2. To improve the autonomous driving policies based on continuous state space, continuous action space, and pixel space [57].

3. To learn the driving behavior and environmental conditions without manual input,

using DRL [11].

4. To solve the inaccuracies, improving loss function, entropy, policy, loss, and learning rate using improved VAE, DDPG, and SAE [53].

5. To maintain the trade-off between policy loss and learning rate [43].

2.5 Proposed Solutions

Autonomous driving is traditionally modelled as a multi-objective control problem with high-dimensional feature space, agent (vehicle) states, and a mono-dimensional discrete action space. We use a VAE to map the vehicle state at a given time and the dynamics of the environment not directly influenced by the agent. We repeat this procedure iteratively in a semi-batch approach to bootstrap the algorithm, starting from a fully random exploration of the environment.

Chapter 3

System Model

This chapter gives a formal description of the method and its core components, and then discusses the technical details of implementation. This chapter also discusses the system model used in this work as well as the rationale behind some of the assumptions made in the analysis.

3.1 Driving Environment and Scenarios

In order to minimize the driving imperfections and degree of randomness of self-driving cars, a simulated driving environment should closely resemble a realistic driving environment [36]. The environment should comprise scenarios that encompass the steadily growing number of functions autonomous driving requires to execute. To achieve this with utmost efficiency, a scenario-based evaluation of driving environment is necessary [37]. The scenarios comprise different traffic scenarios on a wide range of road networks and different driving conditions. A large number of scenario variations and randomized simulation of millions of test kilometers, test parameters and result analysis have led to significant development phases [46]. The automated driving toolbox available in Matlab opens up powerful and efficient possibilities for realistic simulations of autonomous driving including graphical definition of road networks and traffic scenarios comprising a multitude of road users [23]. Some of the features critical

to autonomous driving are:

- Automated driving systems perceive the environment to detect objects surrounding the vehicle
- Visualize vehicle sensor data
- Detect and verify objects in images
- Fuse and track multiple object detections

This chapter presents a mathematical description of a basic driving environment comprising of a road, a vehicle and an obstacle. The mathematical model also outlines the conditions for no collision between the autonomous vehicle and the obstalce, and for safe navigation. The chapter also presents a few driving scenarios based on the driving environment, and delves on one particular scenario that is studied in this thesis later.

3.1.1 Mathematical definition of the driving environment

The environment under consideration consists of a road which describes a driving path or trajectory, a navigating vehicle, and an obstacle (optional).

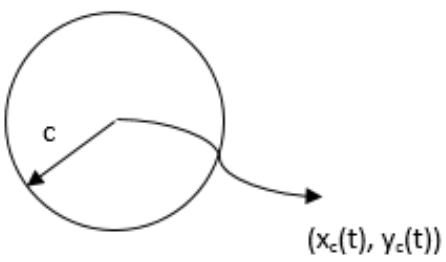
1. The road is defined as follows [66]:

$$\alpha = \{(x_1, y_1 \in \mathbb{R} \times \mathbb{R}) \mid y_1 \text{ - } mx_1 \text{ - } c_R \geq 0\} \tag{3.1}$$

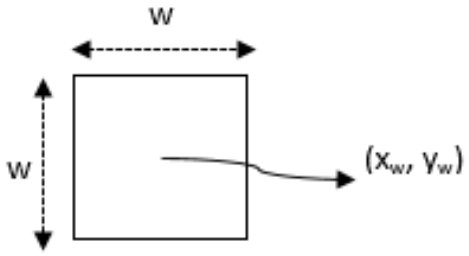
2. In Fig. 3.1, the vehicle is represented as a circle and the obstacle is represented as a square [66], [67].

$$\beta_{veh} = \{(x_2, y_2 \in \mathbb{R} \times \mathbb{R}) \mid y_2 \text{ - } mx_2 \text{ - } \mathrm{c}_{veh} \geq 0\}, \tag{3.2}$$

   where β_{veh} is the initial set of points of the vehicle when placed on the road and c_{veh} is the y-intercept made by the vehicle while travelling in a straight line path.



(a) The vehicle is represented by a circle with centre at $x_c(t), y_c(t)$, and radius c.



(b) Obstacle represented by a square with centre at (x_0, y_0), width w and length w.

Figure 3.1: System model.

3. Equation (3.3) indicates that the vehicle remains on the road at all times [67], where veh_{road} indicates vehicle remains on the road and does not drift off-track.

$$veh_{road} = \alpha \cap \beta_{veh} \tag{3.3}$$

4. The following equation approximates no collision condition. If there are k number of vehicles on the road defined by α, the radius of each vehicle being c_i and c_{i+1} respectively, then [68], [69]

$$\alpha_c^{t_i} \cap veh_{road} = \alpha_c^{t_i} = \frac{k-1}{\sum_{i=0}^{k-2}(|c_{i+1}-c_i|)} \tag{3.4}$$

Here, $\alpha_c^{t_i}$ indicates that a vehicle of radius c is present on the road α at time t_i. In the scenario investigated in this thesis, there is only one vehicle on the road, i.e. $k = 1$, implying that there is no probability of a collision with another vehicle. Eqn. (3.4) also implies that the car remains on the road, the continuous action-space [68], [69].

5. The obstacle is defined as [66]

$$\beta_{obs} = \{(x_3, y_3 \in \mathbb{R} \times \mathbb{R}) \mid x_3 \cos\theta_0 - y_3 \sin\theta_0\} \tag{3.5}$$

and

$$-w/2 \leq x_3 - x_o \leq w/2$$

$$-w/2 \leq y_3 - y_o \leq w/2$$

where θ_0 denotes rotation by a specific angle, and x_o, y_o are the set of points representing the centre of the obstacle at time t [66].

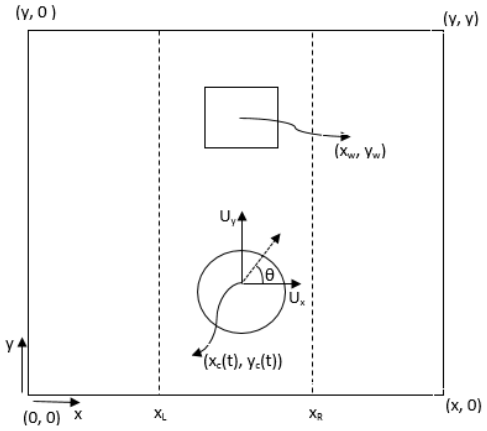


Figure 3.2: Representation of safe navigation scenario around a turn or obstacle [3], [4].

6. The objective of the experiment is to maintain the lane position at all times, and not to drift off the track while taking a turn [29]. This is represented as

$$\min_{<V_r,S>} || \mathbf{V_r} ||, \tag{3.6}$$

where $\mathbf{V_r}$ represents the parameter values, i.e. the set of velocities in a given timestep, at present time and past instances for a given state S.

$$\mathbf{V_r} = \{V_r(t_0), V_r(t_1), ...V_r(t_{N-1})\} \tag{3.7}$$

where N indicates the Nth time frame. All past and present states are in the continuous state space [70].

7. The trajectory followed by the car is described by [64], [66]:

$$x_c(t_i) = x_c(t_{i-1}) + v_x(t_{i-1})\Delta\text{t} + \tfrac{1}{2}\ v_x(t_{i-1}) - v_x(t_i)\Delta\text{t}, \forall i \tag{3.8}$$

$$y_c(t_i) = y_c(t_{i-1}) + v_y(t_{i-1})\Delta\text{t} + \tfrac{1}{2}\ v_y(t_{i-1}) - v_y(t_i)\Delta\text{t},\ \forall i, i = 0, 1, 2.....N-1, t_i = i\Delta\text{t} \tag{3.9}$$

where Δt is the difference between two subsequent timeframes while the vehicle navigates the trajectory. These parameters are later used to calculate the optimal value function $v_\pi^*(s)$ and optimal Q-value $Q^*(s, a)$.

8. The vehicle is said to successfully navigate without hitting the obstacle if:

$$\alpha_c^{t_i} \cap \beta_{obs} = \phi, \tag{3.10}$$

where β_{obs} indicates the presence of other vehicles (obstacles) on the road. In our scenario, β_{obs} is set to zero to simplify the environment and to reduce the number of factors contributing towards problem complexity [67], [71].

Eqn (3.10) is defined over the continuous state-space and action-space. This implies no collision with the obstacle. The car is required to be parallel to the slope of the road and the road is deviod of any sharp turns, implying that the car remains within the road. The above architecture is depicted in Fig. 3.3 [3].

3.1.2 Autonomous driving scenarios

This section introduces the autonomous vehicle landscape. The scenarios highlight multiple issues that require resolution. The scenarios are depicted in an evolutionary hierarchy, with incremental additions to complexities and capabilities in each scenario. The scenarios outline the driving dynamics for safe operations on the road and assist with formulation of a clear solution strategy and co-operation between autonomous vehicles. The scenarios help to analyze the incoming data on the driving environment

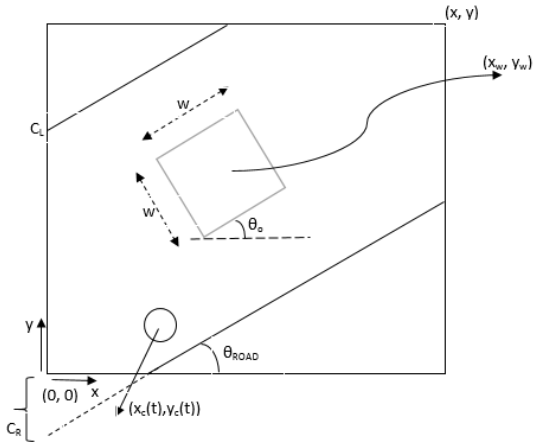


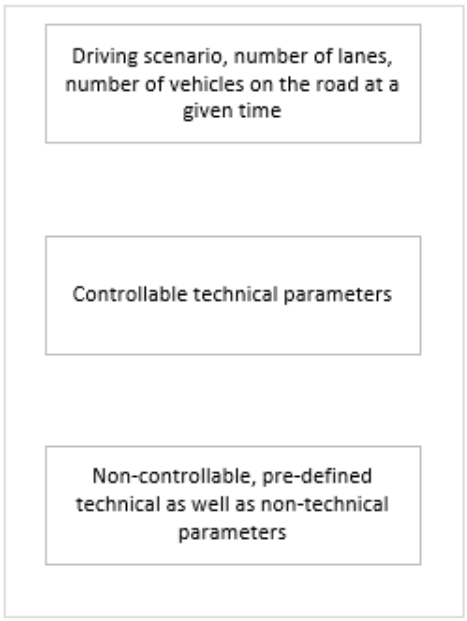
Figure 3.3: Representation of the combined scenario of vehicle and obstacle at a slant to the road [3], [4].

and use deep learning to recognize possible action space and learn to react accordingly in the chaotic environment through extensive testing and simulation [5].

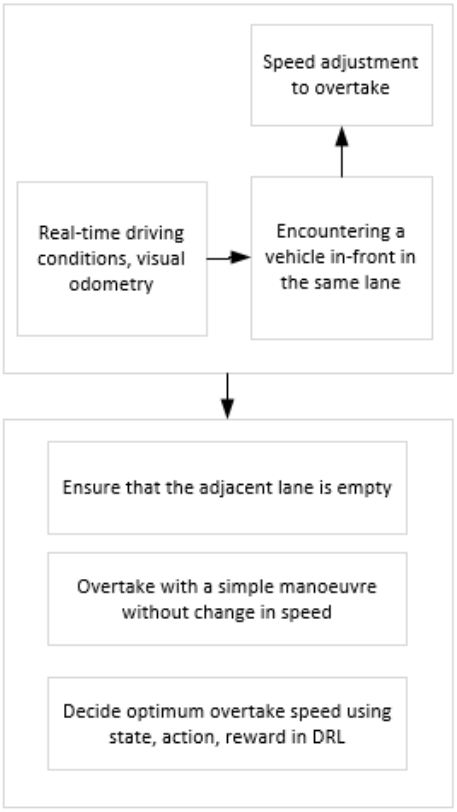
(a) **Scenario 1**: Autonomous driving on an empty road:- This approach has been studied by [36] and the problem is modelled using proportional controller. For driving in known environment, a conventional path planning algorithm is proposed in [46]. The environemnt information is obtained in real-time with maps, and the driving problem is formulated as LQG (linear–quadratic–Gaussian) and solved using linear–quadratic regulator (LQR), a feedback controller [46].

(b) **Scenario 2**: Environment with dynamic obstacles:- In a dynamic environment, each vehicle is assumed to be moving without any knowledge of its next movement, hence it cannot be modeled, rather, optimal control is applied [33]. In this scenario, LQR is applicable as long as the cost function is in the optimum range [33].

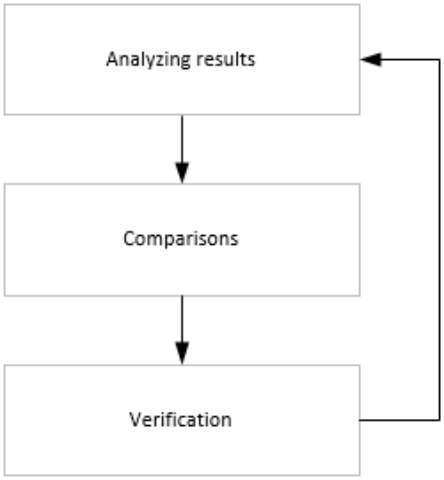
(c) **Scenario 3**: The human-driving data is recorded and the autonomous vehicle is trained to imitate the same:- This resembles supervised learning and has been crtiqued by [30] where the authors have reasoned the unethical human-driving instances and the implications of such behaviour being learnt/imitated by the



(a) Gathering state and environment information.



(b) DRL for autonomous driving manouevres (actions).



(c) Adjusting actions based on reward function.

Figure 3.4: High level representation of the system architecture.

autonomous vehicles.

(d) **Scenario 4**: The system is allowed to learn by itself how to take the decision, which is known as RL:- Beginning randomly, the model is trained to learn how to take better decisions over repeated attempts, reducing errors based on a reward function [72]. After every action, the vehicular agent transitions from one state to another. The execution of a set of actions in a specific state evaluates the suitability of the action in terms of reward [29]. The Q-learning technique aims at learning a policy, which guides an agent to take an action with highest reward [33]. Among various driving scenarios encountered, this work focuses on maintaining the vehicle's position in a specific lane. The experiments are conducted on the lane-keeping scenario, without deviating off the track.

Out of the four scenarios described above, the work in this thesis considers the description given by scenario 4. This is due to the following reasons:

- This scenario with one vehicle trying to maintain its position on a driving trajectory (road/lane) is a sufficiently complex problem to be solved using DRL [73].
- The problem in the scenario can be modelled using MDP/POMDP and can be solved using DRL [22].
- The scenario is neither too simple to ignore underlying complexities and challenges in scene perception and vision, nor is too complex to include a large number of factors that affect the vehicle's state-action-reward cycle.
- Significant work has been done on the preceding scenarios 1, 2, and 3 [30], [33], [46]. However, more challenging scenarios are being incresingly adopted with advanced solution mechanisms finding favourable application to solve autonomous driving problems, as outlined in section 1.2.

