CHALMERS
UNIVERSITY OF TECHNOLOGY

Inverse Driver Learning and Short Time Prediction in Freeway Traffic Situations

Master's thesis in Engineering Mathematics and Computational Science

ADITYA B. SRIDHARA

Department of Mathematical Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

MASTER'S THESIS 2019:EENX30

Inverse Driver Learning and Short Time Prediction in Freeway Traffic Situations

Surrounding vehicle motion prediction using Inverse Reinforcement Learning

Aditya B Sridhara

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Inverse Driver Learning and Short Time Prediction in Freeway Traffic Situations
Aditya B. Sridhara

Supervisor: Martin Sanfridson, AB Volvo
Examiner: Balázs Adam Kulcsár, Chalmers University of Technology

Master's Thesis 2019:EENX30
Department of Mathematical Science
Division of Mathematics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 70 465 8729

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Inverse Driver Learning and Short Time Prediction in Freeway Traffic Situations
Aditya B. Sridhara
Department of Mathematical Science
Chalmers University of Technology

Abstract

Traffic prediction around the vehicle during highway driving is a hard task to solve as the driver's decision are complicated. It is necessary to build a procedure to predict driving decisions, such as lane change, overtaking and many other scenarios around the vehicle. Further, building an accurate prediction model is a requisite to improve the design and validation of Automated Driving Systems (ADS).

To solve the prediction task, the project aims to build a Deep Inverse Reinforcement Learning (DIRL) model to analyze and learn diverse driving behaviour during lane change scenarios. DIRL learns this behaviour by undergoing apprenticeship training from an expert. It learns the reward structure of drivers decision during lane change scenarios from such varied expert driving behaviour, and by utilizing these reward it predicts driving decision.

DIRL model is built upon a sequential time series model and General Adversarial Imitation Learning (GAIL). GAIL assists in learning the rewards and future trajectories are predicted using the sequential model. The model is optimized using reinforcement learning techniques by performing policy gradients on the rewards, that are obtained from the GAIL. The model is trained on expert naturalistic driving data recorded on German highways called HighD. The model predicts lateral, longitudinal position and velocity of surrounding vehicles. The accuracy of these predictions is evaluated by comparing these trajectories with expert data.

Keywords: Trajectory prediction, deep learning, Seq2Seq, GAN, inverse reinforcement learning, MDP.

Acknowledgements

I thank Volvo Truck Technology especially the Vehicle Automation team for providing interesting opportunity and infrastructure to conduct this thesis. I like to express my deepest gratitude to supervisor Martin Sanfridson and examiner Balázs Adam Kulcsár, for providing continuous guidance throughout the course of the thesis.

Aditya B. Sridhara , Gothenburg, April 2019

Contents

List of Abbreviations **xi**

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Thesis Objective . . . 2
1.2 Research questions . . . 3
1.3 Scope and Limitations . . . 3
1.4 Outline . . . 3

2 Theory **5**
2.1 Deep learning . . . 5
2.1.1 Recurrent Neural Network . . . 5
2.1.2 Sequence to Sequence . . . 8
2.1.3 Generative Adversarial Network . . . 9
2.2 Inverse Reinforcement Learning . . . 11
2.2.1 Reinforcement Learning . . . 11
2.2.2 Policy Gradient . . . 14
2.2.3 Inverse Reinforcement Learning methods . . . 14

3 Methods **19**
3.1 Problem formulation . . . 19
3.2 Data . . . 19
3.3 Data processing . . . 23
3.4 Deep Inverse Reinforcement Learning Model . . . 33
3.4.1 DIRL Model . . . 33
3.4.2 Model training . . . 35
3.5 Software library . . . 37

4 Results and Discussion **39**
4.1 Longitudinal features . . . 39
4.2 Lateral features . . . 41
4.3 Discussion . . . 43

5 Conclusion and Future Work **45**
5.1 Conclusion . . . 45

5.2 Future Work 46

List of Abbreviation

ADS Automated Driving Systems. v, 1, 2, 19, 45
ANN Artificial Neural Network. 5

BPTT Back-propagation Through Time. 6

DIRL Deep Inverse Reinforcement Learning. v, 5, 39–42, 44, 45

GAIL Generative Adversarial Imitation Learning. 17
GAN Generative Adversarial Network. 5, 45, 46

IL Imitation Learning. 11
IRL Inverse Reinforcement Learning. 11

LSTM Long Short Term Memory. xiii, 6–8

MaxEnt IRL Maximum Entropy Inverse Reinforcement Learning. 16
MDP Markov Decision Process. 11

NLP Natural Language Processing. 33

RL Reinforcement Learning. 11, 13, 14
RMSE Root Mean Squared Error. 40, 42
RNN Recurrent Neural Network. 5, 6

Seq2Seq Sequence to Sequence Network. 8, 9
SGD Stochastic Gradient Descent. 9, 10

TBPTT Truncated Back-propagation Through Time. 6

List of Figures

1.1 Highway driving scenarios, vehicle 1 (ego vehicle) making a lane change, ADAS of vehicle 1 has to predict the behaviour of vehicle 2 (surrounding vehicle) to make safe lane change. 2

2.1 Single neuron of recurrent neural network [1]. 6
2.2 Unrolling of a recurrent neural network through time for training [1] . . . 6
2.3 Schematic unit of Long Short Term Memory (LSTM) consisting of four components and three gates to control the flow of information. Where rectangles σ and tanh refers to sigmoid and hyperbolic tangent activation respectively and × is Hardamard product and + is addition are vector operation in the cell [1][2] 7
2.4 Seq2Seq model architecture 8
2.5 Graphical representation of GAN 10
2.6 Reinforcement Learning [3] 12

3.1 Data recorded using drone hovering above the highway that can capture the traffic within the 420 m highway segment [4]. 20
3.2 Road segment visualization for HighD datset. The red box are computer vision algorithm processed bounding box representing the vehicles on the highway. The black triangle on the bounding box denotes the driving direction. Each vehicle has unique yellow label consisting the type of vehicle car or truck(C or T), vehicle velocity and vehicleId [4]. 21
3.3 Road coordinate system. The vehicle in the upper lane move from right to left. The vehicles in this lane have x values decreasing as the vehicle moves from right to left and velocity of vehicle is negative in reference to the coordinates. It is converse in the lower lane. 23
3.4 Illustration of _tracks feature considering ego vehicle 4 and 10 in upper and lower lane respectively 24
3.5 The lane change trajectories are shifted to the centre frame, with ego vehicle's 2 and 4, lane change maneuver starting at 210m 28
3.6 The figure illustrates the flip transformation on the data set. The ego vehicle 2 have their lane change maneuvers mirrored along the centre of the road. This results in vehicle trajectories in upper lane having the same moving direction as in the lower lane 29
3.7 The expert trajectories, with ego vehicle in the lane 3 and 6, are mirrored along the lane marking in each of the lanes such that their lane maneuver starts from 2 and 5 respectively. It results in positive y values with time t . 30

3.8 Expert trajectories in the lower lane are shifted to upper lane maintaining the same direction as in the lower lane 31
3.9 The figure illustrates the the complete data transformation's undertaken by the expert data 32
3.10 DIRL network architecture 34

4.1 The plots of longitudinal position and velocity of 4.5s predicted trajectory 41
4.2 Plots of lateral position and velocity of 4.5s predicted trajectory 42

List of Tables

3.1 Significant attributes of HighD dataset 21
3.2 Table shows the description of each recorded video in term of CSV file. The XX in the filename represents the number of the recording ranging from 01 to 60 21
3.3 RecodingMeta file description 22
3.4 tracksMeta file description 23
3.5 Table describing the features of the tracks data 24
3.6 Expert Data features for the DIRL model 31

4.1 Mean and standard deviation of the complete 4.5s predicted trajectories for the longitudinal features 39
4.2 Evaluation of the RMSE of the predicted trajectories for the longitudinal features. RMSE is evaluated for 3 time intervals for the 4.5 s trajectory . . . 41
4.3 Mean and variance of the sampled predicted trajectories for lateral features 42
4.4 Evaluation of the RMSE of the predicted trajectories for the lateral features. RMSE is evaluated for 3 time intervals for the 4.5 s trajectory 43
4.5 Mean and variance of the sampled predicted trajectories for lateral features 43

1

Introduction

Demands for goods and services increase around the world, the main catalyst to service this demands are road transport systems. It may be freight transport or passenger transport[5]. In one of the study it is suggested that transport system is one of the indicators for economic growth and created around 12 million jobs [6]. It is seen that the road transport in the world has been increasing steadily, one of the reasons lie in the disruptive technology like autonomous driving that is changing the transport industry radically by reducing the transport cost making goods cheaper .

Taking into account of the increasing road transport, it necessary to have an efficient and safe transport system. Though its seen that EU roads are one of the safest, accidents on these roads haven't abated as envisioned by European commission [7]. The road transport accounts to, as of year 2017, around 135,000 people seriously injured, and 25,300 people lost their lives.This is estimated to about 70 lives per day and around 120 billion Euro loss per year for Europes's GDP. The European commission road safety policy set reduce this road fatalities by halve by the year 2030 [7].

The measure taken by the European commission is to focus on the vehicle safety that would consecutively increase the standards of road safety. It is found that 94 percentage of accidents can be attributed to human error or human judgment[8]. So to enhance vehicle safety it is imperative to provide integral technical systems for assistance to human drivers. These systems must include crash avoidance system, crash mitigation and crash protection systems for a robust vehicular safety. ADS were built as a precautionary measures in consideration for increased road safety. ADS are proven in providing positive solution in reducing the road fatalities[9]. ADS comprises of systems like Lane Keeping Support systems, Path Planning and Trajectory Predictions.

The major challenges to achieve higher autonomy in ADS is to estimate and predict human intention of surrounding vehicles.The current master thesis project is to provide plausible solution in this direction, to predict human intention around host vehicle.

1.1 Thesis Objective

A momentous task for the ADS is to predict the behaviour or intentions of surrounding vehicles. It is difficult to predict driving intentions of surrounding vehicles, which may include overtaking, changing lane, exiting the highway or taking a turn. All such diverse scenarios arising are numerous and it is difficult to develop a single prediction framework that can encompass all such driving scenarios.

One such driving situation that is largely considered for the study in the current thesis is lane change behaviours. Figure 1.1 shows a situation arising on highway frequently, vehicle 1 called the ego vehicle makes a lane change, to exit the highway. The ego vehicle has to judge the surrounding vehicle 2 motion as the the surrounding vehicle may or may not allow the ego vehicle to pass through. It is imperative to predict the surrounding vehicle motion to have a safe lane change maneuver for the vehicles on the highway. The ego vehicle, a truck, has to consider whether to change lane or not based the surrounding vehicle speed, whether it can change lane safely considering its trailer's length. The thesis considers all the lane change behaviour arising all the type of vehicles, car or truck, for either ego or surrounding vehicle.

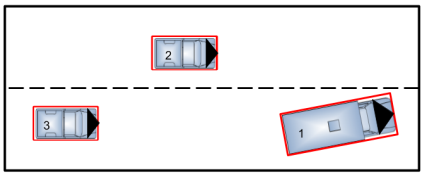


Figure 1.1: Highway driving scenarios, vehicle 1 (ego vehicle) making a lane change, ADAS of vehicle 1 has to predict the behaviour of vehicle 2 (surrounding vehicle) to make safe lane change.

Prediction of surrounding vehicles during lane change is of importance when there is a high density of the traffic on the highway. Analyzing and predicting all the driving behaviour of the ego and surrounding vehicle during lane change would in turn help in negating crashing scenarios.

The aim of the thesis is to provide a suitable solution in predicting the trajectory and motion of the surrounding vehicle by imitating driving behaviours. The end goal is to build a robust model efficient to predict diverse human driving behaviours by predicting vehicular motion, so that it aids in prescribing certain control strategies to ADS.

1.2 Research questions

The master thesis aims at addressing the following question

- How efficiently can Deep Inverse Reinforcement Learning algorithms can predict vehicles behaviour?
- How are the features of the surrounding vehicle mapped to get the optimal reward in inverse reinforcement learning ?
- Can the model predict the future driving behaviour even after the lane change based on the output of the inverse reinforcement learning algorithm?

1.3 Scope and Limitations

The thesis is limited to working on the naturalistic driving data set called HighD [4]. The data contains trajectory of vehicle on straight German highway possessing a simple lane change maneuvers and void of any junction crossing, and lane merging scenarios. Thus our prediction model is built for the lane change maneuvers. As the concept of inverse reinforcement learning is quite novel, to analyze the efficiency of this prediction model clearly, this thesis is restricted to predicting driving intention for two lane, two agent or vehicle lane changing scenarios.

1.4 Outline

The section below is a brief outline for the forthcoming chapters of thesis report.

1. **Theory** : In this section we delve upon theoretical concepts required to build driver behaviour prediction or traffic prediction model. The chapter is mainly divided into two major sub topics namely deep learning methods an inverse reinforcement learning. The concepts concerned to this project are elucidated, along with its underlying mathematical principles.
2. **Methodology**: The chapter initially describes the naturalistic driving data and pre-processing techniques. The chapter presents a step wise formulation in building inverse reinforcement based traffic prediction model for highway lane change scenarios. It also elucidates the complete inverse reinforcement algorithm built for vehicle trajectory prediction. Furthermore, the chapter contains a brief overview of the software libraries utilized in the thesis.
3. **Result**: It presents the experimental results and the result analysis of the deep inverse reinforcement prediction model.

4. **Conclusion and Future work**- The experimental results are contextualized with the objective of the thesis. The chapters also expands on the future course of direction in which the model needs to be investigated.

2

Theory

In the following section, concepts required for DIRL formulation are explained in detail. It comprises of a comprehensive review of the theory along with a detail elaboration of its mathematical framework. The chapter is divided into two topics, deep learning and inverse reinforcement learning methods respectively.