3.2 Problem Formulation

The vehicle needs to be trained to do the following tasks:

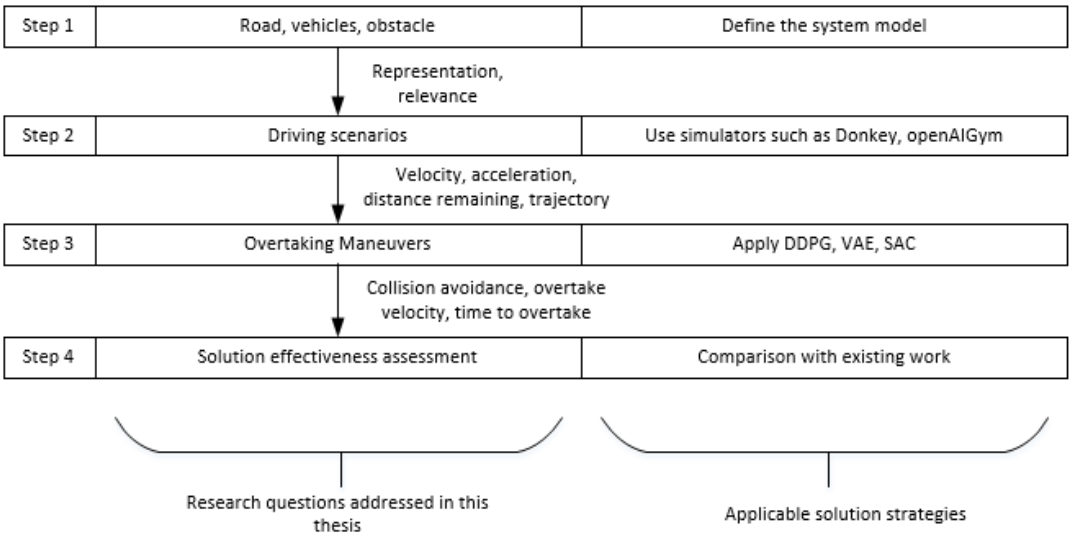


Figure 3.5: Proposed research questions and applicable solution strategies.

- Recognize its environment (lane detection, traffic sign recognition etc.).
- Keep track of the vehicle's state with respect to the environment over a given time-frame.
- Maintain self-localization.
- Plan its actions based on observations.

In the existing literature, simulation based experiments have considered the reward function based on how much time it takes before the user intervenes to take control of the vehicle. The deviation from a trajectory is detected either using data from other sensors, especially in case of other vehicles present on the road, or using Global Positioning System (GPS). The GPS data may also be used to detect when the car needs to decelerate, for instance, if there is an approaching roundabout, a sharp turn, multiple roads converging, a traffic signal, or a stop sign. In our work, the reward

function is based on how and when the autonomous vehicle terminates the driving action and returns to the beginning.

A deep Q-learning network trained on one type of environment can theoretically be extended to another type of environment by training the parameters in the last layer of the network [23]. The CNN in deep Q-learning network facilitates this transfer learning instead of re-training the whole network each time the environment changes. In this thesis, the parameters used in the simulations train the DRL network from the beginning, and then generate image logs in a given time-frame to subsequently train the network based on the safe/unsafe driving actions [49].

As DRL networks automatically adjust and improve their structure in accordance with the reward function and policy loss collected intermittently, a little prior knowledge is sufficient to learn the driving behavior. Based on the driving behavior learned by DRL, an additional algorithm needs to be developed to optimize the reward function and policy loss according to the most recently correlated historical data [12]. Figure 3.4 represents a high-level system architecture where the autonomus vehicle is required to perform the following tasks:

(a) The vehicle tries to navigate in a straight-line driving environment.

(b) In second case, the vehicle learns to safely maneuver a turn.

(c) To visually detect the environment infront.

3.3 Solution Approaches: Preliminaries

Figure 3.6 highlights the problem formulation and the proposed solution approach. We believe this is the first work to show that DRL is a viable approach to implement VAE, DDPG, and SAC to model an autonomous driving maneuver that also integrates Markov decision processes (MDP), planning by dynamic processes, model-free predic-

tions, and policy gradient methods. Fig. 3.6 describes the preiliminary groundwork that facilitates application of reinforcement learning (RL) to model an autonomous vehicle. The initial state of an autonomous vehicle needs to be updated to best adapt to the current state of the driving environment. This involves constant adjustment of speed, acceleration, and steering angle so that the vehicle trajectory is aligned with the road/lane trajectory. The alignment of trajectories for extended time frames of vehicle navigation can be a significant indicator that the vehicle is able to follow a lane and does not deviate off the track. To simplify the problem, we assume that the vehicle has an unobstructed and unoccluded visual access to the driving environment, so that any curves, turns, and obstacles in the vicinity are clearly visible to the vehicle. Once the vehcile is past the initial states, the subsequent iterations take actions to arrive at next states. The suitability of next states is governed by a reward function, that prompts the vehicle to either reinforce the action mechanism, or to avoid an action pattern for that environment. This thesis proposes the formulation of autonomus vehicle navigation problem as an MDP and is solved using DRL to evaluate optimal policy and reward function. The specific DRL approaches used are:

- **VAE** for effcient driving environment analysis,
- **DDPG** for optimal policy calculation and,
- **SAC** for faster arrival at optimal policy, without having to repeatedly train the DRL network for all the timeframes.

3.3.1 Reward shaping

Reward function refers to the feedback obtained from the environment to evaluate the viability of the actions taken. In DRL, a reward function directly influences the behavior adopted by an an agent. An unreasonable reward function may result in unexcepted action, or cause divergence during training. In autonomous driving, a

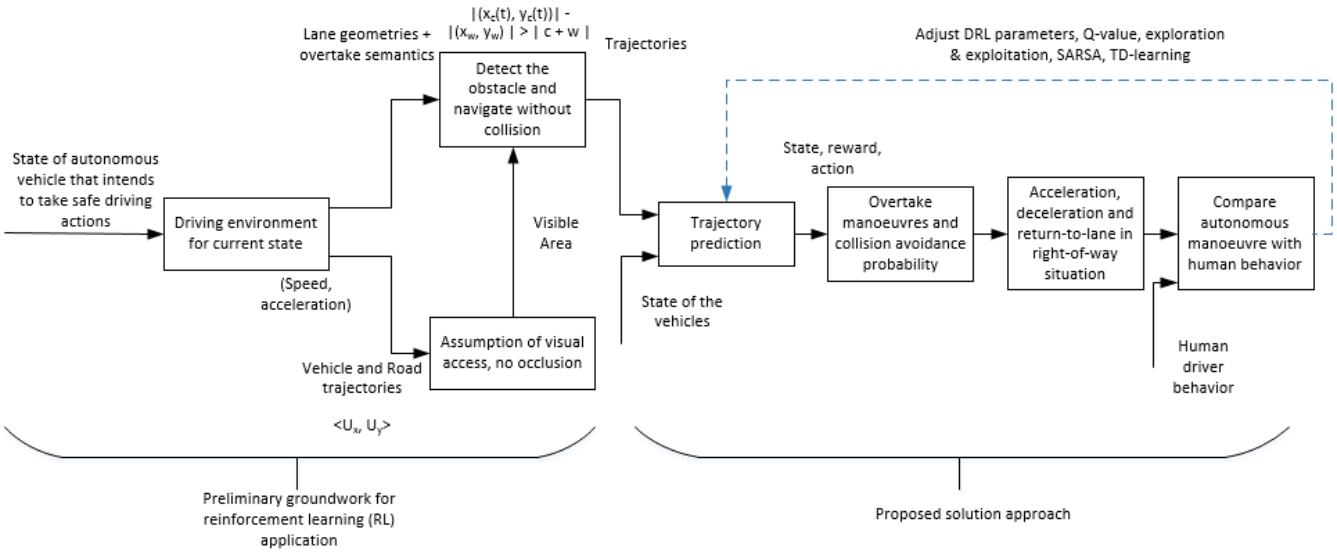


Figure 3.6: Proposed solution approach.

reward function is usually formulated as a linear model that impacts the velovity of the car v, the angle between the road and car's heading θ, and the distance from the middle of the road d. To improve the vehicular agent's actions, the penalty p prevents the agent from deviating off the track, and the self-driving agent gradually aims at achieving human-level driving. Reasonably-safe human drivers tend to keep a relatively fast speed on the straight and slow down at the turning, and try to maintain a smooth steering control at all times. The reward for an action that allows the vehicle to maintain its position on the road is given by [74], [75] (3.11), where the symbols have their usual meaning defined at the beginning of the thesis.

$$r = v(\cos\theta - d) \tag{3.11}$$

This thesis uses an extra penalty as $\Psi * |s_{t+1} - s_t|$, where s_t means the steering action at time step t and Ψ presents the corresponding constant empirical coefficient. The experimental results show better smoothness with the new penalty and the whole

reward function is given as [74], [75]:

$$r = v(\cos\theta - d - \Psi * |s_{t+1} - s_t|) \tag{3.12}$$

In this thesis, we set Ψ to 2 and 3 as driving smoothness tends to reduce with increase in throttle value. This fact conforms to human driving behavior where the faster the car runs, the harder it is to control it, indicated by failure to recognize turn/curve at high speeds. Equation 3.12 is selected over Eq. 3.11 as the application of DRL to autonomous driving intends to solve a set of sequential decision problems. Based on the action selection, the driving environment presents a sequences of states, and the vehicular agent receives rewards at each time step as immediate feedbacks. The eventual goal of the vehicle is to solve the sequential decision problem to derive a policy that maps vehicle's states to actions such as to maximize the sum of the received rewards. Equation 3.12 introduces reward shaping that receives gradual feedback and lets the vehicle know if it is getting closer to the goal at a slightly faster convergence rate [74].

3.3.2 Termination condition

References discuss the influence of different termination criteria in autonomous vehicles. This thesis terminates an episode when the vehicle seems to deviate off the track and the environment is reset if any of the following termination conditions is met:

- If the vehicle deviates off the track, i.e. moves on to the other lane, or throws itself off the edge.
- If the vehicle trudges in wrong direction, i.e. running in a direction other than the specified straight line motion.

3.3.3 Markov decision process

The vehicle is represented as an agent which receives sensory inputs and consequently performs driving actions (manoeuvres) in an environment. The vehicular agent acts on rewards, penalties, and policy losses from the environment with the goal to maximise the rewards it receives, and to minimise the losses [73]. A commonly used machine learning (ML) technique for choosing actions that minimize losses and trains agents to behave optimally is the Bayesian decision theory (BDT). BDT deals with the problem of making optimal decisions or actions that minimize an expected loss. In DRL, the agent learns an action based on the policy function, loss function and state-model. A state S_t is Markov if and only if [76], [77]

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, ..., S_t], \tag{3.13}$$

where $\mathbb{P}$ defines the probability distribution of the variable under consideration. A future action is independent of the past actions and depends on the present state and actions. The state transition matrix P defines transition probabilities from all states s to all successor states s'

$$P_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s] \tag{3.14}$$

A Markov reward process is defined as a tuple $< S, A, P, R, \gamma >$ [76], [77] where

- S is a finite set of states
- A is a finite set of actions
- P is a state transition probability matrix,

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$

- R is a reward function,

$$R_s^a = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$$

- γ is a discount factor, $\gamma \in [0, 1]$

The return G_t is given as [76], [77]

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The state value function $v_\pi(s)$ is given as [76], [77]

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

The action value function $q_\pi(s, a)$ is given as

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

The policy π is given as [76], [77]

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

The partially observable Markov decision process (POMDP) is an MDP with hidden states based on optimal function, Bellman expectation equation and Bellman optimal equation. In this thesis, the autonomous driving problem is formulated as a MDP, on the lines of the work of [73]. The MDP is then solved using policy based DRL mechanisms of DDPG and SAC, accompanied by VAE at appropriate solution stages [76], [77].

3.3.4 Planning by dynamic programming

The dynamic programming (DP) approach modifies the policy evaluation into iterative policy evaluation and improvement. Value evaluation is achieved from a policy $\pi(a|s)$ that leads to an optimal value from state s, $v_\pi(s) = v_*(s)$ (the principle of optimality), if and only if for any state s' reachable from s, π achieves the optimal value from state s', $v_\pi(s') = v_*(s')$ [76], [77].

The DP algorithms may be asynchronous or synchronous with extensions to approximate the value function. Figure 3.7 depicts the solution approach used in this thesis in form of flowchart.

3.4 Solution Approach

The proposed solution using VAE, DDPG, and SAC aims to use reward-function based DRL to learn policies for multiple state-action-reward tuples. The solution enables the optimal policy to generalize, and can be used with continuous actions for states that are yet to be traversed by the vehicular agent. The solution approach is described as follows:

- Let's say the vehicular agent has a choice of taking one of k possible actions $a_1 \dots a_k$.
- Assume that the environment can be in one of m different states $s_1, \dots, s_m$.
- Upon taking an action a_i in the environment in state s_j the vehicle incurs a loss ℓ_{ij}.
- Given the observed data D and prior background knowledge B, the vehicular agent's beliefs about the state of the driving environment are denoted by $p(s|D, B)$.

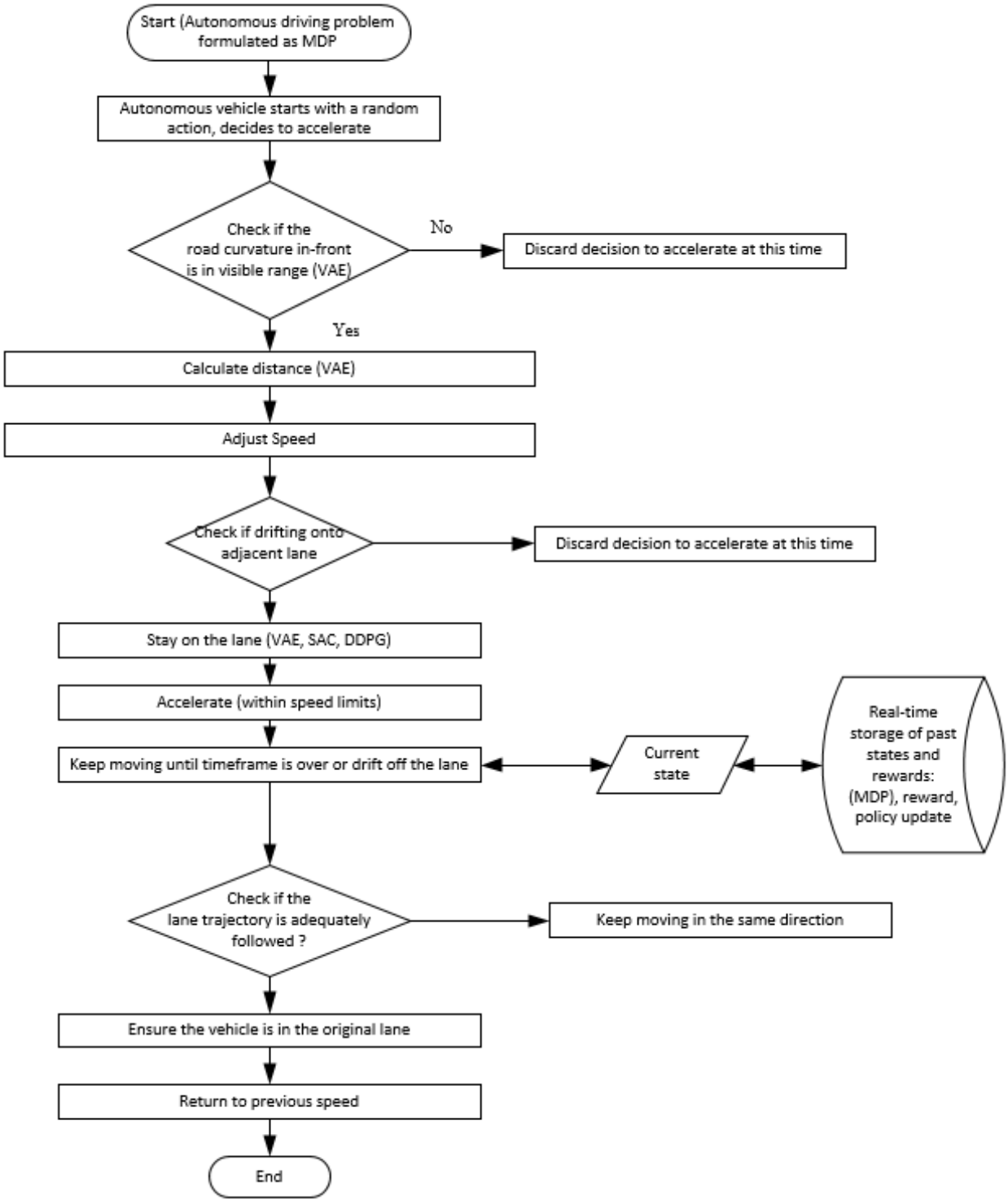


Figure 3.7: Flowchart representing the proposed scheme.