2.1 Deep learning

Deep learning is sub-field of machine learning that solves representation learning task using successive smaller and simpler layered representation of data. Therefore it also named as layered representation learning or hierarchical representation learning. This layer representation helps the computer to understand complex concepts such as images and speech. The layered representation learning is generally modelled using Artificial Neural Network (ANN)[10]. ANN are computational graphs and work as functional approximators $f(x,\theta)$, for input data x. The objective of the network is to calculate the output y, by calibrating its parameter θ. For a thorough explanations and understanding of ANN and its mathematical framework, readers are encouraged to go through [10]. In the following subsection we elaborate on Recurrent Neural Network (RNN), Seq2Seq and Generative Adversarial Network (GAN).

2.1.1 Recurrent Neural Network

The functional approximation done by simple ANN for sequential data is very ineffective as ANN assumes the data to be independent identically distributed and uncorrelated. RNN is a class of neural network that specializes in processing such sequential data like speech, music, language and other time series data efficiently [11, 12]. RNN is constructed by inserting feedback loop over each neuron. This enables the network to share parameters across time and establish a recurrent mechanism [13]. The output of RNN depends on the input fed to the network and the output at previous time step.

RNN takes in an input sequence vector $x = (x_1, \ldots, x_t)$ and approximates this sequential data to output $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_t)$ at each time step t. RNN has a layer of hidden states h_t that summarize the previous states of the input until the current time step.

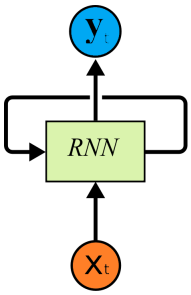


Figure 2.1: Single neuron of recurrent neural network [1].

This layer forms the feedback loop that helps to share information from previous states and forms the recurrent mechanism[13]. At each time step t, RNN simultaneously updates the hidden state and produces an output, based on the current input x_t and the hidden state at previous time step h_{t-1}. The hidden state of the cell are updated according to equation 2.1 and the output of RNN is given by equation 2.2.

$$h_t = f(W_i x_t + W_h h_{t-1} + b_h) \tag{2.1}$$
$$\hat{y}_t = g(W_o h_t + b_o) \tag{2.2}$$

where $f(x)$ and $g(x)$ are activation functions, W_i, W_h, W_o are the weight matrix of the input, hidden and output layers respectively and b_h, b_o are the biases.

Training of RNN is done with the algorithms called Back-propagation Through Time (BPTT)[14] and updated version Truncated Back-propagation Through Time (TBPTT) [15]. The algorithm trains the network through generalized back-propagation after it unrolls the network in the given time T, as shown in Figure 2.2.

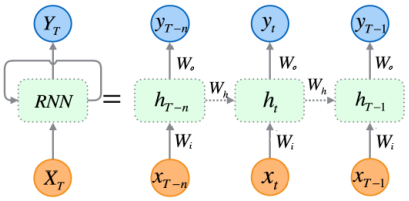


Figure 2.2: Unrolling of a recurrent neural network through time for training [1]

Long Short Term Memory Networks

Despite RNN being utilized to approximate sequential data, it has problem such as vanishing gradient [16] and cannot learn long term dependencies in the data [17]. To overcome these problems LSTM was proposed [18] and was upgraded by including the weight condition of the LSTM [19]. LSTM is a type of where the hidden layer are replaced by LSTM cell, that has a self loop so that gradient can pass through without exploding. There are three gates namely input, forget and output that controls the flow of information to and within the memory cell. During training, LSTM learns to control

the gates which changes the weights of the memory cell. The basic structure of the LSTM cell is shown in Figure 2.3.

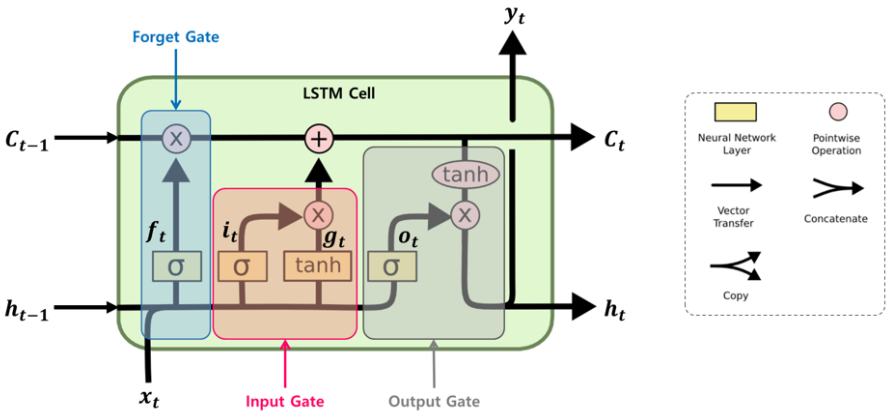


Figure 2.3: Schematic unit of LSTM consisting of four components and three gates to control the flow of information. Where rectangles σ and tanh refers to sigmoid and hyperbolic tangent activation respectively and $\times$ is Hardamard product and $+$ is addition are vector operation in the cell [1][2]

.

LSTM consist of four components that gives an output of cell state C_t and hidden state h_t at each time. The forget gate f_t,introduce by Schmidhuber et al [19] helps to decide the elements of cell state to be passed. Forget gate outputs a vector of 0 or 1, when combined with cell states it can decide the elements to be kept for the current time. The output of the forget gates is given by Equation 2.3. The input gate i_t decides the elements that would be contributing to next cell as shown in Equation 2.4. The output gate o_t of the LSTM cell filters the the elements of the current cell state evaluated by Equation 2.5.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.3}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.4}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.5}$$

The cell state an the hidden state are updated and stored in the memory cell for next time step given by the list of Equation 2.6, 2.7, 2.8.

$$\hat{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.6}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t \tag{2.7}$$
$$h_t = o_t \odot \tanh(C_t) \tag{2.8}$$

where

- W_f, W_i, W_o, W_c are the input weights
- U_f, U_i, U_o, U_c are the recurrent weights

- b_f, b_i, b_o, b_c are the biases
- $\odot$ denotes element-wise multiplication
- σ and tanh are the activation functions

2.1.2 Sequence to Sequence

LSTM is used for modelling sequential data efficiently but it takes in data of fixed input length and generates an output of fixed length. Many real world application like machine translation, speech recognition and time series forecast, requires network to be flexible so as to take input of variable length and produce an output of variable length. Sequence to Sequence Network (Seq2Seq) [12] model is the neural network architecture built using LSTM to overcome the shortcomings of simple LSTM network.

A Sequence to Sequence architecture has two components, an encoder and decoder. The encoder takes input $X = \{x_1 \dots x_t\}$ with variable length and converts into a context vector v which is fed to the decoder. The decoder with use of this context vector gives an output $Y = \{y_1 \dots y_T\}$ of required variable length.The figure shows a basic architecture of the Seq2Seq model.

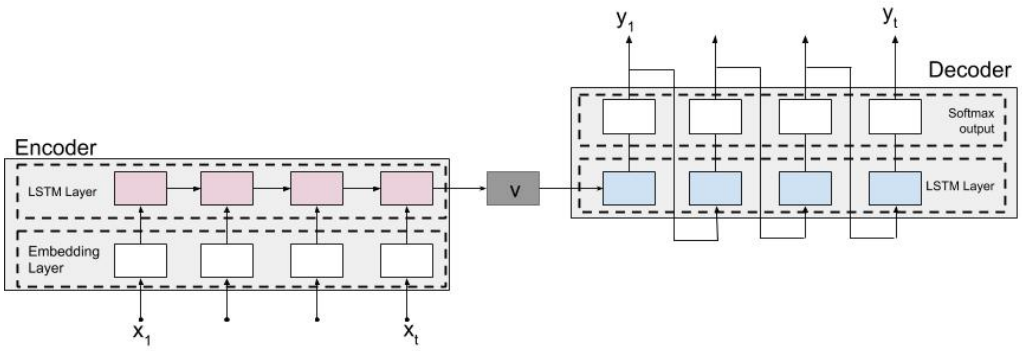


Figure 2.4: Seq2Seq model architecture

The encoder consist of two layer of neural network, embedding layer and LSTM layer respectively to which sequential data $X = \{x_1,,,x_t\}$ is fed. Embedding layer of the encoder generalizes this high dimensional data to low dimensional data by categorizing data into row of an embedding matrix with columns length as the total vocabulary size. Then this embedded data is encoded into on fixed representation context vector v using the LSTM layer.

The decoder consist of layer of LSTM layer with softmax output. It produces a conditional probability of the output given the input sequence Equation 2.9 [12]. The output of the decoder from the time $t-1, y_{t-1}$ is fed as input to the decoder at current

time t to produce an output y_t

$$P(\hat{y}_1 \dots \hat{y}_t / x_1 \dots x_t) = \prod_{t=1}^{T} P(\hat{y}_t / v, \hat{y}_1, \dots, \hat{y}_{t-1}) \tag{2.9}$$

where $v = (x_1 \dots x_t)$, $\hat{y}_t$ is predicted output

$$P(\hat{y}_t / v, \hat{y}_1, \dots, \hat{y}_{t-1}) = \frac{\exp(w^\top f(\hat{y}_t, \hat{y}_{t-1}, x_{1:t})}{\sum exp(w^\top f(\hat{y}_t, \hat{y}_{t-1}, x_{1:t})} \tag{2.10}$$

Training

Output of the Seq2Seq model is a probability density on the total vocabulary length. But the actual value of the output is evaluated using the Equation 2.11

$$y_t = \arg\max_{y \in Y} P(\hat{y}_t / v, \hat{y}_1, \dots, \hat{y}_{t-1}) \tag{2.11}$$

The loss at each time step t is calculated using the cross entropy with the true output density. The loss is calculate using the Equation 2.12. The decoder and encoder are updated using Stochastic Gradient Descent (SGD) [20]

$$\nabla \mathcal{L}_\theta = -\frac{1}{N} \sum_{t=1}^{T} P(y_t) \nabla_\theta \log P(\hat{y}_t) \tag{2.12}$$

2.1.3 Generative Adversarial Network

Generative Adversarial Network (GAN) is a differentiable generator network, where the data is transformed from latent space using a differentiable function [10, 21]. It was initially developed to generate realistic samples of data with the given distribution $p(x)$ [21]. GAN has been used in various field in image translation, sequence generation and model free reinforcement learning task [22, 23, 24].

GAN consists of two networks called *Generator G* and *Discriminator*, *D* respectively. The generator produces synthetic data samples similar to distribution of training data. The discriminator's objective is to examine these sample from the generator and distinguish them whether they are real or fake. Intuitively the generator can be considered as the counterfeiter trying to fool the discriminator by making fake money. Then the discriminator is considered as the police trying to distinguish whether currency is real or counterfeit [25].

Generator G is a neural network having a differentiable function $G(z;\theta^G)$ w.r.t to input noise z and with parameter θ^G. It samples the data from prior latent space with distribution $p_z(z)$ and tries to minimize its cost function $\mathcal{L}_G(\theta^G)$. Similarly discriminator is a neural network, that has a differentiable function $D(x;\theta^D)$ w.r.t to input and the observed samples x with network parameter θ^d. The output of the discriminator is

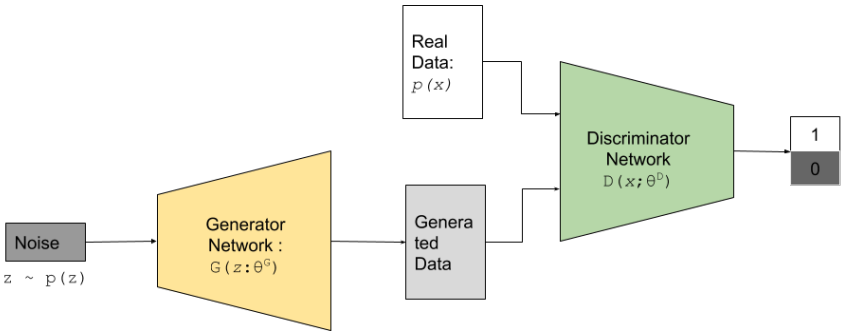


Figure 2.5: Graphical representation of GAN

scalar value representing the probability whether the data x comes from the original distribution $p(x)$ or not, with cost function as $\mathcal{L}_D(\theta^D)$.

GAN is considered as two agents G and D playing a min-max game having a total payoff with value function $\mathcal{V}(\theta^G,\theta^D)$. Solving the minimax game is equivalent to optimizing the generator parameter given by

$$\theta^{G^*} = \arg\min_{\theta^G}\max_{\theta^D}\mathcal{V}(\theta^G,\theta^D) \tag{2.13}$$

The value function is nothing but the $\mathcal{V}(\theta^G,\theta^D) = \mathcal{L}_G(\theta^G) = -\mathcal{L}_D(\theta^D)$. The mathematical formulation of the cost function is given as the *binary cross entropy* defined by the Equation 2.14 [21]

$$\mathcal{V}(\theta^G,\theta^D) = \mathbb{E}_{x\sim P_r}\log D(x) + \mathbb{E}_{z\sim P_g}\log[1-D(G(z))] \tag{2.14}$$

where x is the input data with distribution $\mathbb{P}_r$, z are the samples from prior distribution $P(z)$ generally a Gaussian, and P_g distribution form the output of generator. The loss function of discriminator is maximized and the loss function for the generator is minimized such that P_g converges to P_r, ie $P_r = P_g$. In this min max game the discriminator minimize the cross entropy while the generator tries to maximize this cross entropy [25].