- The optimal action is the one which is expected to minimize loss and maximize utility.

$$a^* = \sum_{j=1}^{m} \ell_{ij}\, p(s_j^d|,) \tag{3.15}$$

The vehicular agent at a given time in a given state has two possible actions to choose from:

(a) ac_1 : accelerate, and

(b) ac_2 : don't accelerate.

In an ideal scenario, if acceleration would lead to an unsafe action such as drifting off the track, or deviating on to the other lane intended for oncoming vehicles, there are two probable outcomes:

(a) s_1^d : safe driving, and

(b) s_2^d : unsafe driving.

Optimal action: The vehicular agent needs to arrive at an optimal action for this decision problem. Based on MDP, the following variables are defined for the vehicular agent:

- States: s_t
- Actions: a_t
- Rewards: r_t

The variable s_t defines the state of the driving environment and the vehicular agent at time t. The vehicular agent takes action a_t and receives reward or penalty r_t . The

reward is assumed to depend on the state and the action. The optimal policy is defined by the eq. (3.16).

$$\pi^* = \arg\min_{\pi} \mathbb{E}[R_t|\pi] \tag{3.16}$$

The initial states are depicted in Table 3.1. Under safe driving condition, if the probability of acceleration is very high, then the loss is set to a high negative value, indicating a positive reward for that action. Under unsafe driving condition, if the vehicle decides to accelerate, the associated loss is a positive value, indicating negative reward, thus prompting the vehicle to refrain from taking the accelerating action. Similarly, if the vehicle is driving in a safe state and decides not to accelerate, the reward remains neutral, indicated by zero loss. Vice versa, if the vehicle is in an unsafe state and decides not to accelerate, although the action will not deteriorate the vehicle's state further, it will not guarantee a return to safe driving either. This is also implied through a zero reward.

Table 3.1: The initial values for states and the transition probabilities.

$P(s_1^d\|ac_1) = 0.99995$	$\ell_{11} = -10000$
$P(s_2^d\|ac_1) = 0.00005$	$\ell_{12} = +0.9$
$P(s_1^d\|ac_2) = 0$	$\ell_{21} = 0$
$P(s_2^d\|ac_2) = 1$	$\ell_{22} = 0$

As per the Markov property in MDP, the driving environment is characterized by the following transition probabilities [76], [77]:

$$P(s_{t+1}, r_t|s_t, a_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots) = P(s_{t+1}, r_t|s_t, a_t) \tag{3.17}$$

The expected rewards are [76], [77]:

$$\mathrm{R}^{a}_{ss'} = [r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'] \tag{3.18}$$

The vehicular agent is characterized by the policy [76], [77]:

$$\pi(s, a) = P(a_t = a|s_t = s) \tag{3.19}$$

An important assumption in this work is that the action at time t is only dependent on the state at time t and not on the previous states [76], [77]. This is an underlying principle of MDPs, that enables vehicular agents to choose actions a_t so as to maximize the sum of discounted future rewards R_t [76], [77]. As per the Markov property of conditional independence, future rewards and states are independent of past rewards, actions, and states given s_t and a_t [76], [77]:

$$\mathrm{P}(\mathrm{s}_{t+1}, r_{t+1}, s_{t+2}, \ldots|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) \;=\; P(s_{t+1}, r_{t+1}, s_{t+2}, \ldots|s_t, a_t) \tag{3.20}$$

If s_t is known, the expected value of the return R_t depends only on a_t, so previous states and actions become irrelevant. This approach is similar to nonparametric regression, ANN, and k-nearest neighbor, which provide the average of the recent outputs as the most likely driving behavior under the current scenario [76], [77]. State-action value function defines that under a given state, how favourable it is to take a given action based on a policy π [76], [77].

3.4.1 Optimal policies and values

Optimal policy is defined as π^* such that:

$$\mathrm{V}^{\pi^*}(s) \geq V^{\pi}(s)\ , \forall s. \tag{3.21}$$

Although there may be more than one optimal policy, in autonomous driving situations, it is imperative to determine if there exists at least one optimal policy for a specific driving environment. The optimal state value function is given as [76], [77]:

$$\mathrm{V}^*(s) = \max_\pi V^\pi(s) \ , \forall s. \tag{3.22}$$

The optimal state-action value function is given as:

$$\mathrm{Q}^*(s,a) = \max_\pi Q^\pi(s,a) \ , \forall s. \tag{3.23}$$

3.4.2 Solving MDPs using Bellman expectation equations

The policy gradient DRL methods and the actor-critic methods can be solved using Bellman equations mentioned below. The parameterized policy defines how the autonomous vehicle selects its actions and the critic appraises each action taken by the vehicular agent in the driving environment [73]. The appraisal is associated with a positive or negative reward function according to which the parameters of the actor are updated. Similarly, the actor's parameters can be updated with a policy gradient that does not necessarily have a critic component. The policy gradient methods such as DDPG adjust the policy parameters based on a sampled reward measured against a baseline value [76], [77]. In DDPG, this baseline is a stationary value that does not update with experience. In SAC, a baseline is estimated from experience, making the method an actor-critic method where the vehicle updates its parameters after each step taken in the driving environment [76], [77]. The Bellman equations allow comparing the results of taking a different action in each state and assist in negating the wrong step, causing the agent to strengthen how it selects the apparent best action. In SAC, the critic components make the gradient point in the apparent best direction without sampling other actions [23], [54].

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s')) \tag{3.24}$$

$$q_\pi(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s', a')) \tag{3.25}$$

Bellman optimality equation is given as [76], [77]:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_*(s')) \tag{3.26}$$

$$q_*(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a')) \tag{3.27}$$

Given the optimal value function, V^*, a popular technique to get optimal policy π^* is the greedy algorithm, V^*. This algorithm specifies that if a vehicular agent has V^*, the actions that appear best after a one-step search will be optimal. V^* turns a long-term reward into a measurable quantity which is locally and immediately available. Q^* is used to get the optimal policy [76], [77].

$$\pi^*(s, a) = 0 \;\; \forall a \;\; s.t. \;\; Q^*(s, a) \neq \max_{a'} Q^*(s, a') \tag{3.28}$$

Lastly, the policy improvement theorem justifies an optimal action in a given state

[76], [77].

$$Q^{\pi}(s, \pi'(s)) \ \geq \ V^{\pi}(s) \, \forall s \ \implies \ V^{\pi'}(s) \ \geq \ V^{\pi}(s)\text{`} \tag{3.29}$$

Chapter 4

Simulation Results and Analysis

This chapter presents the experimental setup and simulations results. There are certain assumptions regarding the drive-terrain and the length for which a self-driving vehicle drives before restarting the driving cycle. These assumptions apply both to the existing models as well the algorithms proposed in this thesis. The next section presents the experimental setup using DonKey simulator. Then, the simulation parameters are defined. The following section presents the performance analysis for VAE+DDPG and VAE+SAC schemes followed by summary of the findings.

4.1 Experimental Setup

In this thesis, the experiments are carried out on the Ubuntu operating system, DonKey simulator, OpenAIgym, and Google Collaboratory. The implementation addresses the need for improvements to the solutions already available, and investigates the effect of deep reinforcement learning (DRL) on the performance of self-driving cars and proposes a scheme for its enhancement. Due to the property of learning by trial and error in DRL, simulator plays an important role. Among various car simulators, the open racing car simulator (TORCS) is widely used in DRL research, providing both front view

images and extracted features. TORCS requires less hardware performance whereas DonKey provides flexibility and realism, hence we replcaed TORCS by DonKey.

In this work, we first define the drive terrain in continuous control domain and propose a penalty in reward shaping. Then, we test the proposed penalty with DDPG, VAE, and SAC, the state-of-the-art algorithms in continuous control domain to present a novel training strategy. The built-in specifications of the DonKey simulator allow the driving data to be captured at a rate of 20 Hz. In the simulation time over 7 hours of driving, the data was divided over three specific categories of driving terrains. The captured frames are 160×320 pixels in the region depicting the vicinity of the middle of the road. Example data captured by the simulator's built-in virtual sensors are the car speed, steering angle, total timesteps, and value-loss. We primarily focus on the camera frames, steering angle, speed, policy entropy, policy loss, and total timesteps. We also record the timestamps at which these parameters were measured and the timestamps the camera frames were captured. The raw sensor-data and the camera frames files are easy to use in DonKey simulator with the help of Docker, a container management software. We pre-processed the camera frames by downsampling them to 40×120 and normalizing the pixel values between -1 and 1.

4.1.1 Scenario setup in DonKey simulator

Let x_t denote the t-th frame of the image dataset. $X_t = \{x_{t-n+1}, x_{t-n+2}, ..., x_t\}$ denotes n frames associated with states of the vehicle at a given time, $S_t = \{s_{t-n+1}, s_{t-n+2}, ..., s_t\}$, and the actions taken $A_t = \{a_{t-n+1}, a_{t-n+2}, ..., a_t\}$ based on the features learnt by the VAE feature extractor. Learning to maintain a straight line path on the road can be defined as estimating the function $F : \mathbb{R}^{40\times120\times3\times n} \times \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^{40\times120\times3}$ that predicts $x_{t+1} = F(X_t, S_t, A_t)$.

In high dimensional dense spaces, the problem of action conditioned transitions and intermediate representations are important, because convergence probabilities and con-

sequent control actions tend to become unstable at higher dimensions [58], [42]. As these dimensions are correlated, and the straight line trajectory observance problem converges slowly or otherwise results in underfitting [27]. When a regularization factor is applied, the DL model learns a transformation while still adjusting the reward function and policy loss due to correlations in time [58]. Previous approaches available in literature have learned the function F directly, in case of processing artificial videos [44]. The function F is learned in a piecewise manner so that the efficiency and performance can be improved separately [78]. The agent then learns an autoencoder to embed the frames captured by the VAE into a Gaussian space $z \in \mathbb{R}^{2048}$, where the dimensionality 2048 was chosen experimentally and the Gaussian assumption was enforced in accordance with Bayes variational autoencoding [43].

The autonomous driving problem is partly restricted to a learning problem constrained in the pixel space defined as a hypersphere of radius ρ [44], [54], [79]. To ameliorate the problem complexity, we decided to learn action prediction with a policy gradient architecture (DDPG) and an actor-critic architecture (SAC), both preceded by an autoencoder:

- a variational auto encoder (VAE) for dimensionality reduction
- a deep deterministic policy gradient (DDPG) to determine actions based on maximized reward
- an action conditioned soft-actor critic (SAC) for learning the transitions

This work learns autonomous driving using VAE, DDPG, and SAC to emulate real world driving. The self-driving vehicle needs to learn the road images, and a VAE encodes the road images into probabilistic Guassian space as well as decodes them to 3D space. We combine VAE and SAC with a learned cost function taken from distances between lane deviations of previous states. Finally, we train a VAE alongside a SAC, where the SAC optimizes both the pixel space and the similarity between features

extracted by the CNN. The DDPG receives random pixel samples as input from the Gaussian distributed latent space and output of the VAE network. We trained the VAE for 200 epochs. Each epoch consisted of $10,000$ gradient updates with a batch size of 64. Batches were sampled randomly from driving data as described in the previous section. The VAE architecture followed the DDPG made of 4 convolutional layers each one followed by batch normalization and leaky-ReLU activations. The VAE consists of convolutional layers where each layer except the first layer follows batch normalization and the ReLu activation function. The output of the VAE decoder network is normalized with a binary cross-entropy cost function. The output size of the encoder network is 2048. This leads to a compact representation, approximately 32 times smaller than the original data dimensionality. After training the VAE, we fix all its weights and proceed to the preprocessing step for training the DDPG model.

The simulation results are presented to show the performance of the proposed improved DRL scheme, comprising VAE+DDPG and VAE+SAC. First, the simulation parameters are introduced followed by the simulation results. Next, the policy losses are compared for driving environments and the entropy. Then, the performance of the proposed appraoch is studied in terms of learning rate and acquiring stable driving state. Table 4.1 highlights the features and parameters considered in the simulation.

Table 4.1: Features considered in the simulation

Feature Name	**Description**
policy_entropy	An initial measure of randomness of vehicular agent's decisions
policy_loss	Mean magnitude of policy loss function
serial_timesteps	Total timesteps for which image frames are sampled
time_elapsed	Time taken for vehicular agent to achieve stability and stay on the intended path
value_loss	The total loss function
n_updates	The number of iterations of VAE+DDPG and VAE+SAC algorithm

4.2 Simulation Parameters

The parameters under consideration in this thesis are described briefly as follows:

- **Cumulative Reward**: It describes the mean cumulative episode reward over all the states of the agent over a specific timestamp. The value usually increases during a successful training session. The general trend in reward is to consistently increase over time with some small ups and downs depending on the complexity of the task. However, a significant increase in reward may not be apparent until the training process has undergone multiple iterations [80], [81].
- **Entropy**: It is a measure of how random the decisions of the autonomous agent are. It should gradually decrease during a successful training process. In case the entropy decreases too quickly or does not decrease at all, the DRL architecture's hyper-parameters are reset to a different initial value both in continuous as well as discrete action space [80], [81].
- **Episode Length**: The mean length of each episode in the driving environment for the autonomous vehicular agent [80], [81].
- **Learning Rate**: It signifies the step size taken at a time by the training algorithm to search for the optimal policy [80], [81].
- **Policy Loss**: The policy is defined as the process for deciding actions that lead to optimal driving in the given scenario. Policy loss describes the mean magnitude of policy loss function. This loss correlates to how much the policy changes during an episode in a given timeframe [80], [81].
- **Value Estimate**: It is the mean value estimate for all states visited by the autonomous agent. It corresponds to how much future reward the agent expects to receive at any given state [80], [81].
- **Value Loss**: It defines the mean loss of the value function update. It correlates to

how well the model is able to predict the value of each state. This should increase while the agent is learning, and then decrease once the reward stabilizes. These values also increase as the reward increases, and then decrease as the reward tends to becomes stable [80], [81]. The parameter updates while learning the state-action transition policy of DRL process for a given timeframe are tabulated in Appendix A.