Training

Training the network is done in two steps but iteratively with alternatively gradient descent, where in discriminator is trained first and then generator trained next. Two batches of samples are drawn from P_r and from latent variable $\tilde{x}$ distribution $G(z) \sim P_g$, which are fed to the generator G. The output of the generator fed to the discriminator, yielding a probability P_r. The discriminator loss are calculated and updated using the SGD [20, 26],

$$\nabla_{\theta^D}\mathcal{V}^D(\theta^G,\theta^D) \quad = \quad -\frac{1}{N}\sum_{n=1}^{N}\left[\nabla_{\theta^D}\log D(x^n) + \nabla_{\theta^D}\log(1-D(\tilde{x}^n))\right] \tag{2.15}$$

Once the discriminator is updated which tries to minimize the cross entropy, new samples $\tilde{x}$ are drawn from P_g and the generators gradients are updates by Equation 2.42

$$\nabla_{\theta^G} \mathcal{V}^G(\theta^G, \theta^D) = \nabla_{\theta^g} \mathcal{L}_G(\theta^G) = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\theta^G} \log(1 - D(\tilde{x}^n)) \tag{2.16}$$

2.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a subclass of techniques of Imitation Learning (IL), where in the the agent learns the optimal behaviour from the set of demonstration of experts. These are methods developed initially to program autonomous robots to perform highly complex task like pick and place [27, 28], robot path planning [29], helicopter maneuvering [30] which had very successful outcomes. The distinction between IRL and imitation learning is that while IRL objective is to only infer the reward function from the expert where as IL goal is to mimic the expert behaviour. IRL is closely related to supervised learning where in prediction task are converted into sequential decision task [31, 32]

Early stages of development of IRL were done by Kalman 1960, where in he recovered the objective function for deterministic linear system with quadratic cost [33]. The use of Markov Decision Process (MDP) frame work to recover the rewards was first done by Russell et al [34], wherein the agent task is modeled as sequence of decision making process, which then can be solved using reinforcement learning techniques [35]. Since then the IRL has gain traction and advantages in using the MDP framework to model the environment as it can be solved using Reinforcement learning techniques. The concept of IRL requires a basic understanding of reinforcement learning techniques for MDP. The following section is a reference to concepts in reinforcement learning and IRL

2.2.1 Reinforcement Learning

Reinforcement Learning (RL) is described as a learning framework, where an agent or learner interacts with the environment to achieve a goal. Unlike a supervised learning where the learning is passive, RL is active as it learns from the reward signal received by the environment continuously. Reward signal is the closed loop feed back received to the agent from interaction between the agents and the environment that occur in discrete time. The goal or the learning behaviour can be represented in form of feed back signal in-between agent and the environment, where one signal represent the choice of the agent (actions), next signal represent the signal based on which the choices are made (states) and the reward signal based on the agent goal(reward) as shown in Figure 2.6

Environment constitutes a situation or scenario in which the agent acts and learns to achieve a specific goal. States s_t are some representation of a complete or all possible

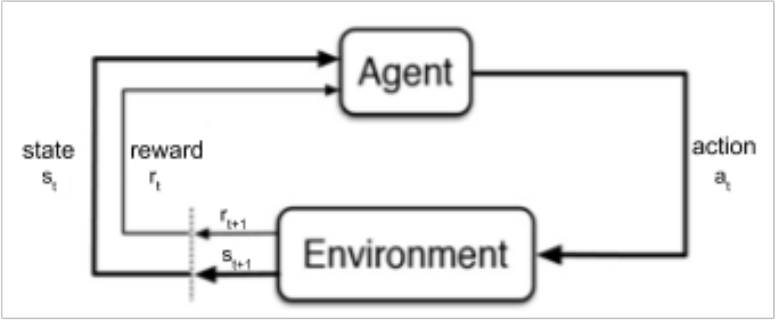


Figure 2.6: Reinforcement Learning [3]

environment state $\mathcal{S}$ that agent receives at each time t, $s_t \in \mathcal{S}$.Based on the states the agent takes a control decision, these are called actions $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ set of action that agent can take at particular state s_t.

Rewards r_t, is a scalar value, it is the incentive the agent receives at a particular state after choosing an action $\mathcal{A}(s_t)$. The agent's goal is to maximize these expected reward R_t, a function defined as a sum of rewards it may receive next from time $t+1$ until the length of the that episode T.

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T = \sum_{k=0}^{T} r_{t+k+1} \tag{2.17}$$

In other words it quantifies how well the agents have performed the task from the current state at time t to the end state T over the each episode [3]. If we consider a continuous control task, that doesn't have a terminal state or do not have an end goal i.e $T = \infty$ the Equation 2.17 does not converge. A discount factor $\gamma \in [0,1]$ is introduced to rectify this problem [36, 37], then the Equation 2.17 can be re written as

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.18}$$

If the agent has $\gamma = 0$ discount factor, it's goal is to choose an action to maximize immediate rewards, on the other hand if the discount factor $\gamma \to 1$ it chooses actions a_t so as to consider long term rewards.

Markov Decsion Process

The Markov Decision Processes (MDP) is a mathematical formalism suitable for modelling and solving decision making or real world control problems. MDP is defined [3] as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S}$ is set of all possible states of environment, satisfying the Markov property

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, s_2, \ldots, s_t) \tag{2.19}$$

- $\mathcal{A}$ is set of actions taken by agent in the state s_t
- $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is the state transition probability,

  $\mathcal{P}(s'|s,a) = P(s_{t+1}|s_t, a_t)$, where s' is the successive state, and s, a are current state and action respectively
- $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function

  $\mathcal{R}(s,a) = \mathbb{E}[r_{t+1}|s_t, a_t]$
- γ is the discount factor $\in [0,1]$

Policy

Policy, π is the characteristic behaviour of the agent. A stochastic policy is defined as probability of choosing an action from the current state, more formally agents policy is the distribution of action over given states as shown in Equation 2.20. The goal of RL is find the optimal policy π^* that acquires the maximum expected rewards.

$$\pi(s_t) = P(a_t|s_t) \tag{2.20}$$

Value functions

State value function, $V^\pi(s)$, is defined as the expected return of the state s, and then following the policy π. It signifies "how good" the current state is [3].

$$V^\pi(s) = \mathbb{E}_\pi[R_t|s_t] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t] \tag{2.21}$$

Action value function, $Q^\pi(s,a)$, is similar to the state value function, but considers the action taken in the current state. It is defined as the expected return from the state s , taking the action a, and then following the the policy π [3].

$$Q^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t, a_t] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t, a_t] \tag{2.22}$$

Optimal Value function

The optimal value function are the expected values following the optimal policy π^*, wherein the policy π^* has the maximum excepted rewards [3]. The optimal value function $V^*(s_t), Q^*(s_t, a_t)$ are given as

$$V^*(s_t) = \max_\pi V_\pi(s_t) \tag{2.23}$$

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t) \tag{2.24}$$

$$\pi^*(a_t|s_t) = \arg\max_a Q_\pi(s_t, a_t) \tag{2.25}$$

2.2.2 Policy Gradient

The policy gradient are optimizing methods for RL, that uses gradient ascent on the policy parameter to obtain an optimal policy which subsequently maximizes reward. Policy gradient methods are suited for agents having continuous action spaces or stochastic policy. We use a functional approximator to estimate the policy, that has a parameter θ, then the policy can be defined in terms of the parameters as $\pi_\theta(a/s)$. The quality of the the policy is quantified in terms of the objective function $\mathcal{J}$. It yields the expected reward for the current policy. It is mathematically defined for continuous action space as

$$\mathcal{J}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[r(\tau))] = \int \pi_\theta(\tau) r(\tau) d\tau \tag{2.26}$$

where τ is the trajectories of the agent, and it can be decomposed into product of the conditional probabilities

$$\pi_\theta(\tau) = \pi_\theta(s_0, a_0, s_1, a_1, \ldots s_T, a_T) = P(s_1) \prod_{t=1}^{T} \pi_\theta(s_t) P(s_{t+1}|s_t, a_t) \tag{2.27}$$

we perform the gradient ascent on the objective function to optimize θ such that it yields the highest expected rewards

$$\nabla_\theta \mathcal{J}(\theta) = \int \nabla\pi_\theta(\tau) r(\tau) d\tau \tag{2.28}$$

we know that

$$\nabla_\theta \pi_\theta(\tau) = \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) \tag{2.29}$$

Therefore, substituting the values of Equation 2.29 to Equation 2.28, we obtain the policy gradient as

$$\nabla_\theta \mathcal{J}(\theta) = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = \mathbb{E}_{\tau\sim\pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right] \tag{2.30}$$

2.2.3 Inverse Reinforcement Learning methods

RL is a technique to learn a behaviour based on rewards. Robustness of the learning algorithm and the objective of the task of RL is succinctly defined upon the rewards. In the real world task rewards can't be specified to the learning task manually [30]. In such cases and most apparent cases we only know the trajectories of the experts and intend to learn these behaviour. IRL is defined "apprenticeship learning to acquire skilled behaviour and to ascertain the reward function of the expert" [34]. If reward structure is learnt, the problem is reduced to finding the policy that can be solved by RL algorithms.

The IRL considers that expert trajectories are output of the MDP. The notation of MDP are mentioned in the earlier section and to derive solutions for IRL problem the author maintains the same notation for consistency. Russell et al [34] established that it is possible to select the best reward function that has the optimal policy. The reward function for the optimal policy can be selected using Equation 2.31, which satisfies the condition 2.33. For a finite state MDP this equation can be solved using linear programming [34].

$$a_n \equiv \pi(s) \in \arg\max_{a \in \mathcal{A}} \sum_{s'} \mathcal{P}(s^{'}) V^{\pi}(s^{'}) \tag{2.31}$$

$$V^{\pi} = (I - \gamma \mathcal{P}_{a_n})^{-1} \mathcal{R} \tag{2.32}$$

for

$$(\mathcal{P}_{a_n} - \mathcal{P}_a)(I - \gamma \mathcal{P}_{a_n})^{-1} \mathcal{R} \geq 0 \tag{2.33}$$

Where $\pi(s) \equiv a_n$ is the unique optimal policy, $a \equiv \mathcal{A} \setminus a_n$ for action set $\mathcal{A} = a_1, a_2, \ldots, a_k$

Feature Expectation Matching

The concept of feature expectation matching was introduced by Abbeel et al [30] and subsequently implemented by Ziebart et al [38]. It can solve even infinite state space IRL problems, where we assumes the reward function as a linear function of features [34].

$$r(s) = w^{\top} f(s) \tag{2.34}$$

$f(s)$ is the feature vector of the state $\mathcal{S}$ and the w is the weight vector. The feature represent the state space of the agent at a given time. The feature vector depends on the agent and the task considered. The value function for the policy π

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t)|\pi] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t w^{\top} f(s_t)|\pi] = w^{\top} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t f(s_t)|\pi] \tag{2.35}$$

For a given expert, trajectories/demonstration τ, *feature expectation* $\mu(\pi)$ is defined on the policy π as

$$\mu(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t f(s_t)|\pi] \tag{2.36}$$

Therefore, the expected value is the linear combination of feature expectation

$$\mathbb{E}[R|\pi] = w^{\top} \mu(\pi) \tag{2.37}$$

The feature matching algorithm maximizes the reward function by matching the feature expectation with the experts feature visitation.

Feature expectation matching is not a robust algorithm as it can map many policies to a single reward function and it can be ambiguous when the demonstration is sub optimal [39].

Maximum Entropy Inverse Reinforcement Learning

Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL), is advanced method to solve IRL problem leveraging the principles of maximum entropy [40]. The algorithm selects a distribution that maximize the entropy that matches the feature expectation of the expert [41, 38]. In MaxEnt IRL the reward is a linear mapping of sum of the feature counts,

$$R(\tau) = w^\top f(\tau) = \sum_{s_j \in \tau} w^\top f(s_j) \tag{2.38}$$

- where, $f(\tau) = \sum_{s_j \in \tau} f(s_j)$ is the feature counts of the trajectory
- $f(s_j) \in \mathbb{R}^k$ feature at each state j
- w is the parameterized weights

The agent/learner, learns the policy distribution $\pi(\tau)$ that maximizes the feature expectation of the expert, having the constraints

$$\mathbb{E}_{\pi \sim L}[f(\tau)] = \mathbb{E}_{\pi \sim E}[f(\tau)], \quad for \quad \sum_{\tau} \pi(\tau) = 1, \quad \forall \tau, \pi(\tau) = 1 \tag{2.39}$$

where

- $\mathbb{E}_{\pi \sim L}[f(\tau)]$ expected feature count on the learners policy
- $\mathbb{E}_{\pi \sim E}[f(\tau)]$ expected feature count on the expert policy

The probability density that satisfies Equation 2.39 is Equation 2.40. According to this the optimal trajectories have the highest likelihood and the sub-optimal path generated by the expert decreases with exponential probability.

$$\pi(\tau|w) = \frac{exp(w^\top f(\tau))}{Z(w)} \prod_{s_{t+1}, s_t, a_t \in \tau} P(s_{t+1}|s_t, a_t) \tag{2.40}$$

$Z(w) = \sum_\tau exp(w^\top f(\tau))$ is the partition function. The parameter w is obtained by maximizing the likelihood $\mathcal{L}(w)$ under the maximum entropy density for the observed data.

$$w^* = \arg\max_w \mathcal{L}(w) = \arg\max_w \sum_{\tau_i} \log \pi(\tau|w) \tag{2.41}$$

w is updated using the gradient descent on the objective function $\mathcal{L}(w)$, which is the difference between the expert's features counts and the learners expected feature counts, given by Equation 2.42.

$$\nabla_w \mathcal{L}(w) = \tilde{f} - \sum_{\tau_i} \log \pi(\tau|w) f(\tau) = \tilde{f} - \sum_{s_j} D_{s_j} f(s_j) \tag{2.42}$$

D_{s_j} is state visitation count. Calculating D_{s_j} is time consuming using quadratic programming. Though it is demanding to calculate equation 2.42, it can be calculated using deep neural network [42]. The major drawback of MaxEnt IRL is that it requires a state transition probabilities. In many real world scenarios like robot action, helicopter maneuvers and car lane change it is difficult to obtain these probabilities.

Generative Adversarial Imitation Learning

MaxEnt IRL fails to work when the state transition dynamic isn't specified. Though many iteration and modification to this approached have been undertaken [39, 43, 44, 45], but still all of these require a state transition dynamics. In other words these are model based IRL approaches that requires a model of the state transitions. On the other hand Generative Adversarial Imitation Learning (GAIL) is model free IRL approaches that do not require any state dynamics and can be scaled easily to the very large environments [46]. GAIL is a type of IRL wherein it doesn't specify explicit reward function it learns the behaviour of the experts form its policy directly. But the rewards can be extracted from GAIL which can be used to train similar MDP models.