4.3 Performance Analysis

This section provides a comparison of vehcile control algorithms using DRL based on policy gradients (VAE+DDPG) and actor-critic (VAE+SAC). The section discusses the results highlighting the probability of the autonomous vehicle transitioning into a new state, based on the cumulative reward achieved at each timeframe and the execution of an action. The results build on the mathematical framework defined in the previous chapter. Since the autonomous driving problem is defined as first-order Markov decision problem, the state of the vehicle at next timeframe is dependent only on the present state and not the past states. This section compares the viability of VAE+DDPG and VAE+SAC approaches in ensuring that the autonomous vehicle arrives at the current state. Another important contribution of this section is the cumulative reward garnered by VAE+DDPG and VAE+SAC approaches. The variations in cumulative reward accumulated over each timeframe in presence of the ergodic distribution of possible states in the VAE encoded driving environment is also discussed in this section.

4.3.1 Driving environment

In the DonKey simulator, driving environment consists of a road with two lanes, differentiated with a lane marker. Although the road curvature needs the vehicle to make

a slight change in steering angle to stay on the road, the road is devoid of any sharp turns and the major part of the trajectory is linear. To stay on the road, the rewards for action at a specific state are supposed to be high for preferable actions. However, once the vehicle is close to achieving good position in the lane, the reward for next action is considerably lower than in the beginning. A section of the driving scene with the road and vehicle is represented in Fig. 4.1 and Fig.4.2. The simulated driving environment is further depicted in Appendix B.

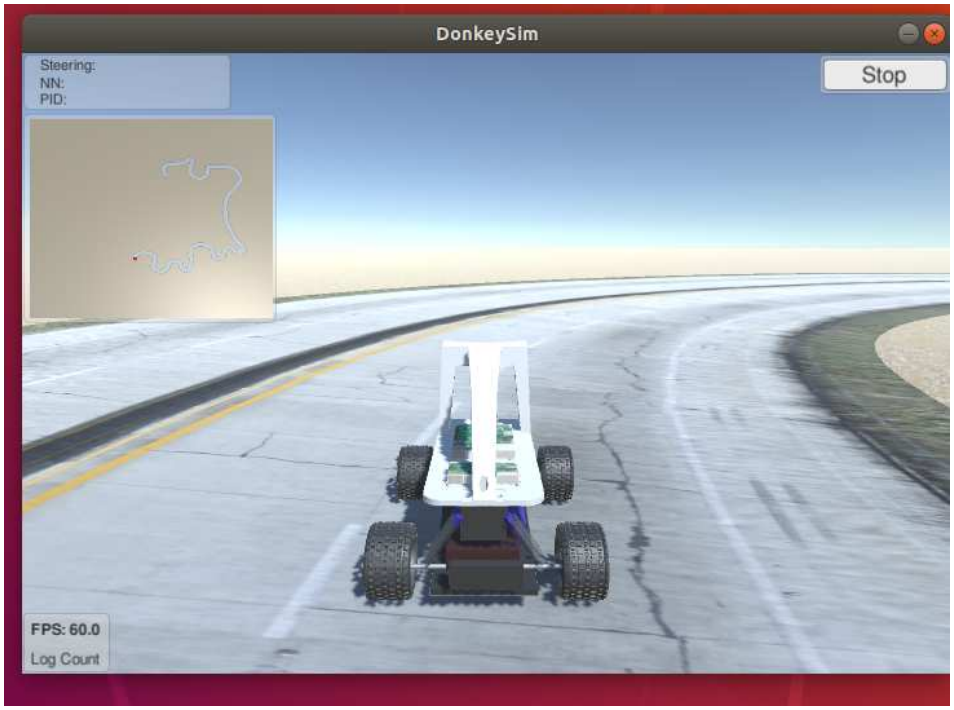


Figure 4.1: Driving manoeuvres in generated road driving environment in Donkey simulator at a given timeframe.

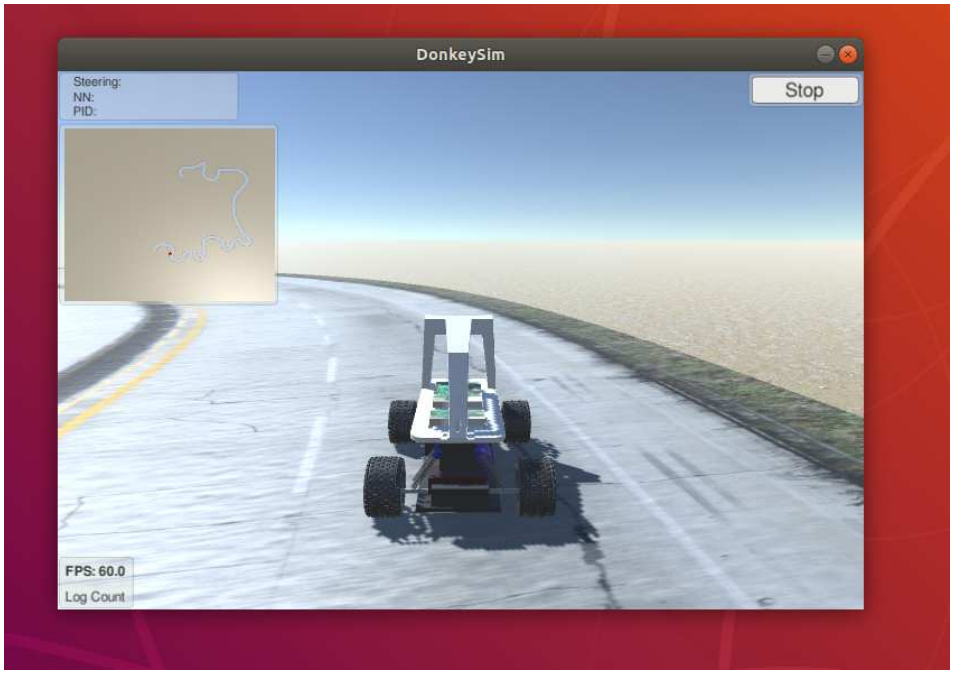


Figure 4.2: Driving manoeuvres in generated road driving environment in Donkey simulator at a subsequent timeframe.

Figure 4.3 represents the variation in cumulative reward with number of timesteps covererd by the vehicle in the simulated driving environment. For the first 20,000 timeframes, the cumulative reward gradually increases. This gradual increase indicates that the vehicle begins with a randomly defined initial state S, and selects a random action A with an aim to maintain its position on the lane. After initial haphazard movements, from 20,000-40,000 timeframes, the cumulative reward increases at the similar rate for VAE+SAC approach, whereas the increase is steeper for VAE+DDPG approach. Furthermore, the VAE+DDPG approach shows minor fluctuations in a specific range, from 10,000-25,000 timeframes. This indicates that once the vehicle learns an optimum action, the deviation from those set of actions attracts a higher penalty as compared to that in the beginning. Consequently, the reward for successive good set of actions is lesser and indicates that the vehicle has to process a smaller set of data to arrive at that action, resulting in nearly smooth cumulative reward.

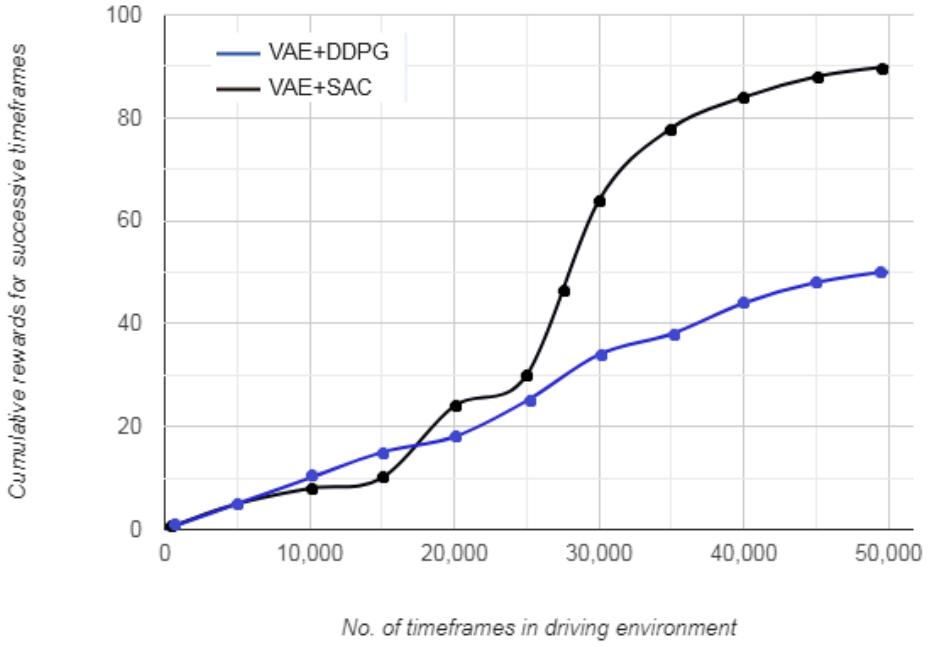


Figure 4.3: Cumulative reward vs no. of timeframes.

Figure 4.4 represents the variations in episode length with the number of timeframes in the driving environment. The episode length defines how long the autonomous vehicle occupies the road before returning to the original position. For the first 10,000 timesteps, the episode length traversed by the vehicle is approximately 900cm for VAE+SAC algorithm and 700cm for VAE+DDPG algorithm. This period indicates the random initial learning by the vehicle that results in a haphazard motion. By the first 10,000 timesteps, the DRL algorithm undergoes sufficient iterations and the episode length traversed by the vehicle begins to constantly increase. After 40,000 timesteps, the episode length does not show a large increase for both VAE+DDPG and VAE+SAC algorithms. The return to original position implies a terminating action, mainly due to the vehicle drifting off the track or merging onto the wrong lane. With increasing number of timesteps, the episode length gradually increases and then converges to an upper maximum value. This is congruous to the pattern followed by cumulative reward, indicating that once the vehicle identifies a set of favourable

actions, the vehicle is able to remain on the road for longer episode lengths, before resorting to terminating action.

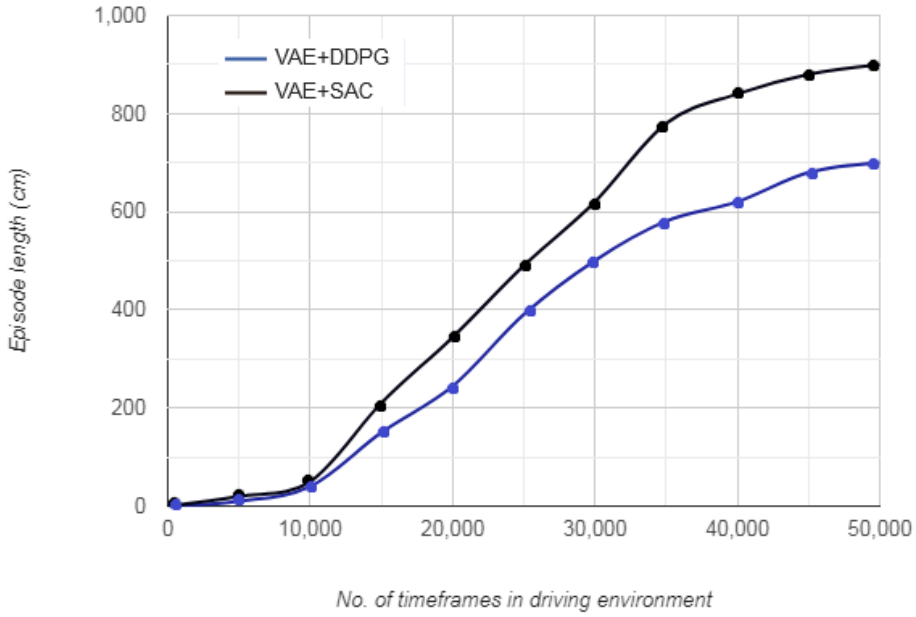


Figure 4.4: Episode length vs no. of timeframes.

Figure 4.5 depicts the value loss vs. the distance traversed by the vehicle before terminating an episode. For the first 10,000 timeframes, the value loss for each successive timestamp indicates that even when the reward function stabilizes, the value loss continues to increase in accordance with the time spent by the vehicle in the driving environment. This implies that at every new timeframe, the vehicle calculates a set of state-action-reward tuple to seek an optimum action. In addition, the value loss tends to exhibit lesser variations based on the multiple state-action-reward cycles that allow the vehicle to arrive at an optimal policy and learn future actions.

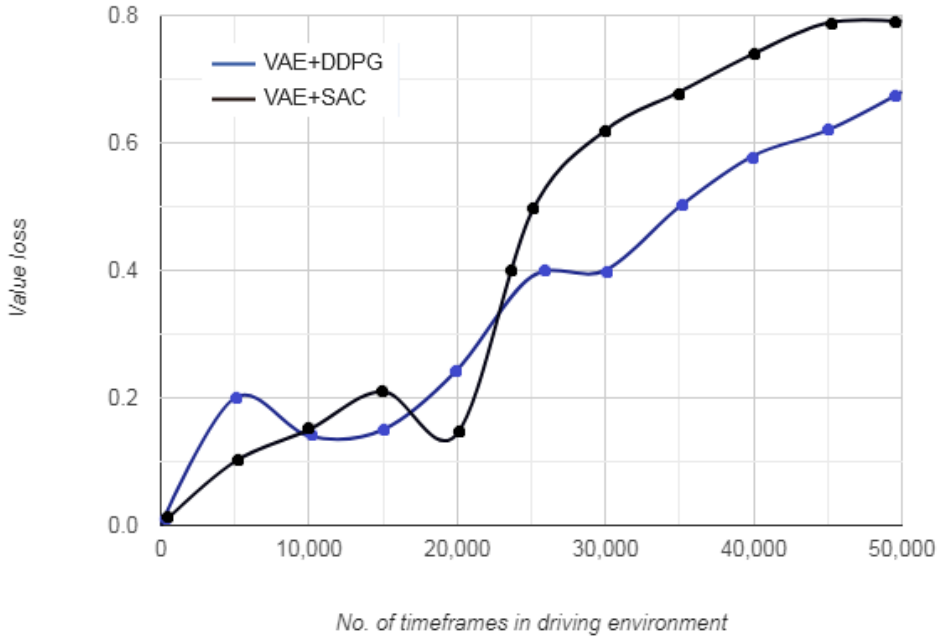


Figure 4.5: Value loss vs no. of timeframes.

4.3.2 Learning losses

The losses describe the delay at arriving at optimal policy in VAE+DDPG and VAE+SAC. The policy losses indicate how much the policy is changing at each timestep and with subsequent actions. During a successful learning phase, the vehicle after starting with random decisions must arrive at more coherent pattern of state, action, and reward.

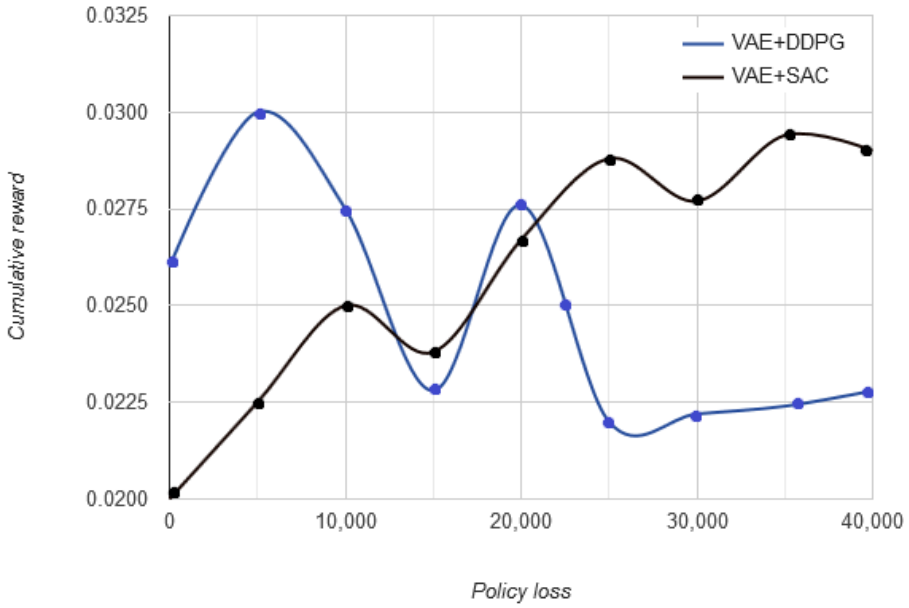


Figure 4.6: Policy loss vs cumulative reward.