The proposed method evaluates the best policy by running the RL on the cost function $\mathcal{C}$ based on the maximum casual IRL [39]. The objective of maximum casual IRL is 2.43 and it can be optimized by RL algorithm 2.44

$$IRL(\pi_E) = \max_{c\in\mathcal{C}}\Big(\min_{\pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]\Big) - \mathbb{E}_{\pi_E}[c(s,a)] \tag{2.43}$$

$$RL(c) = \arg\min_{\pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \tag{2.44}$$

where the $H(\pi) = \mathbb{E}_{\pi}[-\log\pi(a|s)]$, is the entropy of the policy π and $\mathbb{E}_{\pi_E}[c(s,a)] = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t r(s_t)|\pi]$ is the cost function

Occupancy measure is described as the distribution of state-action pairs which are generated by the policy π given by Equation 2.45 [46].

$$\rho(s,a) = \pi(s)\sum_{t=0}^{\infty}\gamma^t P(s_{t+1}/s_t, a_t) \tag{2.45}$$

Then the IRL problem can be described based on the occupancy measure, it is the procedure to find the policy that matches the occupancy measure of expert's policy [46]. It can be shown that the objective of IRL can be reduced to dual of an occupancy measure match problem between the inner loop of causal IRL and the outer loop RL ascent as in Equation 2.46.

$$RL.IRL_{\psi}(\pi_E) = \arg\min_{\pi} -H(\pi) + \psi^*(\rho_{\pi} - \rho_{\pi_E}) \tag{2.46}$$

In this imitation learning setup $-H(\pi)$ is called the policy regularizer and ψ^* is the convex conjugate of the regularizer called the cost regularizer. The GAIL is derived

by choosing the optimal cost regularizer that reduces the Janson - Shanon divergence (D_{JS}) [46].

$$\min_{\pi} \psi^*(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) = D_{JS}(\rho_\pi, \rho_{\pi_E}) - \lambda H(\pi), \qquad \forall \lambda > 0 \tag{2.47}$$

The proposed GAIL algorithm draws a connection between IRL and GAN networks, it utilizes the GAN training to fit the expert trajectories. The occupancy measure of π is the data distribution of the generator and the occupancy measure of expert is the true distribution that the discriminator must distinguish. The objective of GAIL is to find the saddle point in the Equation 2.48

$$\mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi) \tag{2.48}$$

GAIL training is described in the Algorithm 1. Initially the weights of the discriminator network are updates based on the generated trajectories. The weights of the generator $\theta_\mathcal{G}$ are updated using Trust Region Policy Optimization(TRPO) [47] based on the policy π.

Algorithm 1 Generative Imitation Learning Algorithm

Input : Expert trajectories τ_E
Initialize parameters θ and w
for *i = 1, 2, ...* **do**
$\tau_i \leftarrow$ sample trajectories from τ_E
$w_{i+1} \leftarrow$ Update the the weight parameters from w_i with gradients

$$\mathbb{E}_{\tau_i}[\nabla_w \log D(s,a)] + \mathbb{E}_{\tau_E}[\nabla_w \log(1 - D(s,a))]$$

$\theta_{i+1} \leftarrow$ Update the policy parameter from θ_i using TRPO

$$\mathbb{E}_{\tau_i}[\nabla_\theta \pi_\theta(s) Q(s,a)] - \lambda H_\theta(\pi_\theta)$$
$$Q(s,a) = \mathbb{E}_{\tau_i}[\log D_{w+1}(s,a) | s_0, a_0]$$

end

GAIL is well suited to imitate complex behaviour in large and high-dimensional environments. Chelsea Finn et al [24] provides this mathematical equivalence between the MaxEnt IRL to the GAN.

3

Methods

This chapter is a detailed elaboration of the methodology implemented to solve driving behaviour prediction. Initially, the chapter formulates the Inverse driver learning objectives in terms of deep inverse reinforcement learning framework. Then chapter deals with the data employed to undertake this task and its preprocessing techniques. Finally, the chapter deals with building the Deep Inverse Reinforcement Models.

3.1 Problem formulation

The goal of the thesis project is to achieve a suitable prediction model of driver's intention, in case of lane change scenarios to help in improving ADAS systems. The prediction models must be robust to predict varied situations. Consider a highway driving scenario the ego vehicle needs to predict the surrounding vehicle intentions, whether the vehicle would allow the ego vehicle to pass, does the vehicle reduce the speed and many such characteristics features of the surrounding vehicle. Our aim is analyze various behavior of the surrounding vehicle during lane change scenarios for prediction.

The ego vehicle, the surrounding vehicle and their interacting behaviour during lane change are considered experts agents. These expert agents behave optimally during the lane change. The expert's behaviour are considered to be MDP, whose output is a time series state trajectories, that are extracted form a reliable data resource. These extracted trajectories are considered to be expert trajectories τ_E. The task is to analyze all these trajectories τ_E of the surrounding vehicle in the data and build a model to predict such driving patterns for future reference to the ADS. So IRL techniques are employed to understand the reward structure of this expert trajectories τ_E. Based on the reward the driving behaviour of the surrounding vehicle are predicted.

3.2 Data

The project requires an experts data so that the model can learn their diverse driving behaviour by performing IRL. There are various open access Naturalistic Driving Dataset (NDD) especially for capturing driving behaviours in traffic flow [48, 49]. KITTI

is a NDD recorded using lasers and cameras mounted on the car, which is mostly used for computer vision application [49, 50]. It is not suitable for our DIRL model as it lacks highway traffic scenarios. The most widely used NDD is Next Generation SIMulation (NGSIM) especially for highway scenarios [48, 51, 52]. NGSIM contains recording of United States highway traffic, collected using digital video cameras mounted on highway. But NGSIM data poses serious challenges while analyzing, as the data is noisy and requires filtering. The major problem of NGSIM is that numerous data points for the position, speed and acceleration are erroneous [53], these data points are impossible to achieve for a vehicle in normal driving scenario. To avoid such problems we consider the much more accurate NDD called HighD [4].

HighD stands for Highway Drone, it is a NDD recorded across German highway using drones. The data was taken at six different highway location around the Cologne area. It was recorded using high resolutions camera mounted on the drone hovering at a fixed position above the highway as in Figure 3.1. This way of recording helps to provide an unbiased data set as the drivers isn't aware of the data being recorded, so their behaviours are uninfluenced and it also provides an aerial perspective of traffic flow. There are several advantages of recording using drones which includes accurate measurement of naturalistic behaviour, data has good static and dynamic description and has better privacy protection [4].

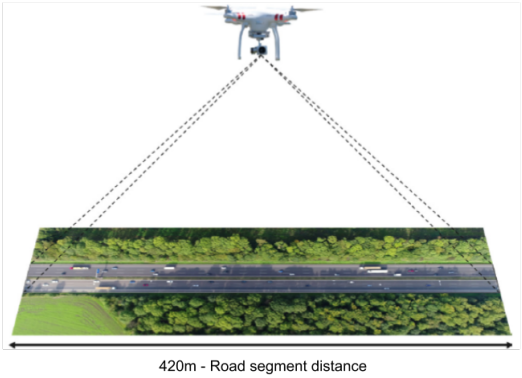


Figure 3.1: Data recorded using drone hovering above the highway that can capture the traffic within the 420 m highway segment [4].

HighD data was recorded using 4k video camera having 25fps. At each recording location, road segment of length 420 meter is considered and a total of 60 video recording with an average duration of 17 minutes per video were recorded. The data can be visualized using the codes provided by the authors of the dataset as shown in the Figure 3.2. Some of the salient features of HighD dataset are mentioned in the Table 3.1

Data description

Video recording are mapped into data points using computer vision algorithm. Each video recording is converted into 3 CSV files containing the data of recorded location,

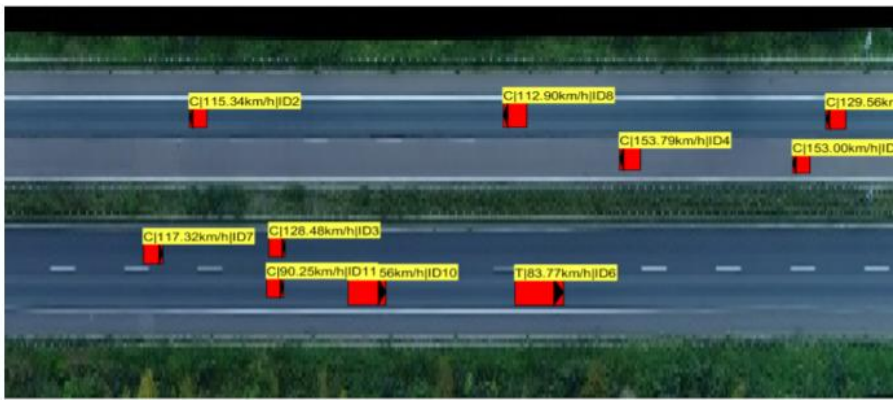


Figure 3.2: Road segment visualization for HighD datset. The red box are computer vision algorithm processed bounding box representing the vehicles on the highway. The black triangle on the bounding box denotes the driving direction. Each vehicle has unique yellow label consisting the type of vehicle car or truck(C or T), vehicle velocity and vehicleId [4].

HighD data	Measurements
Total recording duration in hours	16.5
Lanes per direction	2-3
Total number of vehicles recorded	110,000
Recording distance per location in meters	400-420

Table 3.1: Significant attributes of HighD dataset

vehicle description and lane trajectories. This high resolution data would be very useful in extracting the experts behaviour easily. The Table 3.2 below shows the description of CSV file contents.

File Name	Data description
XX_recordingMeta.csv	Meta information about the recording setup
XX_tracksMeta.csv	Information of lanes and lane changes in the recorded video
XX_tracks.csv	File containing vehicle trajectories

Table 3.2: Table shows the description of each recorded video in term of CSV file. The XX in the filename represents the number of the recording ranging from 01 to 60

XX_recordingMeta

The files contain the data of the recording setup for a given recording XX, as XX range from 01 to 60. These files are use full in processing the data. Table 3.3 shows the detail of this file.

Feature columns of recordingMeta	Feature Description
id	Id given to each recording
frameRate	Frame rate of video camera = 25fps
locationid	Id given to each of the six recorded location
speedLimit	Speed limit of highway at that recording location
month	Month in which the recording was undertaken
weekDay	Weekday of the recording
startTime	Starting time of the recording
duration	Total time of the recording for a given locationid
totalDriverDistance	Total driven distance of all tracked vehicles
totalDrivenTime	Total driven time of all tracked vehicles
numVehicles	Total number of vehicles,including car and Truck recorded at that recording session
numCars	Total number of cars recorded
numTrucks	Total number of trucks recorded
upperLaneMarkings	Upper lane markings of the road segment
loweLaneMarkings	Lower lane markings of the road segment

Table 3.3: RecodingMeta file description

XX_tracksMeta

The current files for a given file number XX has a brief summary of trajectories for all the vehicle recorded on the road segment. tracksMeta files are most useful when the data needs to be filtered based on the lanes, class(car or trucks) and lane change. The Table 3.4 shows features of trackMeta data .

XX_tracks

Tracks data file contain vital information of the vehicle trajectories. This file represents the state information of all the vehicles recorded containing information of position, velocity, following vehicle id and many other state features in the traffic. The state information of vehicle is a time series starting form the initial frame to final frame of the vehicle. This time series data of each vehicle is listed in ascending order based on the vehicleId.

To understand the state features of this file, we need to learn the frame of reference to which HighD was recorded. Figure 3.3 shows the road segment coordinate system to which the vehicle states features are contextualized. The road segment is seen from the top view from the drones perspective and the vehicle move from left to right, or vice versa. The origin of the coordinate system is at the top left corner of the road segment. The x values increase as it moves to right and the y values increases when vehicles moves towards the bottom of the road.

XX_tracks contains the state features of the all recorded vehicles, the complete details the features are mentioned in Table 3.5. Figure 3.4 shows the illustration for these features when considering the ego vehicle, shown in yellow bounding box.

Feature columns of tracksMeta	Feature description
id	Id of the vehicle recorded, in ascending order based on time of first seen in the video camera
width	The width of the bounding box used to post-process the vehicle, represents the length of the vehicle
height	The length of the bounding box used to post-process the vehicle, represents the width of the vehicle
intialFrame	Video frame in which vehicle is first observed
finalFrame	Video frame in which vehicle is last captured
numFrames	Total number of frames in which vehicle is observed from start to final
class	Type of vehicle (car or truck)
drivingDirection	Driving direction of the vehicle from the point of drone. Either 1 for the left direction (upper lanes) or 2 for the right direction (lower lanes).
traveledDistance	total distance covered by the vehicle
minXvelocity	Minimum velocity of the vehicle in the given driving direction
maxXVelocity	Maximum velocity of the vehicle in the given driving direction
meanXVelocity	the mean velocity of the vehicle
minDHW	Minimal Distance Headway (DHW) to a preceding vehicle
minTHW	Minimal Time Headway (THW) to a preceding vehicle
minTTC	Minimal Time-to-Collision (TTC) to a preceding vehicle
numLaneChanges	Total number of lane changes by the vehicle

Table 3.4: tracksMeta file description

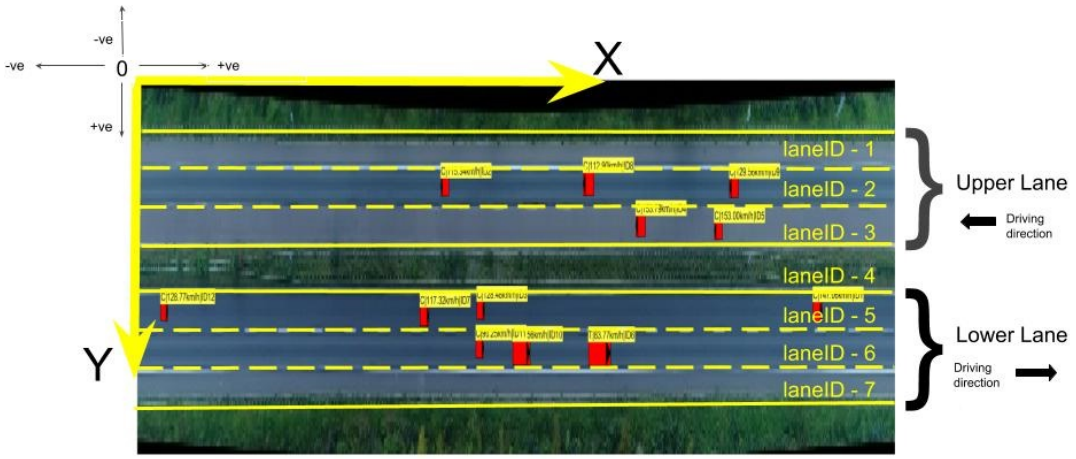


Figure 3.3: Road coordinate system. The vehicle in the upper lane move from right to left. The vehicles in this lane have x values decreasing as the vehicle moves from right to left and velocity of vehicle is negative in reference to the coordinates. It is converse in the lower lane.