The fig. 4.6 and fig. 4.7 highlight the fact that at the time of prediction of next state and to choose an appropriate action, the vehicle uses the cumulative reward in variation with policy loss to predict the next best action to take in the driving environment. The input state is the Q values for all actions and the maximum cumulative reward for taking an action impacts the next reward predicted.

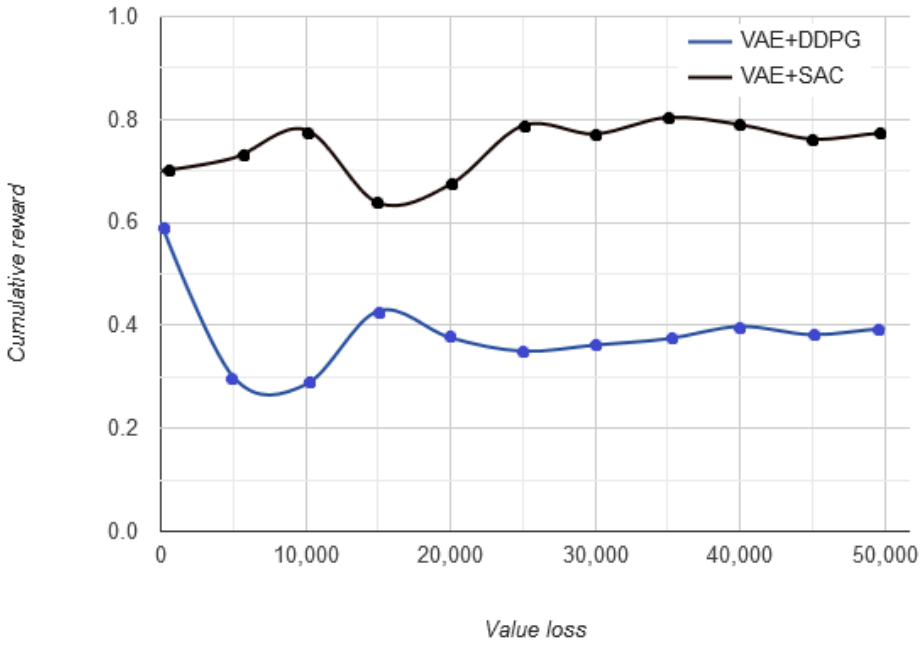


Figure 4.7: Value loss vs cumulative reward

4.3.3 Optimal driving policy

The optimal driving policy indicates that the optimal action is taken at a given state. At a given state whether the action is optimal or not is plotted in policy vs entropy as shown in Fig. 4.8. As the vehicle approaches optimal decision, the randomness in decisions tends to decrease. During a successful scene understanding of the driving environment, decreasing randomness indicates that the vehicle has learnt optimally.

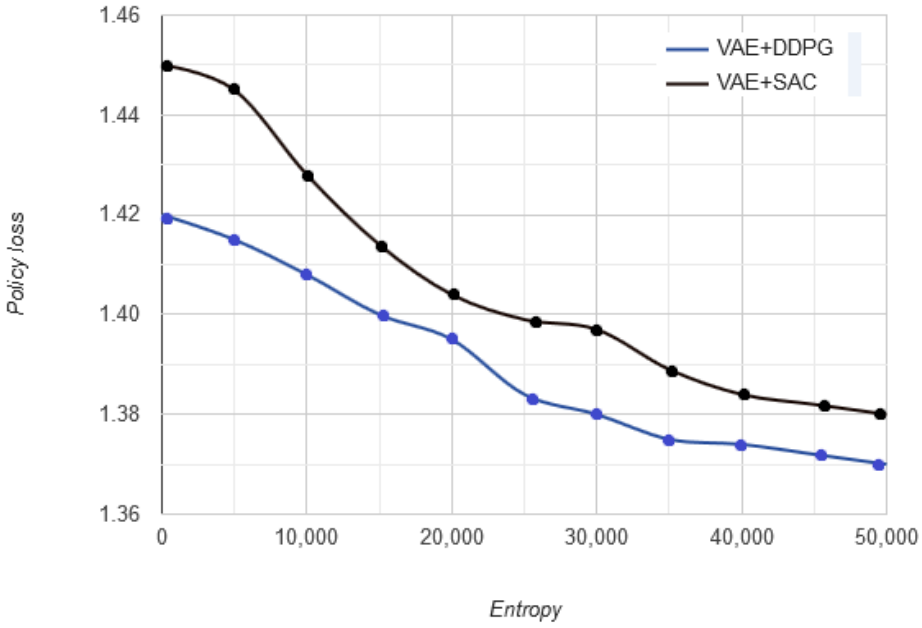


Figure 4.8: Policy loss vs entropy.

In the beginning, as the vehicle moves in a haphazard direction, the algorithm takes larger number of timeframes to adjust vehicle behavior. However, as the vehicle approaches closeness to the driving environment, the haphazard motion is replaced by a more stable trajectory with lesser random movements. This is plotted in Fig. 4.8. With the policy, the randomness of the actions taken by vehicle are quantified by the entropy of that probability distribution. A greater entropy indicates that the actions taken by the agent are more random.

The policy loss is an additional cost-function that the vehicular agent can predict and observe from the driving environment. In DRL apploaches such as DDPG and SAC, these losses are defined and synthesized from unlabeled inputs (processed through VAE), and the variations in losses defined by the reward function. The policy losses lead to elimination of environmental factors and quantities that are irrelevant to solving the autonomous driving problem. For example, the vehicle might estimate the sand regions (Fig. B.5) or the presence of warehouse objects (Fig. B.9) during a driving

task. Or, in other cases,the vehicle might try to predict how close it is to a terminal state. Accordingly, it may deviate from maximizing the DR learning objectives and get involved in local self-imposed rewards. Here, the advantage of policy-based solution approaches over model-based approaches is that a model of the driving environment transitions between the states and actions is used to take subsequent actions and ultimately to directly improve the policy function and value function. In contrast, the model based approaches do not directly improve the main DRL objective, but facilitate the representation of learning process while continuously improving learning stability. Policy and value function based learning bypass the need for explicitly defined relationship between the original input-output being learned.

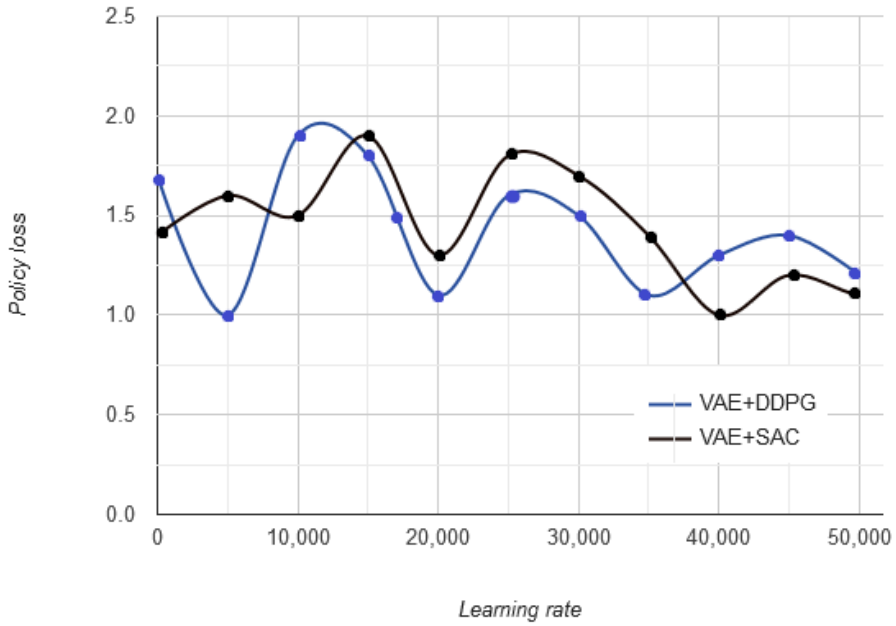


Figure 4.9: Policy vs learning rate.

The policy loss vs. learning is depicted in Fig. 4.9. It provides the autonomous vehicle with updated reward functions based on VAE feature-set by summation across multiple time-steps. In SAC algorithm, the loss function consists of the policy loss (actor), and the value loss (critic), where an entropy loss for the policy is also added. This discourages the agent from converging early to sub-optimal deterministic policies.

For driving task, the loss computation is as follows: suppose the vehiclular agent is awarded a reward +1 for staying on the lane and -1 for drifting off track. Moreover, it is worth investigating how many more state-action pairs can be generated before the termination action is taken. When posed as a general value function problem, the standard DRL problem learns to first maximize the rewards by staying on the track for given timestamps, and also predict how many successful moves remain before termination. The vehicular agent predicts closeness to a terminal state while learning the standard policy. The vehicle tries to maximize the expected reward while also maximizing entropy by acting as less randomly as possible.

4.3.4 Performance comparison for VAE+DDPG vs VAE+SAC

This subsection compares the performance characteristics of VAE+DDPG approach vs VAE+SAC approach for the autonomous vehicle to learn driving behavior by arriving at an optimal state-value function v_* after a specific timestep encompassing different iterations of function $F(X_t, S_t, A_t)$ representing state-action tuple for a specific environment.

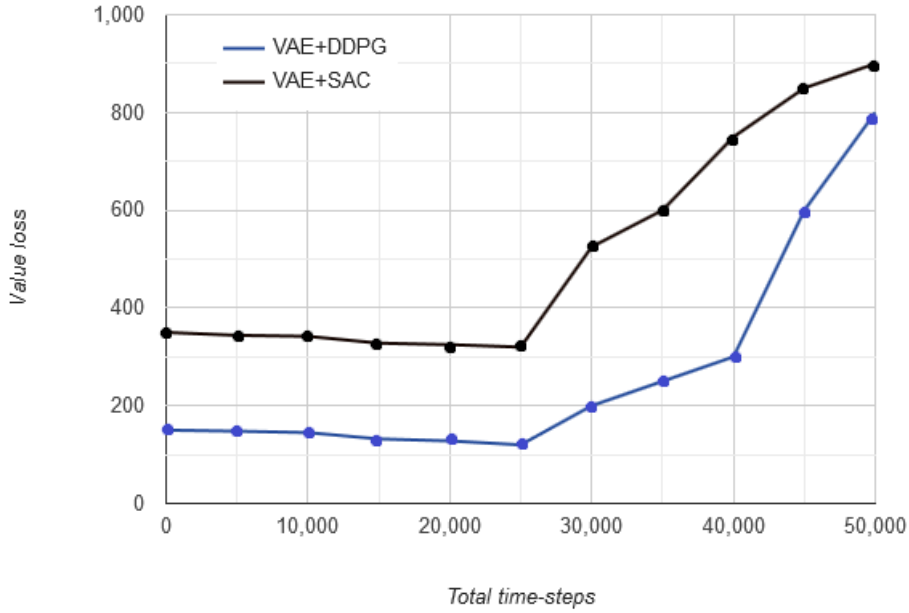


Figure 4.10: Value loss vs Total timesteps.

As seen from Fig. 4.10, for same number of timesteps, after the driving scenario images are processed through VAE, both DDPG and SAC converge after approximately the same number of timesteps. However, the initial randomness in vehicle motion is less in SAC based approach as compared to DDPG based approach. For upto 5,000 timesteps during training phase, the vehicle depicts more haphazard movement to arrive at an optimum action for VAE+DDPG and settles comparatively quicker with VAE+SAC approach. As seen in Fig. 4.11, the mean cumulative reward becomes constant after approximately 50,000 timesteps, indicating that optimal value function V^* and optimal state-action value function $Q^*(s, a)$ has been approximated by the vehicular agent.

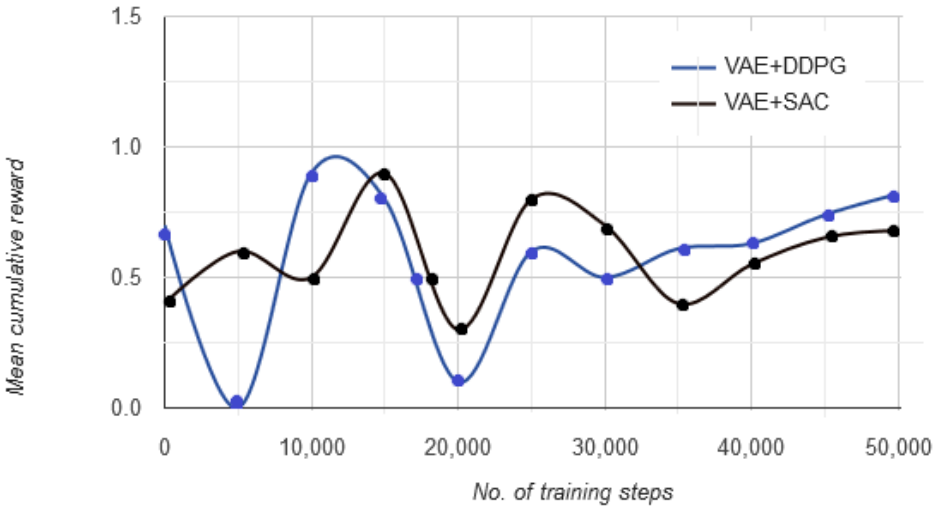


Figure 4.11: Comparison of mean cumulative reward vs number of training steps for VAE+DDPG and VAE+SAC.

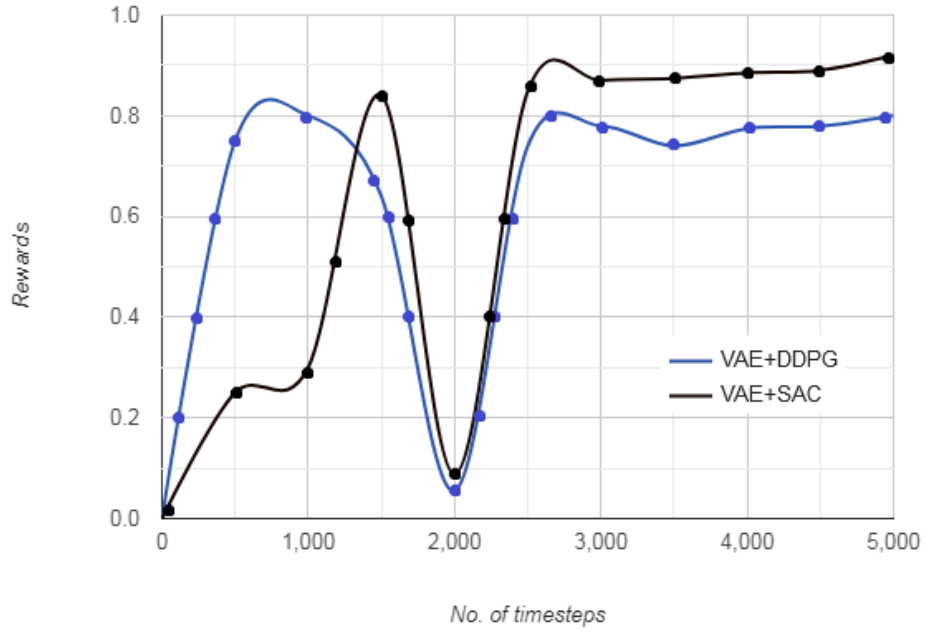


Figure 4.12: Comparison of rewards vs number of training steps for VAE+DDPG and VAE+SAC.