3.3 Data processing

Current section deals with data extracting and processing techniques for the HighD data set that were essential in training DIRL models. The major steps in this process are to extract all the vehicle trajectories of ego and surrounding vehicle for lane change scenarios, in a two lane highway.

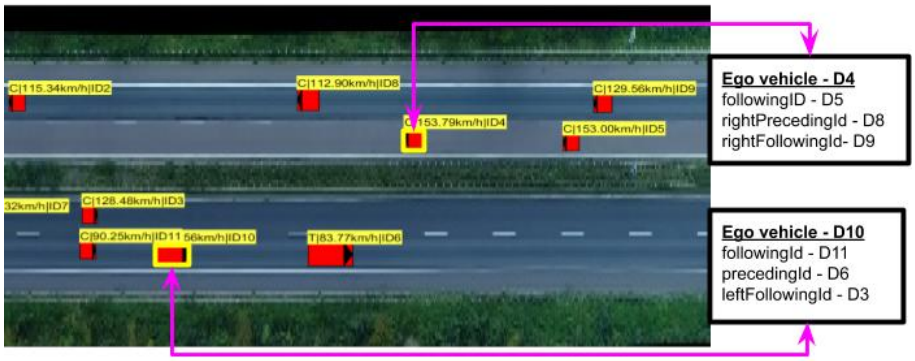


Figure 3.4: Illustration of _tracks feature considering ego vehicle 4 and 10 in upper and lower lane respectively

State features of Vehicle in tracks	Features description
frame	Current frame
id	Vehicle id
x	Position of vehicle in x direction in the current frame
y	Position of vehicle in y direction the current frame
width	The width of the bounding box used to post-process the vehicle, represents the length of the vehicle
height	The length of the bounding box used to post-process the vehicle, represents the width of the vehicle
xVelocity	Longitudinal velocity of the vehicle, positive in the positive x direction
yVelocity	Lateral velocity of the vehicle, positive in the positive direction
xAcceleration	Longitudinal acceleration of the vehicle
yAcceleration	Lateral acceleration of the vehicle
frontSightDistance	Distance from the current position of the vehicle till the end of the road segment in the driving direction
backSightDistance	Distance from the current position of the vehicle till the end of the road segment in the opposite driving direction
dhw	Distance headway measure
thw	Time headway
ttc	Time to collision
precedingXVelocity	velocity of the preceding vehicle in x direction
precedingId	vehicle id of the preceding vehicle in same lane
followingId	vehicle id of the vehicle following in the same lane
leftPrecedingId	The id of the preceding vehicle on the adjacent lane on the left in the direction of travel.
leftAlongsideId	The id of the adjacent vehicle on the adjacent lane on the left in the direction of travel.
leftFollowingId	The id of the following vehicle on the adjacent lane on the left in the direction of travel
rightPrecedingId	The id of the preceding vehicle on the adjacent lane on the right in the direction of travel.
rightAlsongsideId	The id of the adjacent vehicle on the adjacent lane on the left in the direction of travel.
right FollowingId	The id of the following vehicle on the adjacent lane on the left in the direction of travel
laneId	Id of the lane in which the vehicle is moving as in Figure 3.3

Table 3.5: Table describing the features of the tracks data

Lane Change data extraction

HighD data are recorded on highways having 2 and 3 lanes per direction. Considering our projects limitation we only extract lane change trajectories for two lane scenarios. Two lane here refers to highway's having two lane in the upper lane and two lanes in lower lane. Further in the report, lane is always mentioned in context to number of lanes in a single driving direction on the highway.

The data contains details of all the vehicle on the highway. In regards to project only the vehicles trajectories necessary were extracted. The following section explains the steps involved in extraction of lane change maneuvers, it also includes the data processing techniques. There are 60 recordings taken at different location, wherein each video recording is converted into 3 CSV file as mentioned in the earlier section. The projects make use of recordingMeta, tracksMeta to filter the necessary trajectories form the XX_tracks file.

Step 1 - Two lane data

Initial task is to learn file number XX, as XX ranges from 01 to 60 that posses the recording for two lane highway's. For this lowerlaneMarking feature column in the XX_recordingMeta file are taken into consideration to list all the XX_tracks file having two lanes. The Algorithm 2 lists a total of 13 files containing two lane highways. The algorithm iterates trough XX for on all recordingMeta files and counts the number of lowerlanemarkings, if there are 3 lane marking then the algorithm stores the file number else iterates further until end.

Algorithm 2 Two lane files

Input : HighD - XX_recordingMeta, XX_tracksMeta, XX_tracks
Output: Twolane_filename

```
for XX ← 01...60 do
    C ← count  lowerlaneMarkings in XX_recordingMeta
    if C = 3 then
        two_lane_files = XX
    else
        continue
    end
end
```

Step 2 - Extract the lane change data

After the files containing two lane highway are extracted, the following task was to extract lane change maneuvers. But initially the two lane files are divided into train and test set. Initially all the vehicle in the two lane highway don't change lane, we select only the vehicle that change lane. The vehicle that's change lane is referred as ego vehicle and the following vehicle, after the ego vehicle changes lane is called surrounding vehicle. A single following vehicle is called surrounding vehicle that is specific to scope of this project.

The complete time series data of the ego vehicle are extracted, by going through the numLanchages feature column of XX_tracksMeta. If the vehicleid has lane change values as 1, we choose such vehicle id and extract their complete highway data from XX_tracks file. To select surrounding vehicle trajectories for each of the ego vehicle,

LaneId column is searched from the selected ego vehicles the place the laneId changes, we look at the corresponding followingId column to know the surrounding vehicleid. Using this vehicleid, track data for the surrounding vehicle are extracted. The extracted surrounding vehicle tracks is appended column wise with corresponding ego vehicle. If there is no such surrounding vehicle for the ego vehicle or the data is devoid of surrounding vehicle, then the ego vehicle track data was eliminated. At this step the data has all ego vehicle and the corresponding surrounding vehicle tracks/trajectory data.

Algorithm 3 Lane change data

Input : HighD - XX_recordingMeta, XX_tracksMeta, XX_tracks , two_lane_filename
Output: Lane_change_trajectories

```
for XX ← 01...60 do
    if XX = two_lane_files then
        for i ≤ length(numLaneChange) do
            lanechange= XX_tracksMeta(numlanechange(i))
            if lanechange = 1 then
                lanechange_vehicle_id = XX_tracksMeta(id(i))
                ego_data = XX_tracks(id(lanechange_vehicle_id))
                surrounding_vehicle_id = ego_data(followingId)
                surround_data = XX_tracks(surrounding_vehicle_id)
                Lane_change_trajectories = concat(ego_data,surround_data)
            end
            Lane_change_trajectories = append(Lane_change_trajectories)
        end
    else
        continue
    end
    return Lane_change_trajectories
end
```

Step 3 - Filter data

The data extracted until now contains all the ego vehicle and corresponding surrounding vehicle trajectories. But these data may also include the trajectory data before and after the lane change occurs that is irrelevant to the model training. The data has to have only relevant information as training data the the lane change behaviour of the ego vehicle and the surrounding vehicle.

The average time for lane change is 4-4.5s [54], but we have trajectories for the complete road segment which is about 10s. The current project requires only relevant lane change trajectories, hence the need for the reduction in size of lane change trajectories. The data is reduced to about 7.5s trajectories, two sec is the history before lane change and 4.5s is lane change maneuver. To extract only these trajectories algorithm goes

through the laneId feature of the ego vehicle data. This is because in HighD the laneId changes when the vehicle's centre cross the lane marking that is when the bounding box centre crosses the the centre lane marking.

Initially laneId of the ego vehicle is noted at the start frame. As and when the laneID changes the value for given vehicle id, the corresponding frame is considered as the pivoting frame or lane change frame. The trajectories are extracted for the given vehicle by considering the feature before and after this lane change frame.

As mentioned earlier, the data is recorded at 25fps (0.04sec), therefore algorithm takes into account 2.5s about 50 frames before the lane change frame as the initial frame and 125 frames after the lane change as the final frame. The data between these initial and final frames are extracted. The ego vehicle's and the corresponding surrounding vehicle's data considered only between these initial and final frame and the rest are eliminated.

At the end of this step the expert data contains lane change scenarios of ego and surrounding vehicle with 7.5 sec trajectories concated by column. The Algorithm 4 show the data reduction process.

Algorithm 4 Data Filtering

Input : Lane_change_trajectories
Output: Filtered_data

ego_vehicle_id← from Lane_change_trajectories
for $i \leq length(ego\_vehicle\_id)$ **do**
    temporary_data ← Lane_change_trajectories for ego_vehicle_id(i)
    lane_change_frame ← frame in which vehicle changes laneid
    initial_frame, final_frame = lane_change_frame-124, lane_change_frame+40
    Filtered_data← temporary_data between initial_frame and final_frame
    Filtered_data ← append Filtered_data
end
return Filtered_data

Data Transformation

The expert data contains the behaviour all across the road segment which is non uniform in context of training the model. Feeding such data to DIRL model as an expert would prove disastrous, since the prediction of future states may depend on the place at which lane change occurs and there are are very few demonstration in highD data to learn all such behaviours on the complete road segment. The goal of IRL is to learn the lane change behaviour, irrelevant of the place of lane change of the ego vehicle. To make the data set robust for training and testing the following transformation techniques are employed.

1. **Centering lane change maneuvers**

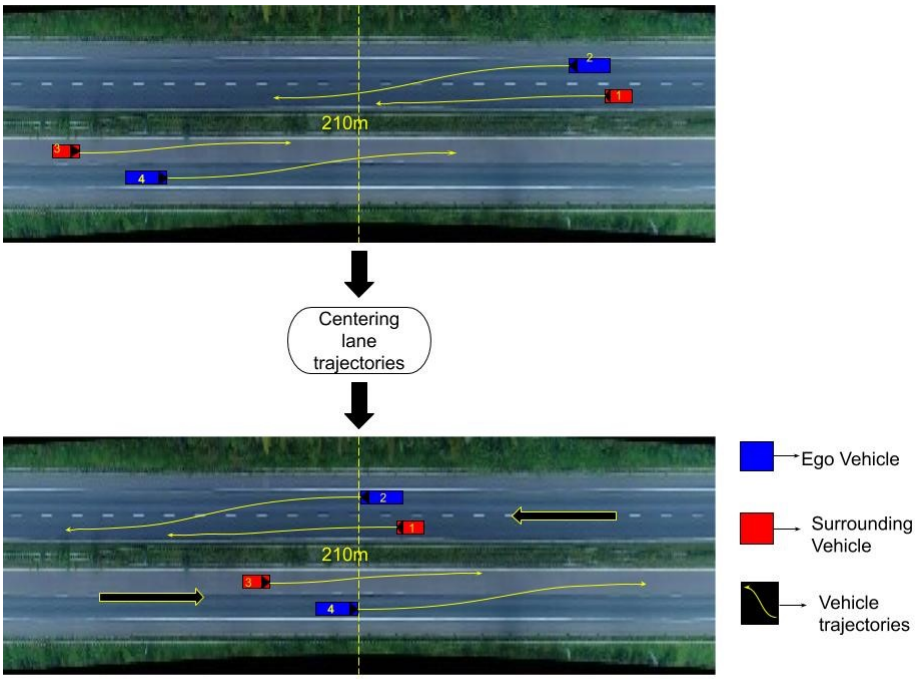


Figure 3.5: The lane change trajectories are shifted to the centre frame, with ego vehicle's 2 and 4, lane change maneuver starting at 210m

The lane change maneuver of the extracted data can start from any position on the road segment. That is the x position of ego vehicle in initial frame are varied. To make this uniform the ego vehicle is always considered start the lane change with x position at the mid way of the road segment, i.e at 210 m. This is executed by shifting the x values of the time series of all the ego vehicle data, so that all the ego vehicle x values has lane change trajectories starting from 210m. The surrounding vehicles x values are shifted with equivalent amount to its ego vehicle. The Figure 3.5 shows an example of centering function on the expert data.

2. **Mirroring along road centre**

The expert data collected consist of trajectories in both the upper lane and the lower lane. In the upper lane the vehicle move from left to right, with coordinate system defined in the Figure 3.3. x position of the vehicle reduces as it moves from left to right. In the initial frame the vehicle has the 410m and the last frame would have 0m. But the vehicles in the lower lane the has a completely converse x values for initial and final frames. If this data is fed to our model, it may predict the future behaviour based on the lane in which lane change occurs. To make the prediction uninfluenced by the lane, the trajectories of vehicle in upper and lower x values must be uniform, it should either increase with time or decrease with time, but never both.

Flip the upper lane vehicle trajectories, so that it can be assumed the vehicle x values increase with time. It can also be said that the vehicles trajectories are mirrored with respect to the mid segment of the road. An illustration the of this

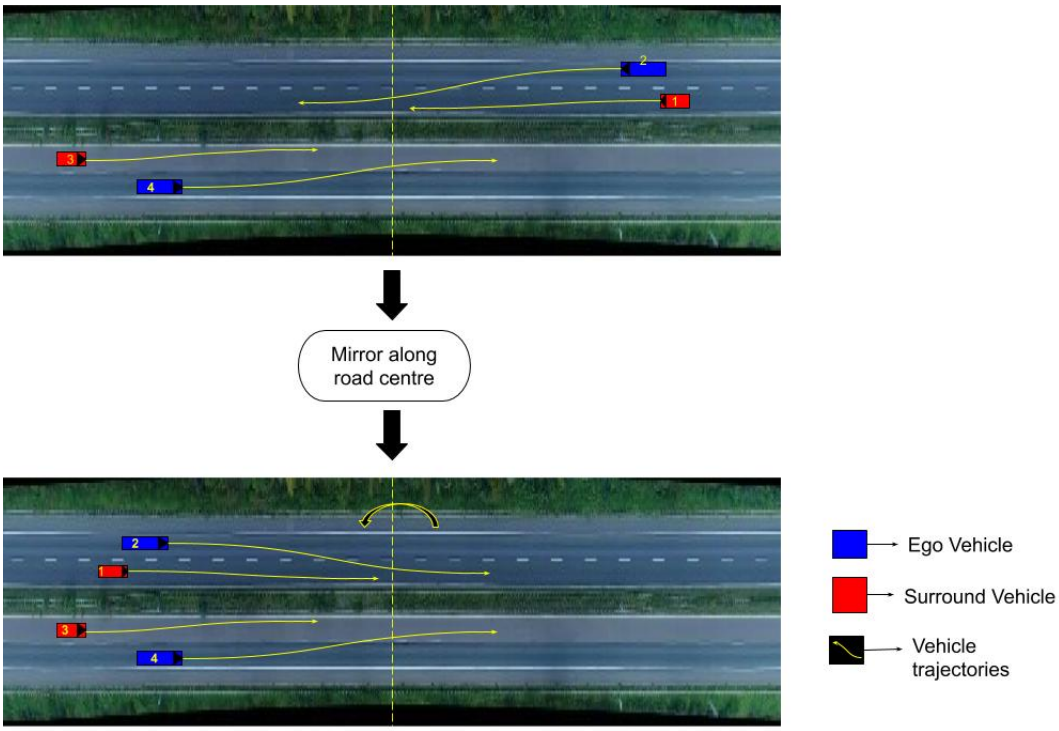


Figure 3.6: The figure illustrates the flip transformation on the data set. The ego vehicle 2 have their lane change maneuvers mirrored along the centre of the road. This results in vehicle trajectories in upper lane having the same moving direction as in the lower lane

transformation is shown in the Figure 3.6

3. **Mirroring along the lane markings**
   In the two lane highway the vehicle moves in one of the two lanes, that is vehicle moves in lane 2 or 3 in the upper lane, 5 or 6 in lower lane as in Figure 3.3. The vehicle changes lane either from 2 to 3 or 3 to 2 in the upper and 5 to 6 or 6 to 5 in the lower lane. Therefore during the lane change the values of y either increase or decrease over the time based on the lane it starts, which again it is unsuitable as training data. By previous convention the data whose y values decrease in time are transformed.

   Consider the ego vehicle starts from lane 3 and changes lane to 2. Over the course of lane change the y values decreases based on the coordinate system prescribed. While ego vehicle starting from 2 has the y value increased during the lane change. This non uniformity in the behaviour of lane change can be modified by mirroring the lane change maneuvers along the lane marking. The lane change that starts form lane 3, ending in lane 2 will be mirrored along lane marking between lane 2 and lane 3, this would transform data as though lane change are starting from the 2 and ending in 3. So would the corresponding surrounding vehicle data mirrored in accordance to its ego vehicle. This mirroring techinques

is applied to expert trajectories with ego vehicle starting from 6, which mirror along the lane marking between 5 and 6.

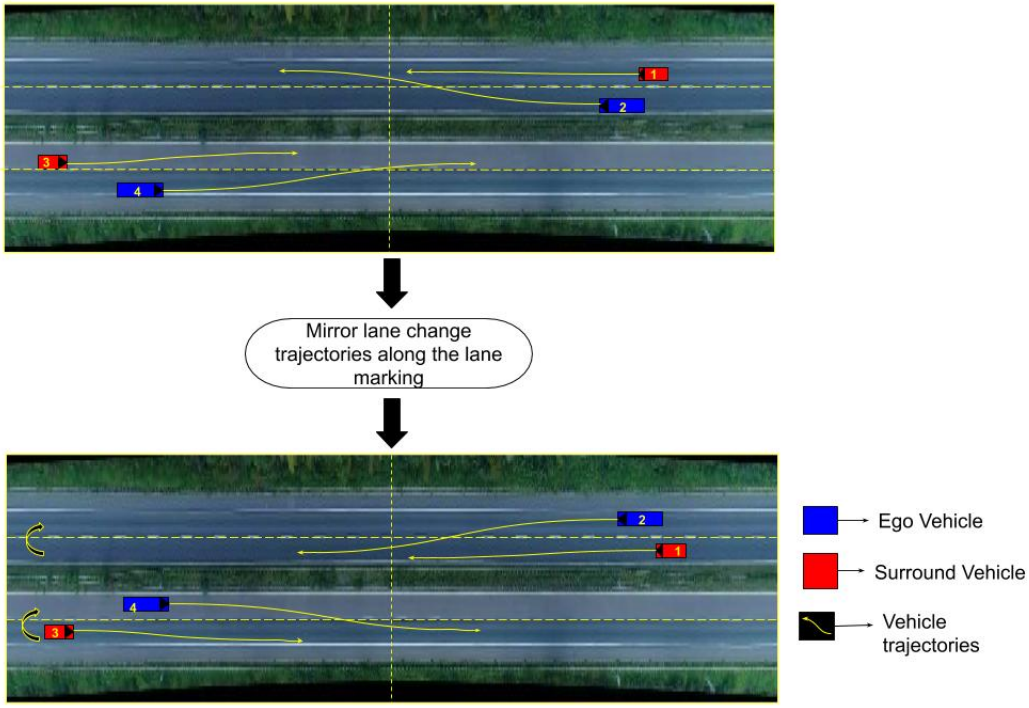


Figure 3.7: The expert trajectories, with ego vehicle in the lane 3 and 6, are mirrored along the lane marking in each of the lanes such that their lane maneuver starts from 2 and 5 respectively. It results in positive y values with time t

4. **Shifting lower lane**
   In this step the lane change occurring in the lower lanes are shifted, as though they are happening in the upper lane. Consider as though the lower is kept on top of the upper the lane by shifting its trajectories along the centre of the road as shown in the Figure 3.8.

These data transformation techniques are used to process the lane change trajectories to form the expert data for the DIRL model. The vehicle trajectories in each lane undergoes at least one of these transformation or all of these transformation based on lane in which the lane change starts.The Figure 3.9 describes all the transformation based on the ego vehicle starting their lane change for each lanes. The transformation performed simultaneously to the surrounding vehicle with respect to its ego vehicle.

The expert data extracted has to many features. The features that are irrelevant are eliminated, only data from the features x(position), y (position), xVelocity and yVelocity of the both ego vehicle and surrounding vehicle are kept. In addition to this four more features are added to the data point, which are the difference in the x, y xVelocity and yVelocity between the ego vehicle and surrounding vehicle. After this the data set of the expert contains the features as mention in the Table 3.6.

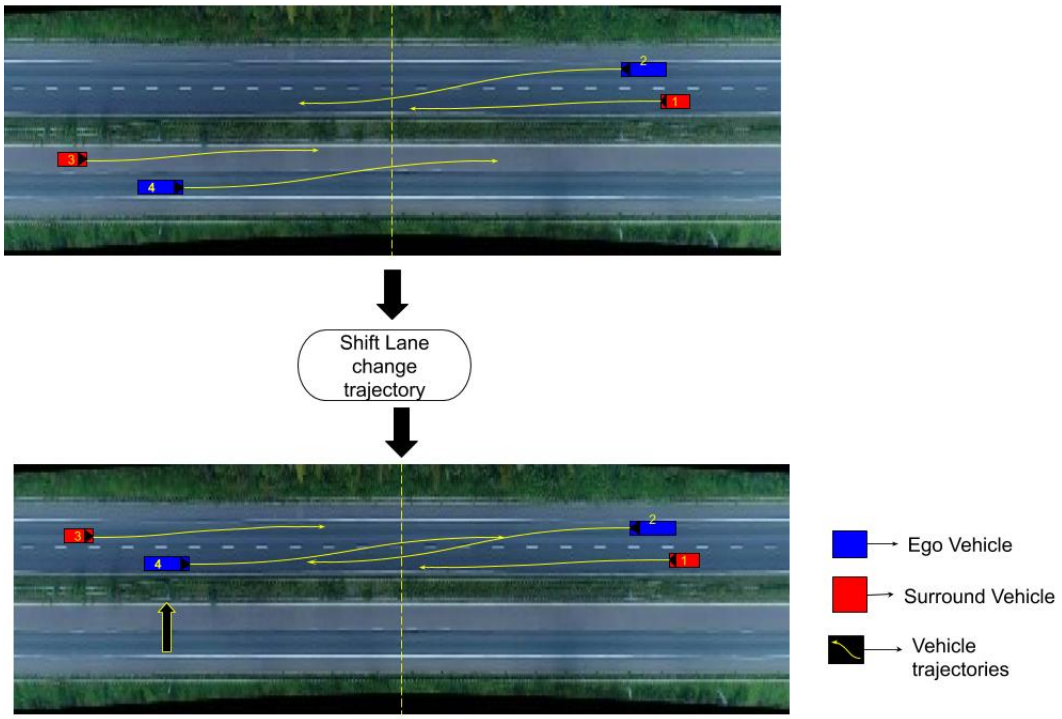


Figure 3.8: Expert trajectories in the lower lane are shifted to upper lane maintaining the same direction as in the lower lane

Expert features	Feature Description
x_E	Position of ego vehicle in x direction
y_E	Position of ego vehicle in y direction
$xVelocity_E$	Longitudinal velocity of ego vehicle
$yVelocity_E$	Lateral velocity of ego vehicle
x_s	Position of surrounding vehicle in x direction
y_s	Position of surrounding vehicle in y direction
$xVelocity_S$	Longitudinal velocity of surrounding vehicle
$yVelocity_E$	Lateral velocity of ego vehicle
x_Δ	Distance between ego and surrounding vehicle in x axis
y_Δ	Distance between ego and surrounding vehicle in y axis
$xVelocity_\Delta$	Difference in longitudinal velocities of the ego and surrounding vehicle
$yVelocity_\Delta$	Difference in lateral velocities of the ego and surrounding vehicle

Table 3.6: Expert Data features for the DIRL model

Expert data has very high precision, this can be of great advantage in augmenting the data. Since the two lane road segment has fewer lane change scenarios augmenting data helps the model for accurate prediction. The data has 0.04s sample time, for a 7.5 s trajectory the length of vehicle trajectory for each features is about 175 time steps. Length of the sequence is large , therefore at each expert time series data is split into two sequences with 0.08s sample time. This serves two purposes, it first helps in augmenting the data set and the second it reduce the sequence length for prediction.

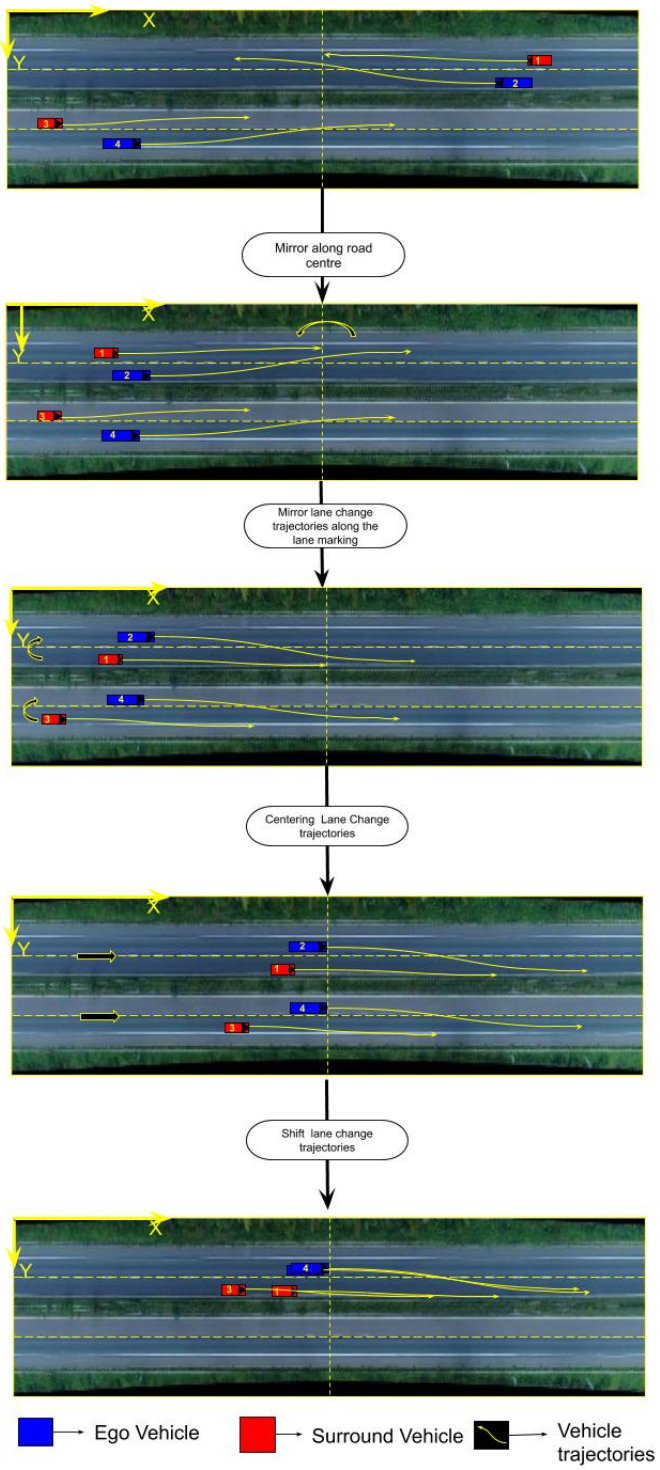


Figure 3.9: The figure illustrates the the complete data transformation's undertaken by the expert data

3.4 Deep Inverse Reinforcement Learning Model

The current section is a description of the model building procedure for Deep Inverse Reinforcement Learning for the traffic prediction. The data collected has the vehicle trajectories of both the ego and the surrounding vehicle and are considered as the experts behaviours. The project learns the intention of the surrounding vehicle by predicting the trajectories of the surrounding the vehicle before and after the lane change. The data is times series value of the vehicles position and velocity. The DIRL models in this project is influenced from the research papers in the Natural Language Programming and Imitation learning [55, 56, 57, 46].