The episode length was used as the actual current episode length to compute the targets. However, when applied as it is, this delays access to the environment until the episode is over and does not provide significant benefit. Alternatively, VAE + SAC

approximates the current episode length by the running average of episode lengths computed from the most recent 5,000 episodes. This improves learning performance, and is memory efficient for distributed on-policy DRL. In addition, it eliminates the need to retain the state-action information until episode termination to compute value loss. Furthermore, they are independent of cumulative rewards and other domain dynamics that might render driving environment representation challenging. These factors are very important in driving environment domain visuals applicable to episodic environments.

The fig. 4.12 presents a plot of rewards vs. the number of steps in the driving environment. The rewards represent the entropy of the learned policy giving an insight into how the vehicular agent learned to navigate the driving environment as well as managed to keep the episode length high (upto 900cm). In the driving environment, the autonomous vehicle agent has to choose from a set of possible actions, i.e. move North, South, West, East, or stay at the current location, all while accelerating, decelerating, or maintaining the same velocity. The reward for the first 1,000 timesteps indicates that the vehicle proceeds with psuedo direction, so it has learned a policy with 0-80 % probability of going haphazardly. From 1,000-3,000 timesteps, the rewards reorganize, indicating reset of state-action-reward pair. After 3,000 timesteps to 5,000 timesteps, the rewards are almost stationary for both VAE+DDPG and VAE+SAC approaches. This indicates that the autonomus vehicle has learned the driving environment and has decided on the optimal action. Also, the reward is slightly higher for VAE+DDPG as compared to VAE+SAC.

4.3.5 Steering smoothness based on loss function

In the previous two subsections, we considered driving tasks that are related to the structure of the learning in VAE+DDPG and VAE+SAC. When the cumulative reward ceases to vary considerably, we consider predicting the reward received at the next

time-step. Given the state sequence, the vehicle aims to predict the reward. That is similar to value learning where the agent only cares about the immediate reward which may be a positive reward, a negative reward, or zero. Due to images collected in driving environment, a data imbalance problem might arise. To mitigate data imbalance problems, class-balancing would require many episodes of samples with zero and non-zero rewards to be provided during training. The variation in haphazard movement initially depicted by the vehicle while trying to learn an optimal driving policy is shown in Table 4.2.

Table 4.2: Steering smoothness based on loss function

Loss function	**fps**	**value_loss**	**total_timesteps**
0.2	9.131	44.682	10,569
0.4	7.714	36.580	22,883
0.6	5.593	38.651	29,315
0.8	4.700	25.739	26,296

4.3.6 Summary and findings

This section discusses the research findings and experimental verification of the VAE+DDPG and VAE+SAC schemes to facilitate learning and navigation in autonomous vehicles. The findings reveal that:

- DRL is applicable in situations where the agent needs to make a decision based on more than one contributing factor. In autonomous driving, the vehicular agent needs to make a sequence of decisions in order to achieve its objective and accumulate rewards.
- DRL does not rely on the assumption that the vehicular agent calculates the losses for each action-state pair. The losses are gradually and cumulatively learnt from the driving experience, supported by the constantly-varying reward function.

- DRL bypasses the need for a pre-defined model that describes how the observed data relates to the states of the environment.
- In autonomous driving scenarios, it may be infeasible to enumerate all possible actions and states, especially when the environment consists of continuous state and action spaces. DRL shows considerably safe driving behavior while the vehicle learns from a limited set of environmental features.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

The proposed DRL technique uses VAE, DDPG, and SAC to implement and analyze a combination of policy function, reward, and penalty to ensure that the autonomous vehicle stays on the track for maximum time in a given timeframe. The DRL strategy, a combination of DDPG, VAE, and SAC algorithms, teaches an autonomous vehicle to drive in a continuous state-space. In a particular driving state, based on the past instances of off-track deviations, terminations, or collisions with obstacles over several frames of previous iterations, the vehicle reinforces its behavior to maximize the reward function. The frames captured at a specific rate also allow simulating possible actions taken by the vehicle using several instances of driving. This thesis presented research on learning driving in incremental stages instead of learning end-to-end. We first trained the DRL model with DDPG based on cost functions to generate realistic looking images of the road. The proposed work emphasizes a step forward toward reducing the uncertainties pertaining to vision and interaction involved in autonomous driving. The work provides a perspective on using policy functions to ensure optimal and fast learning in autonomous vehicles.

The maximum likelihood principle that underlies deep learning applications and probabilistic DL models can indicate the distribution of possible outcomes. Bayesian deep learning models make it possible to capture the uncertainty occurring in real-world applications. Capturing and interpreting the driving environment and possible set of actions a vehicle can take is effectively done using MDP for modeling the environment and generating complex distributions using VAE. The interpretation of the gathered data to execute meaningful action is done using policy gradient or actor critic based DRL methods.

Recent advances in statistical inference have significantly expanded the toolbox of probabilistic modeling of autonomous agents, specifically in robotics and games. Value function approximation leads to optimal policy over a broad class of probabilistic models containing a large number of driving parameters, and scalable methods based on stochastic gradient descent and distributed computation engines. DRL bypasses the need to apply probabilistic modeling over massive data sets. One important application of DRL has been to include deep neural networks within a probabilistic/Markov model to capture complex non-linear stochastic relationships between random variables. These advances in conjunction with the release of novel DRL algorithms such as SAC, VAE, and DDPG have greatly expanded the scope of application of DRL to solve MDP based on probabilistic models. In this thesis, we elaborated on and apply the main concepts, methods and tools of deep neural networks within a MDP framework to estimate the probability of an autonomous vehicle staying on the track.

Performing basic driving maneuvres such as staying on track, detect lane markings, and accelerate/decelerate, such tasks when solved using non-DRL methods require direct access to state variables as well as well-designed hand-engineered features extracted from sensory inputs. Deep reinforcement learning (DRL) can learn complex policies with high-dimensional observations as inputs. The driving environment images offer a suitable mechanism to apply them for autonomous driving to learn to drive on a

road in a manner similar to human driving. However, applying DRL to autonomous driving still remains very challenging as conventional DRL algorithms require a large number of training samples for learning, which is infeasible and time-consuming in real-world driving scenarios. The contribution of this thesis is twofold. We proposed two DRL algorithms, VAE+DDPG and VAE+SAC. The combination of these techniques leads to smooth policy update in value function based DRL with enhanced capability of automatic feature extraction. Using VAE pre-processing enhances the sample efficiency and facilitates the learning process with fewer but robust samples.

5.2 Future Work

We aim to include deep reinforcement learning (DRL) and machine learning (ML) at various aspects of autonomous driving to extend the work into foreseeable connected and autonomous vehicle (CAV) infrastructure. Some future research directions are proposed below:

- It is investigative if videos consisting of complex textures could be processed in real-time with DRL. As human beings excel in processing complex, unforeseen, and unexpected circumstances, a similar level of trust in self-driving technology is undermined until these vehicles can match human intelligence for sustained time-periods.
- We aim to extend the VAE to learn the pixel space defined by Gaussian framework to generate realistic looking frames, images and videos predicting the autonomous vehicle behaviour. This would be a step forward towards receiving feedback-based corrective action ahead of the next timeframe.
- In order to tune the hyperparameters and weights involved in the training, the DRL model used in the thesis can be appended with probabilistic DL models.

Bayesian deep learning models make it possible to capture the uncertainties occurring in real-world applications to model the distribution of possible outcomes.

- DRL is prone to perturbation attacks, where a small modification in input may lead to drastic changes in learning policy functions, optimal decisions, and resulting actions. Hence, incorporating cybersecurity measures in the learning aspect of autonomus vehicles is crucial to broader societal impacts of such vehicles.
- In order for autonomous vehicles to increase the uninterrupted drive-time, DRL techniques can be used in conjunction with structured probabilistic models. Probabilistic DRL would provide computationally efficient tools to learn features from latent variables.

Appendices

Appendix A

Parameter updates while learning state-action transition policy

approxkl	4.7656602e-05
clipfrac	0.0
explained_variance	0.0293
fps	351
n_updates	1
policy_entropy	0.6931137
policy_loss	-0.00085980096
serial_timesteps	128
time_elapsed	3.58e-06
total_timesteps	128
value_loss	44.276386

(a) Parameter-update 1

approxkl	2.1149872e-05
clipfrac	0.0
explained_variance	-0.000894
fps	969
n_updates	2
policy_entropy	0.6929701
policy_loss	-0.0004953899
serial_timesteps	256
time_elapsed	0.366
total_timesteps	256
value_loss	23.76554

(b) Parameter-update 2

approxkl	0.00018347782
clipfrac	0.0
explained_variance	0.00482
fps	983
n_updates	3
policy_entropy	0.6930482
policy_loss	-0.0027523388
serial_timesteps	384
time_elapsed	0.5
total_timesteps	384
value_loss	40.459938

(c) Parameter-update 3

approxkl	6.747682e-05
clipfrac	0.0
explained_variance	0.0121
fps	1071
n_updates	4
policy_entropy	0.69261056
policy_loss	-0.00084777543
serial_timesteps	512
time_elapsed	0.631
total_timesteps	512
value_loss	33.19421

(d) Parameter-update 4

approxkl	7.634766e-05
clipfrac	0.0
explained_variance	0.00675
fps	1066
n_updates	5
policy_entropy	0.6913911
policy_loss	-0.0011928186
serial_timesteps	640
time_elapsed	0.752
total_timesteps	640
value_loss	25.366756

(e) Parameter-update 5

approxkl	2.6476517e-05
clipfrac	0.0
explained_variance	-0.0131
fps	1103
n_updates	6
policy_entropy	0.69117665
policy_loss	-0.00041625777
serial_timesteps	768
time_elapsed	0.873
total_timesteps	768
value_loss	42.974476

(f) Parameter-update 6

Figure A.1: Parameter values at different stages of learning.

Appendix B

Images during driving

Figure B.1: Driving scenarios and environments in DonKey simulator.

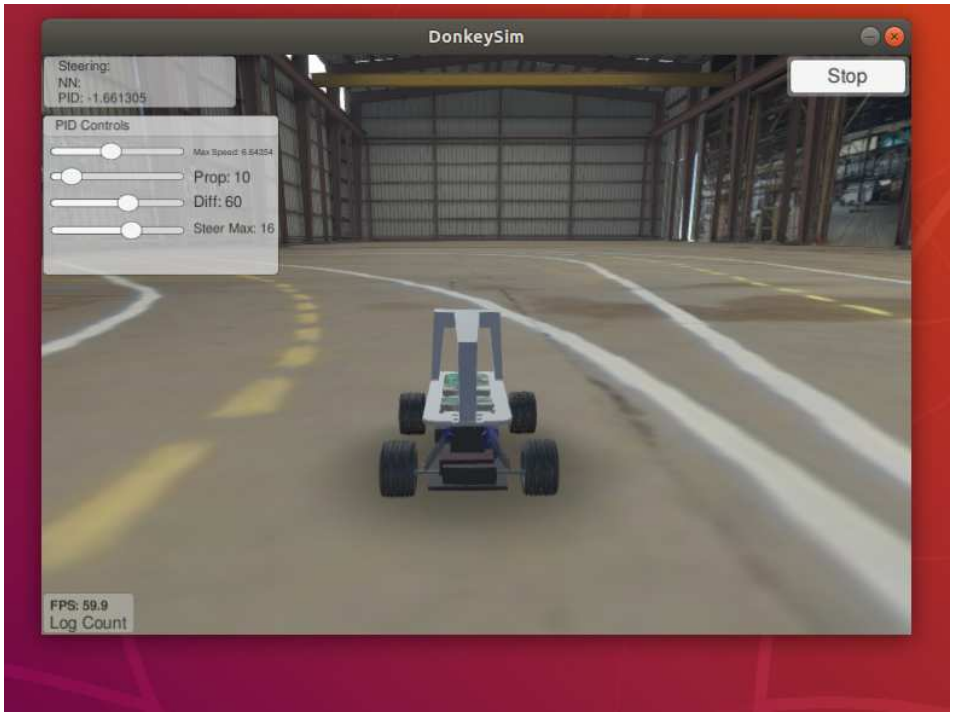


Figure B.2: Warehouse driving environment in Donkey simulator.

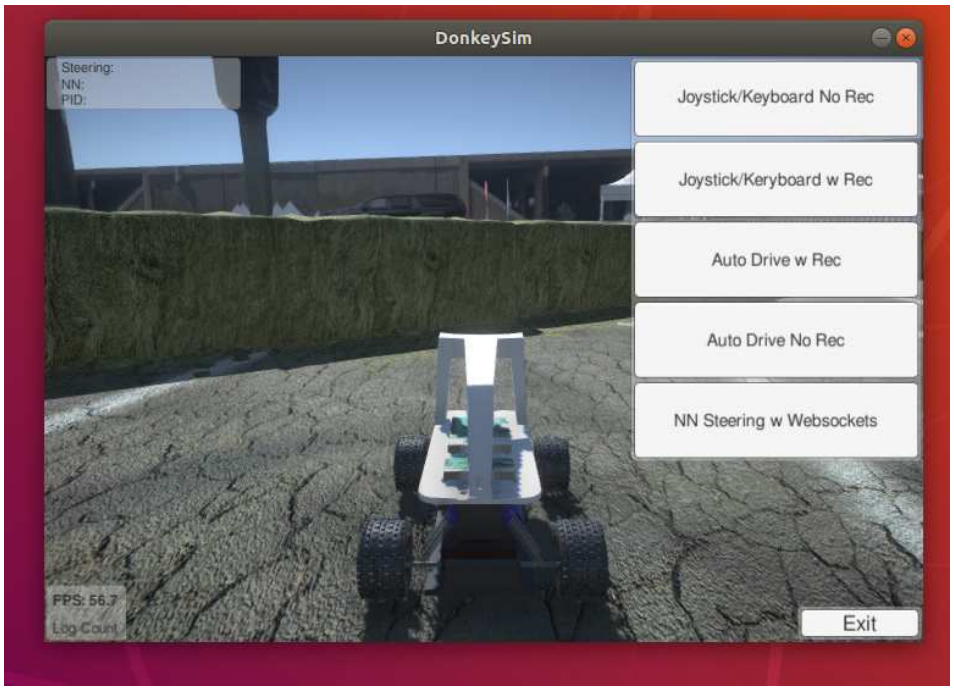


Figure B.3: Sparkfun AVC driving environment in Donkey simulator with ability to capture driving images for analysis.

Figure B.4: Driving environment in Donkey simulator at a given timeframe.

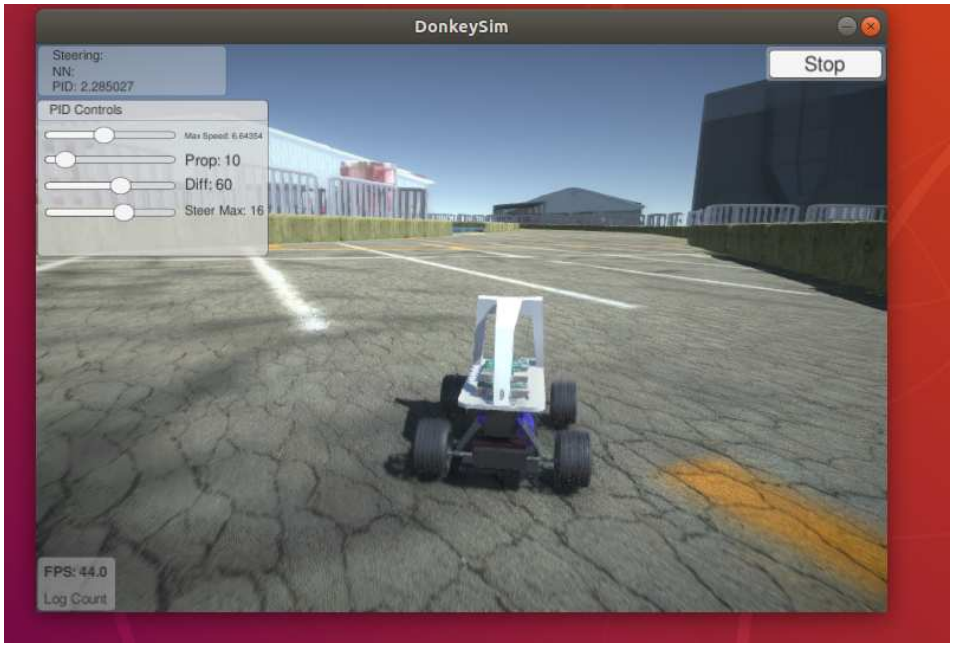


Figure B.5: Driving environment in Donkey simulator at a given timeframe.

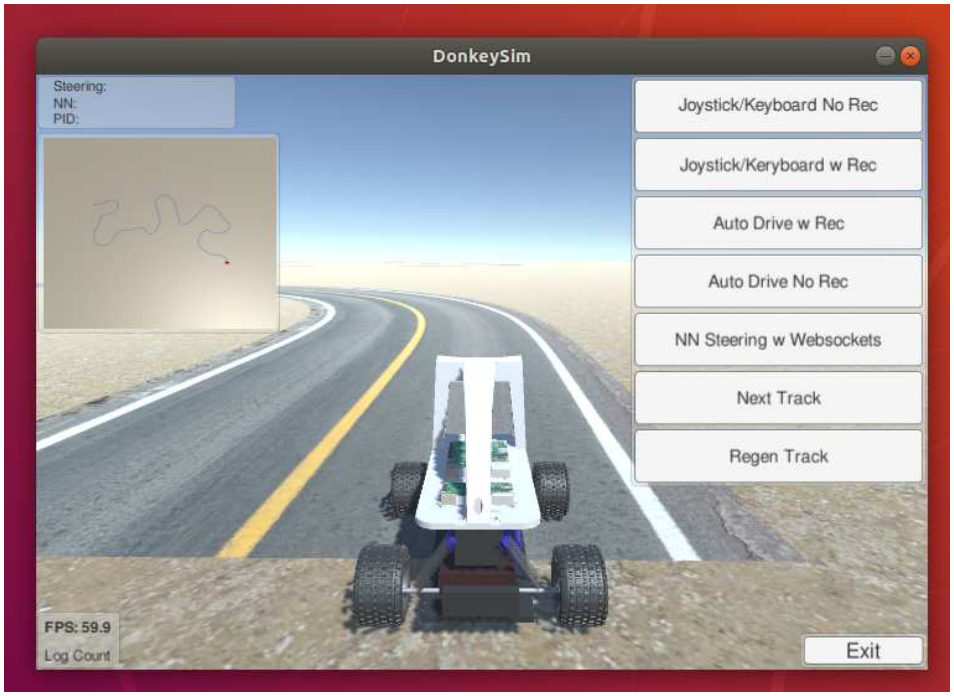


Figure B.6: Generated road driving environment in Donkey simulator at a given timeframe.

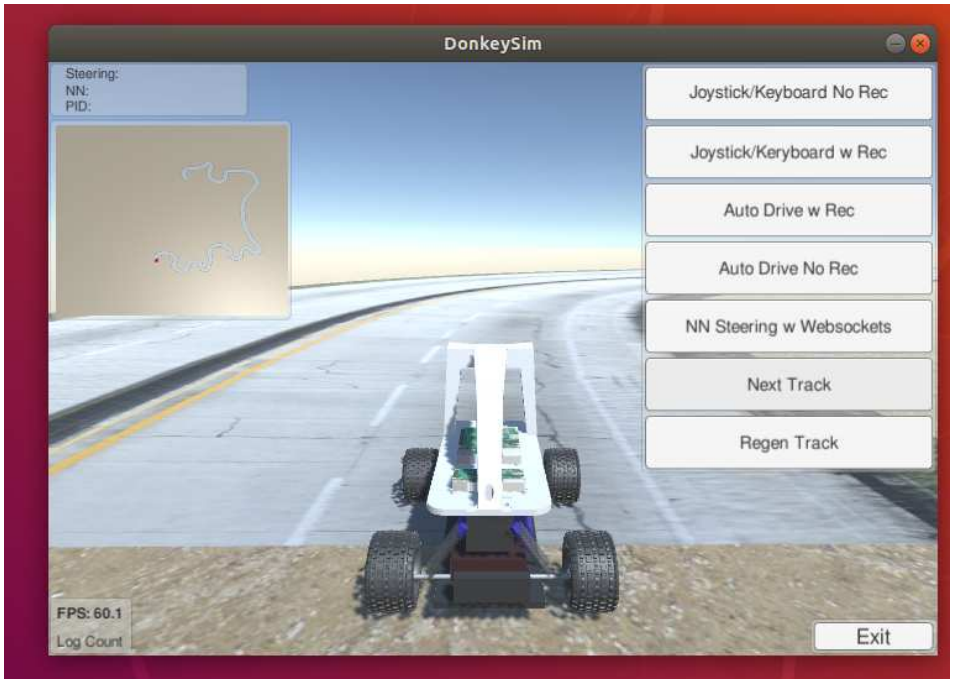


Figure B.7: Generated road driving environment in Donkey simulator at a given timeframe with captured images.

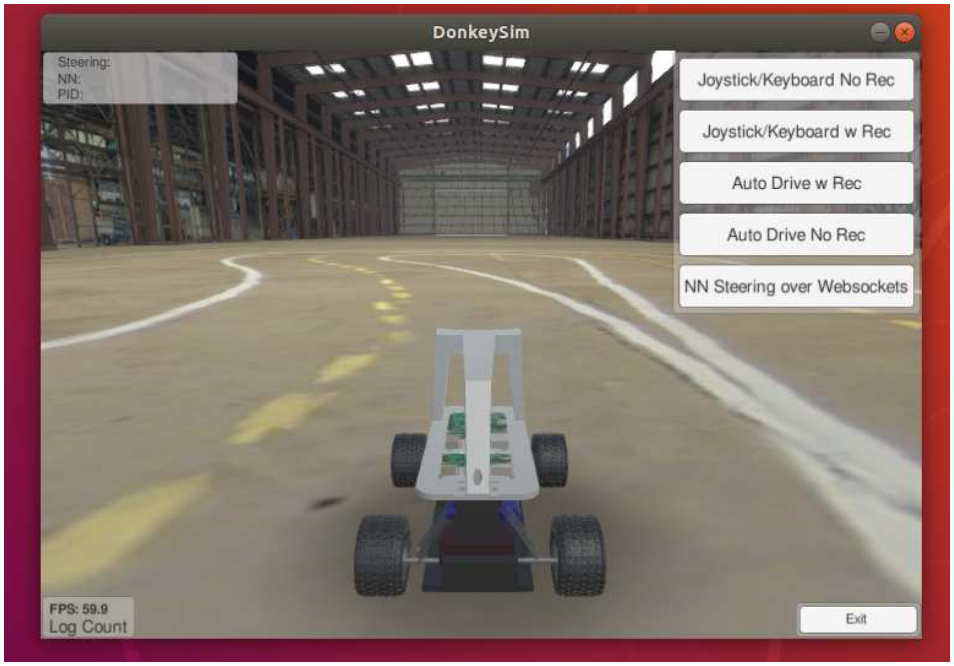


Figure B.8: Donkey simulator driving state 1.

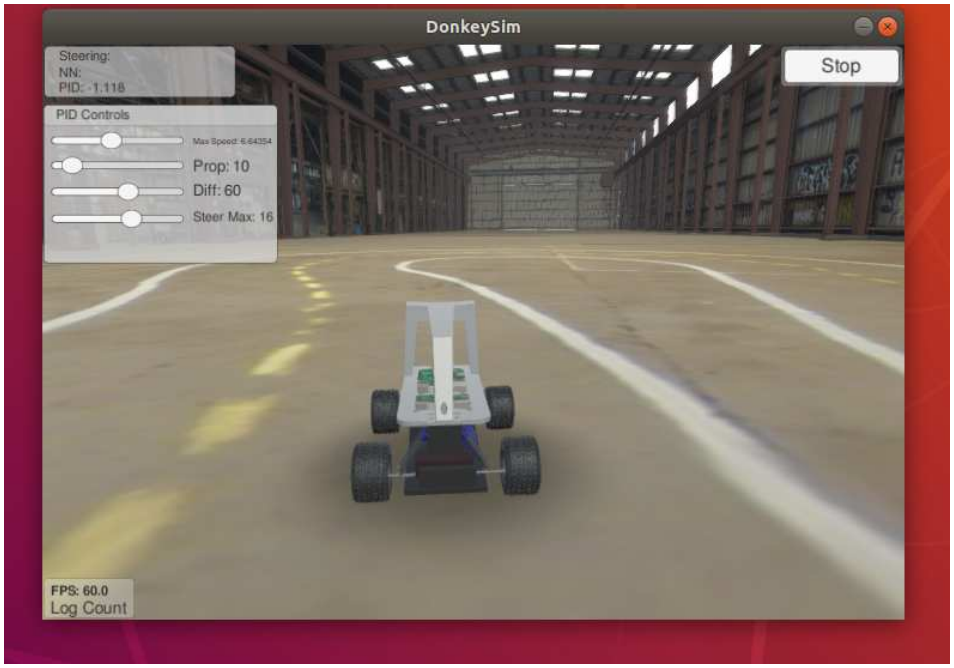


Figure B.9: Donkey simulator driving state 2.

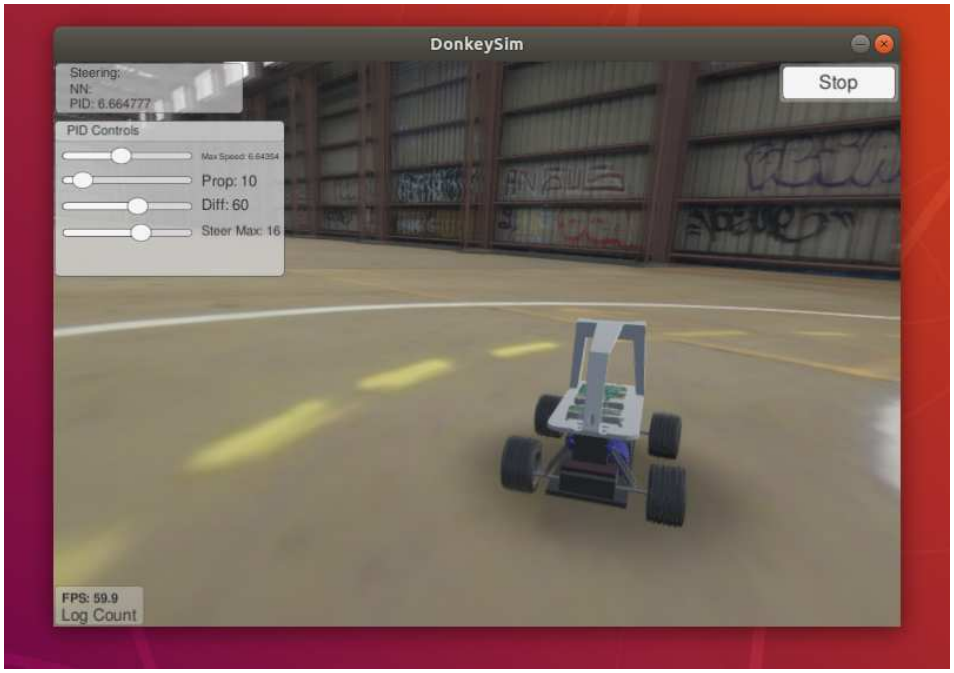


Figure B.10: Donkey simulator driving state 3.

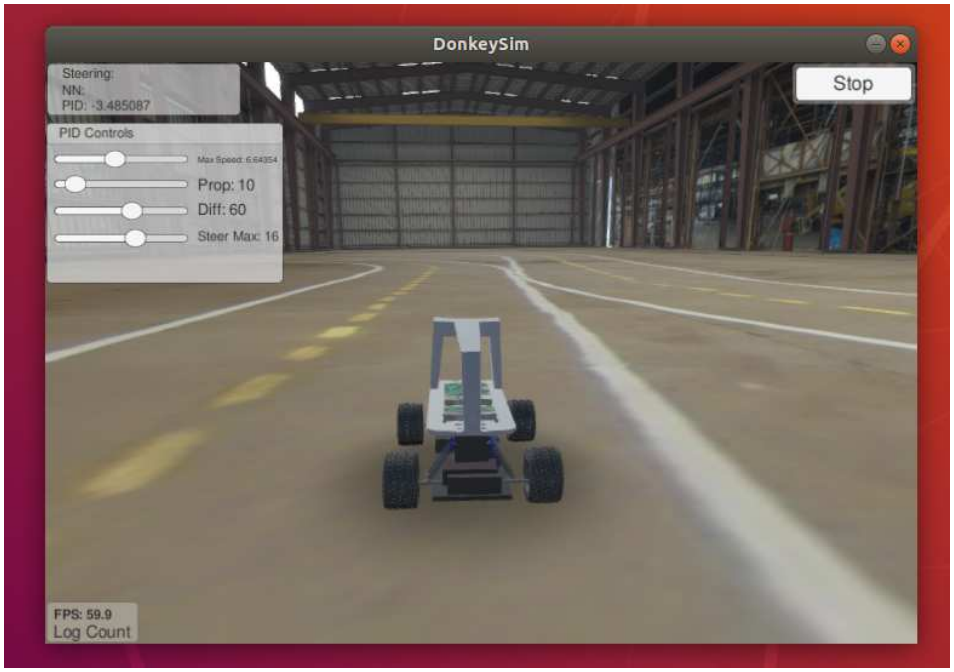


Figure B.11: Donkey simulator driving state 4.

Figure B.12: Driving environment road trajectory 1 for autonomous vehicle navigation in Donkey simulator.

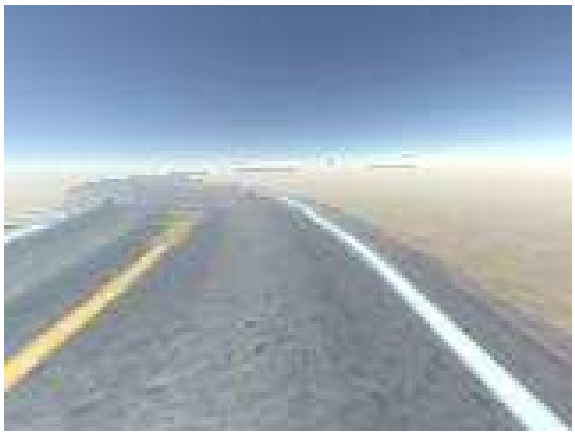


Figure B.13: Driving environment road trajectory 2 for autonomous vehicle navigation in Donkey simulator.