The expert trajectories τ_E collected has the the state space values of the vehicle and lacks the state transition dynamics to perform the usual MaxEntr IRL to learn the driver behaviour. GAIL is most suited for our task, it learns the policy of the expert, which in turn is a model free MaxEnt IRL [24]. GAIL can intuitively understood as the model that learns the task given the trajectories, wherein the generator G learns the policy π using the RL algorithm and the discriminator network helps us obtain the rewards at time t [24].

The project makes some advancements in the GAIL from the original GAIL architecture. The generator in the original proposed GAIL was simple MLP used to learn behaviour for discrete task but wasn't efficient for continuous control task [46]. But trajectories in our task have stochastic policy which isn't effectively capture the by normal MLP of GAIL. So the project builds DIRL model based upon similar architecture as Sequential Generative Adversarial Network(SeqGAN). SeqGAN was built to generate a text sequence using GAIL principles, it can capture stochastic policy easily.

In SeqGAN, the generator of the GAN is a Sequence to Sequence(Seq2Seq) network [55]. Seq2Seq is used in Natural Language Processing (NLP) for text translation wherein the input sequence needs to be translated into different language [12], but these are also used in time series prediction in various field of the weather prediction, vehicle trajectories to predicts time series data efficiently [58, 59]. It consists of two LSTM layers, where one reads the input sequence and the others extracts the translated output. The input sequences are converted to single vectorized representation by the first LSTM encoder, and fed to the second LSTM decoder, that extracts the output which is conditioned on the input vector. The discriminator in the DIRL model has a similar architecture to that of generator. The basic architecture of the DIRL model is shown in the Figure 3.10.

3.4.1 DIRL Model

The task of the project is to predict the future trajectories when provided with previous history. The trajectories are nothing but state features of the expert behaviour. s_t is

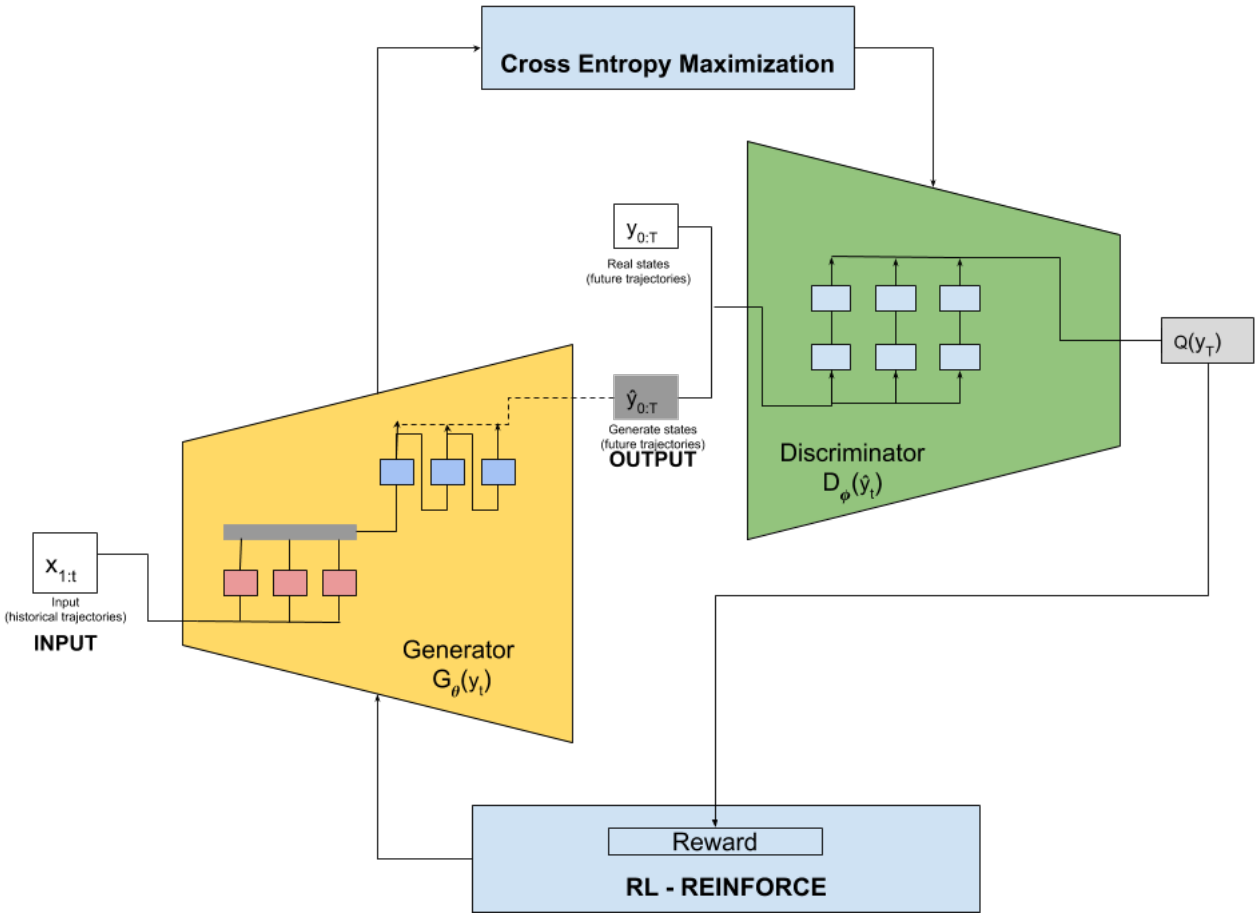


Figure 3.10: DIRL network architecture

the state features at time t for single feature representing one of x, y , xVelocity and yVelocity. Complete state of the vehicle behaviour for the lane change maneuver at time t is $f(s_t)$, which is nothing but the complete feature set in Table 3.6. The model is explained for the single state s_t feature trajectories, but the same formulation holds true to a complete feature set.

Let τ represent the set of all the expert trajectories extracted from the data.

$$\tau_E = \{\tau_1, \tau_2, \ldots \tau_n\} \tag{3.1}$$

wherein each expert trajectories is a times series of state representation of lane change maneuver

$$\tau_i = \{x_1^{(i)}, x_2^{(i)}, \ldots x_t^{(i)}, y_1^{(i)}, \ldots y_T^{(i)}\} \tag{3.2}$$

$$\tau_i = \{\tau_h^{(i)}, \tau_l^{(i)}\} \tag{3.3}$$

where $\tau_h = \{x_1, x_2, \ldots x_t\}$ represents the historical trajectories and the $\tau_l = \{y_1, \ldots y_t\}$ are the lane change trajectories of the expert. $x_{1:t}$ are in the input state trajectories that are fed to the model, and $y_{1:T}$ are future trajectory. The project makes use of x_t, y_t to distinguish between historical states and predicted states but these are nothing but s_t

of the expert trajectory data.

The historical trajectories are fed to the generator, G that encodes historical trajectories and generates a future trajectories. The seq2seq network of the generator, models the conditional density of the future trajectories provided the input sequence of historical trajectories.

$$P(\tau_l|\tau_h) = P(y_1 \dots y_t|x_{1:t}) \tag{3.4}$$

$$P(\tau_l|\tau_h) = \prod_{t=1}^{T} P(y_t|c, y_1, \dots y_{t-1}) \tag{3.5}$$

where c = $\{x_1 \dots x_t\}$, is cell states fed to the decoder. The decoder of seq2seq in generates an output sequence auto-regressively, as in at each time t, the decoder takes the input from previous decoded output and generates the output at the current time. The output of this seq2seq model is a probability given by the softmax function.

$$P(y_t|y_{t-1}, y_{1:t}) = \frac{\exp(w^\intercal f(y_t, y_{t-1}, x_{1:t})}{\sum exp(w^\intercal f(y_{t-1}, x_{1:t})} \tag{3.6}$$

The parametric output of the generator for the complete sequence is $G_\theta(s_{1:t})$. At each time step t the output of the generator is given by

$$G_\theta(y_t) = P(y_t|c, y_1, \dots y_{t-1}) \tag{3.7}$$

The discriminator, D_ϕ is fed with with the samples of generator and real data. It determines whether the state trajectories of the generator are real or fake. The parametric output of discriminator is given as

$$D_\phi(\hat{y}_t) = P(\hat{y}_t = y_t^{real}|y_{1:t-1}) \tag{3.8}$$

The output of the discriminator is used to train the generator using RL and the discriminator was updated using SGD.

3.4.2 Model training

The output of generator at time t is probability of the state y_t. It can be said that action a_t taken by the generator at time t is s_t. The state of the generator is the states generated until the time t.

$$Policy = \pi(y_t) = P(y_t|\tilde{y}_{1:t}) \tag{3.9}$$

$$State = \tilde{y}_{1:t} = \{y_1 \dots y_{t-1}\} \tag{3.10}$$

The reward is specified by the the discriminator, and we define the state value function as shown in the equation

$$r_t = \log D_\phi(\hat{y}_t) \tag{3.11}$$
$$Q^\pi(y_t) = \mathbb{E}_{\pi \sim G_\theta(y_t)}\left[D_\phi(\hat{y}_t)\right] \tag{3.12}$$

Training our model is similar to training the GAIL 1. The output of the discriminator are used to update the generator parameter θ by policy gradient methods. The discriminator parameter ϕ are updated using generated policy.

The policy gradient method that is considered for our project is REINFORCE. REINFORCE is a policy gradient method based on monte-carlo policy estimation.The policy gradient as is given by from the theory section

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{G_\theta}\left[R_t \nabla_\theta \log \pi_\theta(s_t)\right] = \mathbb{E}_{G_\theta}\left[Q^\pi(s,a)\nabla_\theta \log \pi_\theta(s_t)\right] \tag{3.13}$$

In the present model the policy gradient can be written as

$$\nabla_{G_\theta} \mathcal{J}(\theta) = \mathbb{E}_{s_t \sim G_\theta}\left[Q^\pi(s_t)\nabla_\theta \log G_\theta(s_t)\right] \tag{3.14}$$

Monte Carlo estimate with roll out policy is applied to unknown future trajectories from the current step t until the final step T. It is evaluated using the same generator and is referred as G_{MC} [55]. The sampled next state action value for $s_{t+1}$is given by

$$Q^\pi(s_t) = \begin{cases} \frac{1}{N}\sum D_\phi(\hat{s}_t^n), \forall s_t^n \in G_{MC}, & t < T \\ D_\phi(\tilde{s}_{1:T}), & t = T \end{cases} \tag{3.15}$$

Once the sequence is generated its parameter G_θ of generator are updated using the policy gradients in Equation 3.14

$$\theta \leftarrow \theta + \alpha_G \nabla_\theta \mathcal{J}(\theta) \tag{3.16}$$

where α_G is the learning rate.

Then the discriminator generates the state action value for the policy gradients, its parameter ϕ are updated by normal cross entropy as defined in Equation 2.14

$$\nabla_{D_\phi} \mathcal{J}(\phi) = \mathbb{E}_{s \sim data}\left[\nabla_\phi \log D(s_{1:T})\right] + \mathbb{E}_{s \sim G_\theta}\left[\nabla_\phi \log(1 - D(\hat{s}_{1:T}))\right] \tag{3.17}$$

Discriminator is updates by Equation 3.18, α_D is the learning rate of the discriminator

$$\phi \leftarrow \phi + \alpha_D \nabla_\phi \mathcal{J}(\phi) \tag{3.18}$$

The complete training sequence of DIRL model undertaken is illustrated in the Algorithm 5

Algorithm 5 Deep Inverse Reinforcement Learning for lane change behaviour

Input : Expert data - $\tau_i = \{\tau_1, \tau_2, \ldots \tau_n\}$
Initialize the weight parameters θ, ϕ

repeat
  $\tau_i \leftarrow$ sample trajectories
  for $i = 1$ *to G steps* **do**
    $y_t \leftarrow$ Generate state trajectory from previous $G_\theta(y_t)$
    $Q^\pi(y_t) \leftarrow$ State action value from the discriminator $D_{\phi i+1}(\hat{y}_t)$
    $\theta_{i+1} \leftarrow$ Update the generator parameter using the REINFORCE
  end
  for $i = 1$ *to D steps* **do**
    $y_t \leftarrow$ Generate state trajectory from previous $G_\theta(y_t)$
    $Q^\pi(y_t) \leftarrow$ State action value from the discriminator $D_\phi(\hat{y}_t)$
  end
  $\phi_{i+1} \leftarrow$ Update the discriminator using the cross entropy
until *Number of steps*;

3.5 Software library

The project makes use of the python language to perform the necessary task. The advantage of using python language is its easy to use syntax, many inbuilt functions, several open source library to easily perform analysis and build our model. These library are well developed in performing various specialized task like, data manipulation, mathematical operation and help in building neural networks easily.

Pandas

The project makes use of pandas library to extract and build the expert data frame required for of DIRL model. In addition it is extremely useful in data transformation and imputing the Nan values if there are in the expert data frame.