Bibliography

[1] V. Papathanasopoulou and C. Antoniou, "Towards data-driven car-following models," *Transportation Research Part C*, vol. 55, pp. 496–509, 2019.

[2] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Choi, "Action-Driven Visual Object Tracking With Deep Reinforcement Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2239–2252, 2018.

[3] M. Bíl, R. Andrasik, J. Sedoník, and V. Cícha, "ROCA: An ArcGIS toolbox for road alignment identification and horizontal curve radii computation," *PloS one*, vol. 13, no. 12, p. 0208407, 2018.

[4] F. Gross and P. Jovanis, "Estimation of the Safety Effectiveness of Lane and Shoulder Width Case-Control Approach," *Journal of Transportation Engineering*, vol. 133, no. 6, pp. 362–369, 2007.

[5] T. Inagaki and T. Sheridan, "A critique of the SAE conditional driving automation definition, and analyses of options for improvement," *Cognition, Technology and Work*, vol. 21, no. 4, pp. 569–578, 2019.

[6] J. Barkenbus, "Self-driving Cars: How Soon Is Soon Enough?," *Issues in Science and Technology*, vol. 34, no. 4, pp. 23–26, 2018.

[7] A. Lieto, M. Bhatt, A. Oltramari, and D. Vernon, "The role of cognitive architectures in general artificial intelligence," *Cognitive Systems Research*, vol. 48, pp. 1–3, 2018.

[8] T. Miki, T. Ohya, H. Yoshino, and N. Umeda, "The Overview of the 4th Generation Mobile Communication System," in *The Fifth International Conference on Information, Communications and Signal Processing*, pp. 1551–1555, December 2005.

[9] A. Miglani and N. Kumar, "Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges," *Vehicular Communications*, vol. 20, p. 100184, 2019.

[10] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational intelligence and neuroscience*, vol. 2018, pp. 7068349–13, 2018.

[11] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. ISSN:1556-4967, 2019.

[12] D. Yang, K. Jiang, D. Zhao, C. Yu, Z. Cao, S. Xie, Z. Xiao, X. Jiao, S. Wang, and K. Zhang, "Intelligent and connected vehicles: Current status and future perspectives," *Science China Technological Sciences*, vol. 61, no. 10, pp. 1446–1471, 2018.

[13] G. Nardini, A. Virdis, C. Campolo, A. Molinaro, and G. Stea, "Cellular-V2X Communications for Platooning: Design and Evaluation," *Sensors (Basel, Switzerland)*, vol. 18, no. 5, p. 1527, 2018.

[14] S. Deb, L. Strawderman, D. Carruth, J. DuBien, B. Smith, and T. Garrison, "Development and validation of a questionnaire to assess pedestrian receptivity toward fully autonomous vehicles," *Transportation Research Part C*, vol. 84, pp. 178–195, 2017.

[15] R. Robertson, S. Meister, W. Vanlaar, and M. Mainegra Hing, "Automated vehicles and behavioural adaptation in Canada," *Transportation Research Part A*,

vol. 104, pp. 50–57, 2017.

[16] M. Kyriakidis, R. Happee, and J. de Winter, "Public opinion on automated driving: Results of an international questionnaire among 5000 respondents," *Transportation Research Part F: Psychology and Behaviour*, vol. 32, pp. 127–140, 2015.

[17] N. Merat, A. Jamson, F. Lai, M. Daly, and O. Carsten, "Transition to manual: Driver behaviour when resuming control from a highly automated vehicle," *Transportation Research Part F: Psychology and Behaviour*, vol. 27, pp. 274–282, 2014.

[18] X. Ji, X. He, C. Lv, Y. Liu, and J. Wu, "Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits," *Control Engineering Practice*, vol. 76, pp. 41–53, 2018.

[19] C. Zhang and J. Kim, "Multi-scale pedestrian detection using skip pooling and recurrent convolution," *Multimedia Tools and Applications*, vol. 78, no. 2, pp. 1719–1736, 2019.

[20] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. ISSN:1556-4959, 2019.

[21] D. Zhao, D. Liu, F. Lewis, J. Principe, and S. Squartini, "Special Issue on Deep Reinforcement Learning and Adaptive Dynamic Programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2038–2041, 2018.

[22] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[23] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[24] V. Banks and N. Stanton, "Driver-centred vehicle automation: using network analysis for agent-based modelling of the driver in highly automated driving systems," *Ergonomics*, vol. 59, no. 11, pp. 1442–1452, 2016.

[25] C. Wang, J. Delport, and Y. Wang, "Lateral Motion Prediction of On-Road Preceding Vehicles: A Data-Driven Approach," *Sensors (Basel, Switzerland)*, vol. 19, no. 9, p. 2111, 2019.

[26] W. Yang, X. Zhang, Y. Tian, W. Wang, J. Xue, and Q. Liao, " Deep Learning for Single Image Super-Resolution: A Brief Review," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019.

[27] M. Buda, A. Maki, and M. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[28] A. Dairi, F. Harrou, M. Senouci, and Y. Sun, "Unsupervised obstacle detection in driving environments using deep-learning-based stereovision," *Robotics and Autonomous Systems*, vol. 100, pp. 287–301, 2018.

[29] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, 2019.

[30] D. Birnbacher and W. Birnbacher, "Fully Autonomous Driving: Where Technology and Ethics Meet," *IEEE Intelligent Systems*, vol. 32, no. 5, pp. 3–4, 2017.

[31] M. Marcano, J. Matute, R. Lattarulo, E. Martí, and J. Pérez, "Low Speed Longitudinal Control Algorithms for Automated Vehicles in Simulation and Real Platforms," *Complexity*, vol. 2018, pp. 1–12, 2018.

[32] E. Debada and D. Gillet, "Virtual Vehicle-Based Cooperative Maneuver Planning for Connected Automated Vehicles at Single-Lane Roundabouts," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, pp. 35–46, 2018.

[33] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation Research Part C*, vol. 97, pp. 348–368, 2018.

[34] O. Garcia, G. Vitor, J. Ferreira, P. Meirelles, and A. de Miranda Neto, "The VILMA intelligent vehicle: an architectural design for cooperative control between driver and automated system," *Journal of Modern Transportation*, vol. 26, no. 3, pp. 220–229, 2018.

[35] D. Shin, K. Park, and M. Park, "Effects of Vehicular Communication on Risk Assessment in Automated Driving Vehicles," *Applied Sciences*, vol. 8, no. 12, p. 2632, 2018.

[36] S. Lefevre, A. Carvalho, and F. Borrelli, "A Learning-Based Framework for Velocity Control in Autonomous Driving," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 32–42, 2016.

[37] W. Wang, J. Xi, and D. Zhao, "Learning and Inferring a Driver's Braking Action in Car-Following Scenarios," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 3887–3899, 2018.

[38] C. Guindel, D. Martín, and J. Armingol, "Traffic scene awareness for intelligent vehicles using ConvNets and stereo vision," *Robotics and Autonomous Systems*, vol. 112, pp. 109–122, 2019.

[39] D. Xie, L. Zhang, and L. Bai, "Deep Learning in Visual Computing and Signal Processing," *Applied Computational Intelligence and Soft Computing*, pp. 1–13, 2017.

[40] C. Webster, "Alan Turing's unorganized machines and artificial neural networks: his remarkable early work and future possibilities," *Evolutionary Intelligence*, vol. 5, no. 1, p. 35, 2012.

[41] M. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, A. Van Esesn, B.C. Awwal, and V. Asari, " The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," *arXiv.org, https://arxiv.org/ftp/arxiv/papers/1803/1803.01164.pdf*, 2018.

[42] M. Biehl, C. Guckelsberger, C. Salge, S. Smith, and D. Polani, "Expanding the Active Inference Landscape: More Intrinsic Motivations in the Perception-Action Loop," *Frontiers in neurorobotics*, vol. 12, p. 45, 2018.

[43] S. Nie, M. Zheng, and Q. Ji, "The Deep Regression Bayesian Network and Its Applications: Probabilistic Deep Learning for Computer Vision," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 101–111, 2018.

[44] H. Haddad, Z. Bouyahia, and N. Jabeur, "Transportation Service Redundancy From a Spatio-Temporal Perspective," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 157–166, 2019.

[45] A. Brunetti, D. Buongiorno, G. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing*, vol. 300, pp. 17–33, 2018.

[46] X. Geng, H. Liang, B. Yu, P. Zhao, L. He, and R. Huang, "A Scenario-Adaptive Driving Behavior Prediction Approach to Urban Autonomous Driving," *Applied Sciences*, vol. 7, no. 4, p. 426, 2018.

[47] C. Guindel, D. Martin, and J. Armingol, "Fast Joint Object Detection and Viewpoint Estimation for Traffic Scene Understanding," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, pp. 74–86, 2018.

[48] P. Pahlavani, M. Poor Arab Moghadam, and B. Bigdeli, "Car Following Prediction Based on Support Vector Regression and Multi-adaptive Regression Spline by Considering Instantaneous Reaction Time," *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, vol. 43, no. S1, pp. 67–79, 2019.

[49] N. Wuong, D. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[50] H. Ye, G. Li, and B. Juang, "Deep Reinforcement Learning Based Resource Allocation for V2V Communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[51] Y. He, N. Zhao, and H. Yin, "Integrated Networking, Caching, and Computing for Connected Vehicles: A Deep Reinforcement Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.

[52] J. Bischoff, M. Maciejewski, T. Schlenther, and K. Nagel, "Autonomous Vehicles and their Impact on Parking Search," *Multimedia Tools and Applications*, vol. 11, no. 4, pp. 19–27, 2019.

[53] D. Blei, A. Kucukelbir, and J. McAuliffe, "Variational Inference: A Review for Statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[54] O. Sigaud and F. Stulp, "Policy search in continuous action domains: An overview," *IEEE Signal Processing Magazine*, vol. 113, pp. 28–40, 2019.

[55] D. Zhao, D. Liu, F. Lewis, J. Principe, and S. Squartini, "Special Issue on Deep Reinforcement Learning and Adaptive Dynamic Programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2038–2041, 2018.

[56] S. Parisi, V. Tangkaratt, J. Peters, and M. Khan, "TD-regularized actor-critic methods," *Machine Learning*, vol. 108, no. 8, pp. 1467–1501, 2019.

[57] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hybrid Trajectory Planning for Autonomous Driving in On-Road Dynamic Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. ISSN:1524-9050, pp. 1–15, 2019.

[58] J. Chen, "The Evolution of Computing: AlphaGo," *Computing in Science and Engineering*, vol. 18, no. 4, pp. 4–7, 2016.

[59] A. Raffin, "Learning to Drive Smoothly in Minutes: Reinforcement Learning on a Small Racing Car," *Web resource https://towardsdatascience.com/learning-to-drive-smoothly-in-minutes-450a7cdb35f4*, 2019.

[60] K. Divakarla, A. Emadi, S. Razavi, S. Habibi, and F. Yan, "A review of autonomous vehicle technology landscape," *International Journal of Electric and Hybrid Vehicles*, vol. 11, no. 4, pp. 320–345, 2019.

[61] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. Díaz-Rodríguez, and D. Filliat, "Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics," *In: Proceedings of the Workshop on Structure and Priors in Reinforcement Learning at International conference on learning representation*, no. http://arxiv.org/licenses/nonexclusive-distrib/1.0, pp. 1–17, 2019.

[62] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 865–873, 2015.

[63] L. Liang, H. Ye, and G. Li, "Toward Intelligent Vehicular Networks: A Machine Learning Framework," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 124–135, 2019.

[64] S. Bernardini, M. Fox, and D. Long, "Combining temporal planning with probabilistic reasoning for autonomous surveillance missions," *Autonomous Robots*, vol. 41, no. 1, pp. 181–203, 2017.

[65] C. Wen, S. Jin, K. Wong, J. Chen, and P. Ting, "Channel Estimation for Massive MIMO Using Gaussian-Mixture Bayesian Learning," *IEEE Transactions on Wireless Communications*, vol. 14, no. 3, pp. 1356–1368, 2015.

[66] L. Tran, J. Kossaifi, Y. Panagakis, and M. Pantic, "Disentangling Geometry and Appearance with Regularised Geometry-Aware Generative Adversarial Networks," *International Journal of Computer Vision*, vol. 127, no. 6, pp. 824–844, 2019.

[67] R. Andrášik and M. B., "Efficient Road Geometry Identification from Digital Vector Data," *Journal of Geographical Systems*, vol. 18, pp. 249–269, 2016.

[68] F. Rosey and J. Auberlet, "Trajectory variability: Road geometry difficulty indicator," *Safety Science*, vol. 50, no. 9, pp. 1818–1828, 2012.

[69] O. Karaduman, H. Eren, H. Kurum, and M. Celenk, "Road-Geometry-Based Risk Estimation Model for Horizontal Curves," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1617–1627, 2016.

[70] L. Hammarstrand, M. Fatemi, A. Garcia-Fernandez, and L. Svensson, "Long-Range Road Geometry Estimation Using Moving Vehicles and Roadside Observations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2144–2158, 2016.

[71] S. Hamdar, L. Qin, and A. Talebpour, "Weather and road geometry impact on longitudinal driving behavior Exploratory analysis using an empirically supported acceleration modeling framework," *Transportation Research Part C*, vol. 67, no. 2, pp. 193–213, 2016.

[72] A. Rasouli and J. Tsotsos, "Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–19, 2019.

[73] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah, "Learning to Drive in a Day," *https://arxiv.org/pdf/1807.00412.pdf*, 2018.

[74] L. He, Y. Chu, and C. Shen, " A Design of Reward Function in Multi-Target Trajectory Recovery with Deep Reinforcement Learning," *IEEE 8th Joint International Information Technology and Artificial Intelligence Conference*, vol. 10.1109/ITAIC.2019.8785878, pp. 286–293, 2019.

[75] L. Matignon, G. Laurent, and N. Le Fort Piat, " Reward function and initial values: Better choices for accelerated Goal-directed reinforcement learning," *Lecture Notes in Computer Science*, vol. 1, no. 4131, pp. 840–849, 2006.

[76] Mausam and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*. Morgan and Claypool, 2012.

[77] X. Guo and Hernández-Lerma, *Continuous-time markov decision processes: theory and applications*. Springer-Verlag, 2009.

[78] J. Guo, X. Gong, W. Wang, X. Que, and J. Liu, "SASRT: Semantic-Aware Super-Resolution Transmission for Adaptive Video Streaming over Wireless Multimedia Sensor Networks," *Sensors (Basel, Switzerland)*, vol. 19, no. 14, p. 3121, 2019.

[79] E. Arnold, O. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A Survey on 3D Object Detection Methods for Autonomous Driving Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.

[80] U. Technologies, "Using TensorBoard to Observe Training," *Web resource https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Using-*

Tensorboard.md, 2019.

[81] Aureliantactics, "Understanding PPO Plots in TensorBoard," *Web resource https://medium.com/aureliantactics/understanding-ppo-plots-in-tensorboard-cbc3199b9ba2*, 2018.