Tensorflow

TensorFlow is built on the principal of computational graphs, that performs the necessary mathematical operations.It developed under Apache 2.0 licence by Google . It is most useful in building neural networks starting from simple to complex and large scale models. The project makes use of this library to build the generator and discriminator neural network for the DIRL model.

4

Results and Discussion

The current chapter deals with experimental results obtained form the DIRL performed on the expert data. Chapter includes the brief discussion on the performance analysis of our model.

The DIRL model was trained for 500 epoch with an equal learning rate of 0.0001 for both the generator and discriminator networks. At each epoch, a random sample of historical trajectory is fed as input and was trained according to Algorithm 5. The DIRL model was trained on each feature separately to reduce the computational complexity. As the model had to learn 12 features for predicting 4.5s of future trajectory. The statistics in this section are based on 100 samples of output from the DIRL model during the testing period. The result shown in this section are divided into longitudinal and lateral features, this helps in recognizing the influences of each feature on the predicted trajectory. The predicted trajectory is a time series, and the project measures the general statistical parameter instead of the similarity measures between time series. It helps in analyzing the basic structure of the data and the predicted trajectory.

4.1 Longitudinal features

The Table 4.1 illustrates the mean and standard deviation of predicted trajectory. The results correspond to both the longitudinal features, position and velocity of the ego, surround and their difference.

Longitudinal features		Mean	Standard Deviation
Position (m)	ego	347.69	42.046
	surround	308.72	34.55
	difference	39.88	7.49
Velocity (m/s)	ego	33.11	0.021
	surround	27.20	0.085
	difference	5.91	0.075

Table 4.1: Mean and standard deviation of the complete 4.5s predicted trajectories for the longitudinal features

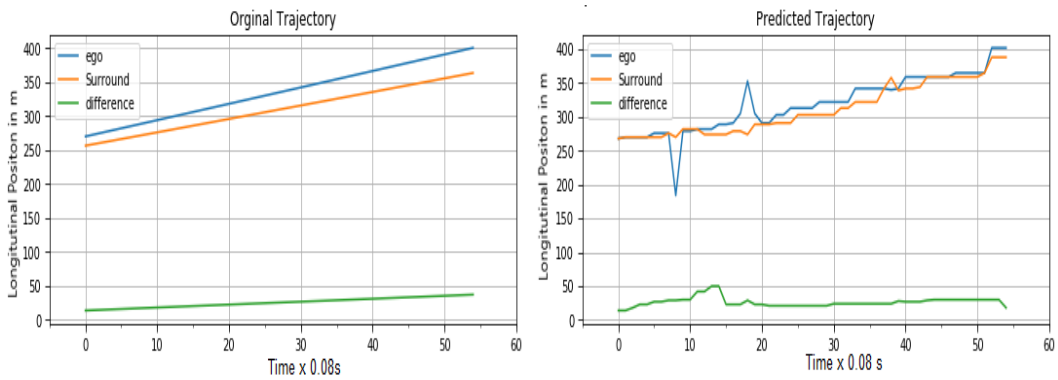
The longitudinal position of the vehicle is always increasing and has high vari-

ance, as it represents vehicle position while moving along the length of the highway. The mean longitudinal position of the ego vehicle is about 350m and varies from about 230m until the end of 450m. This due to the data transformation wherein the ego vehicle's historical trajectories start from 210m and the corresponding lane change trajectory of the predicted trajectory starts around 250m. The high change in positional features is reflected in the standard deviation. But on the contrary the longitudinal velocities do not vary along the time, it is almost uniform for 4.5s. This may be due to the highway driving scenario, as the vehicle maintains a relatively constant speed on the highway.

Predicted trajectory

The sequential data predicted by the DIRL model is illustrated in the Figure 4.1. The output of the predicted trajectory is compared with actual trajectory of the vehicle. It can be deduced that the longitudinal position follows a linear trend and the velocity trajectory is remains almost constant. The results. Longitudinal velocity has the similar prediction as the the original trajectory unlike the position. It seems that the training epoch was sufficient to capture the longitudinal velocity but not the longitudinal position.Since the time series data of the longitudinal velocity has very small variance along the 4.5s trajectory compared to the longitudinal position.

Longitudinal Position



Root Mean Squared Error

The Table 4.2 shows Root Mean Squared Error (RMSE) evaluated for predicted trajectory. The table is divided into three equal time intervals for the 4.5s trajectory output, predicted by the DIRL model. The division of output time series into time intervals helps to analyze the robustness of the model to predict long-time series with only historical trajectories. DIRL is very good at predicting the initial times series until 1.5s as it has less deviation. But has relatively weaker performance in the second interval, as the deviation from the original trajectory as the lane change behaviour is maximum at this place. But in the third interval, it significantly improves its performance with very small errors. This behaviour can be also observed form the Figure 4.1

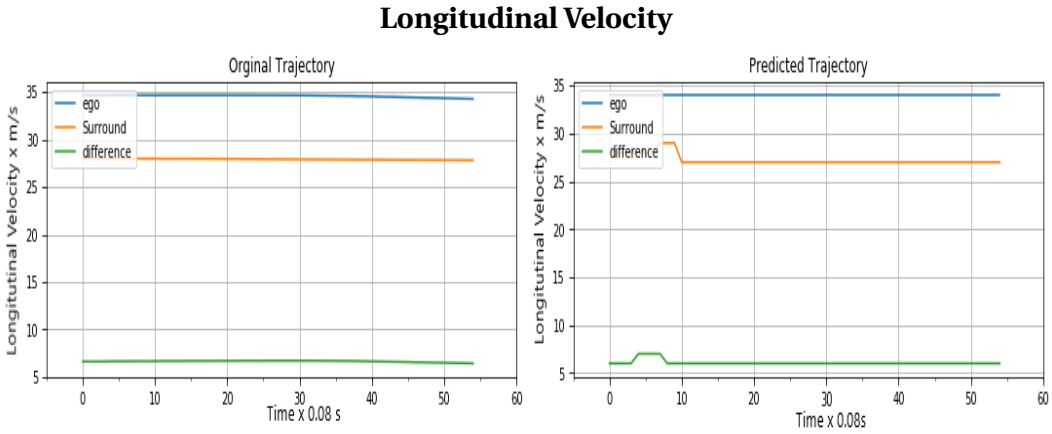


Figure 4.1: The plots of longitudinal position and velocity of 4.5s predicted trajectory

Features		RMSE		
		0s -1.5s	1.6s - 3s	3.1s -4.5s
Position (m)	ego	3.455	3.85	3.74
	surround	3.385	4.162	4.127
	difference	3.021	3.91	3.324
Velocity (m/s)	ego	1.0105	1.036	0.969
	surround	1.002	1.054	0.957
	difference	0.978	0.998	0.972

Table 4.2: Evaluation of the RMSE of the predicted trajectories for the longitudinal features. RMSE is evaluated for 3 time intervals for the 4.5 s trajectory

4.2 Lateral features

In this section, the output of the model is evaluated for the lateral features. The lateral features of both the position and velocity were scaled before being fed to the DIRL model. Scaling was necessary as the changes in the lateral direction are too small to be captured by the networks. Table 4.5 shows the average lateral displacement of the vehicle during the lane change. The displacement in the lateral position was comparatively small compared to a longitudinal position. The surround vehicle has the highest displacement signifying its lane change behaviour. But the more influencing feature is the lateral velocity that has a significant change in values for the mean and standard deviation.

Predicted trajectory

The lateral position and velocity of the predicted trajectory is compared and plotted as shown in the Figure 4.2. The lateral position and the lateral velocity follows similar trend unlike the longitudinal features.The lateral velocity has a non linear trend especially the surround vehicles trajectory.

Lateral features		Mean	Standard Deviation
Position (m)	ego	12.08	0.965
	surround	14.54	0.01
	difference	2.45	0.953
Velocity (m/s)	ego	0.678	0.18
	surround	0.041	0.009
	difference	0.638	0.019

Table 4.3: Mean and variance of the sampled predicted trajectories for lateral features

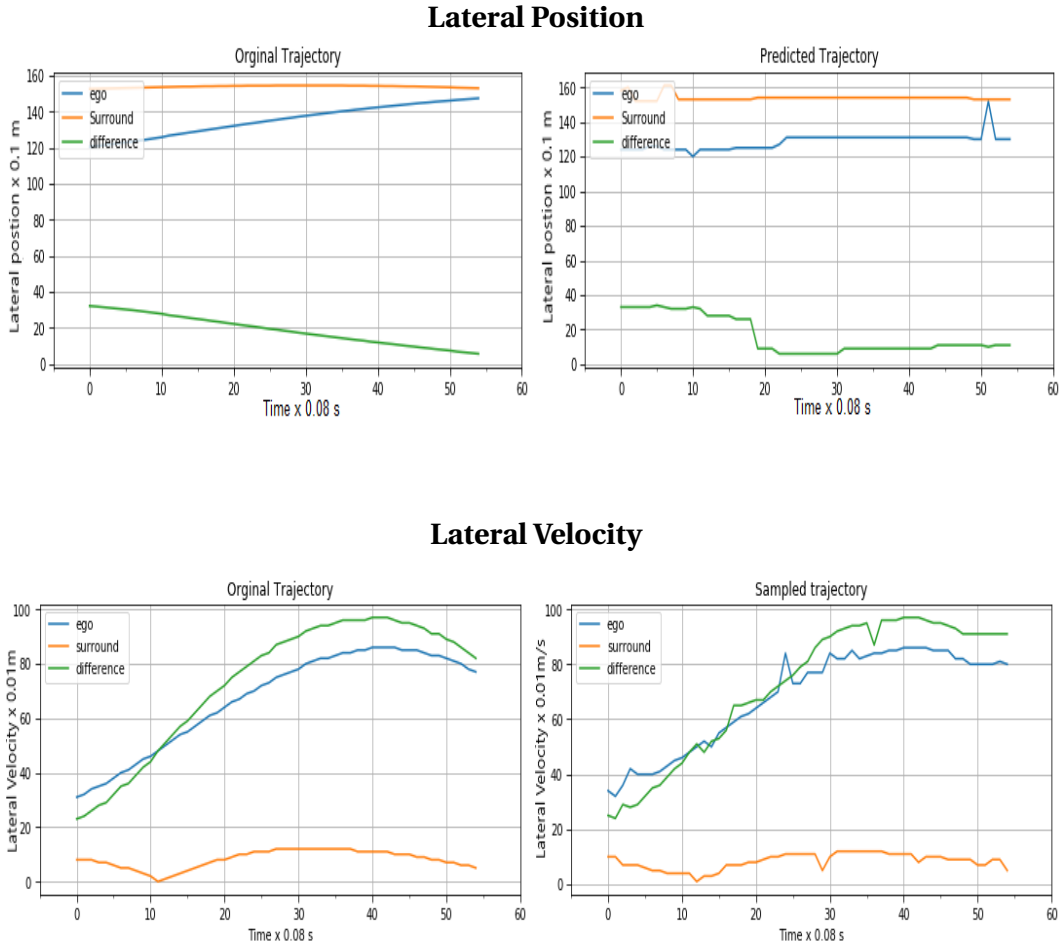


Figure 4.2: Plots of lateral position and velocity of 4.5s predicted trajectory

Root Mean Squared Error

The robustness of the DIRL model for lateral features is evaluated using the RMSE for the predicted trajectories. The RMSE is evaluated at three time intervals as shown in the Table 4.4. The model performs significantly better at predicting the ego vehicle trajectories position and velocity for all three-time intervals. Moreover the model has the highest accuracy in predicting the ego vehicle's position in the third interval and difference position in the second interval.The model performance in predicting the driving behaviour reduces after 1.6s .

Lateral Features		RMSE		
		0s -1.5s	1.6s - 3s	3.1s -4.5s
Position (m)	ego	2.181	2.894	0.816
	surround	2.058	2.98	1.637
	difference	1.537	0.57	1.093
Velocity (m/s)	ego	0.474	0.749	0.589
	surround	0.141	0.284	0.113
	difference	0.42	0.671	0.751

Table 4.4: Evaluation of the RMSE of the predicted trajectories for the lateral features. RMSE is evaluated for 3 time intervals for the 4.5 s trajectory

4.3 Discussion

In this section, the overall performance of the DIRL is evaluated and analyzed. Table 4.5 show the overall RMSE values for all the features, predicted trajectory. RMSE for all the features is significantly lower except for the lateral velocity. This can also be viewed in Figure 4.2. The predicted trajectory of lateral velocity has a noisy behavior compared to the rest of the predicted trajectory.

features		RMSE
Longitudinal Position (m)	ego	3.687
	surround	3.98
	difference	3.41
	Average	3.61
Longitudinal Velocity (m/s)	ego	1.007
	surround	1.014
	difference	0.98
	Average	1.001
Lateral Position (m)	ego	1.963
	surround	2.258
	difference	1.082
	Average	1.757
Lateral Velocity (m/s)	ego	0.604
	surround	0.179
	difference	1.87
	Average	0.875

Table 4.5: Mean and variance of the sampled predicted trajectories for lateral features

The RMSE of the position for longitudinal and lateral is 2.68m and similarly the RMSE for the velocity is 1.005m/s. This results show that the DIRL model predicts the the positional feature more accurately than velocity feature . Since the model was the scaled for the velocity features , the error also are scaled, this may be on of the reason that the velocity feature prediction has weaker performance.

DIRL was set up to choose random values from the given vocabulary size based on the probability output from the generator's soft-max to predict the next time trajectory. It helped the DIRL model to choose diverse values to capture varied driving behaviour and get the reward for all such values. The probability distribution from the softmax layer of the generator was spread out for data with high variance such as longitudinal position and lateral velocity. The discriminator provided this prediction with relatively good rewards even though the prediction was not accurate for the current input(historical trajectory). Training generator with this reward weakens the generator performance to provide a noisy time series prediction. The spikes and noise illustrated in the predicted trajectory for the longitudinal position and lateral velocity feature Figure 4.1,4.2 are due to the above reasons. The DIRL model compensates for accuracy to predict varied trajectories and driving behaviour.

The feature set of the expert includes the difference column, but analysis concentrated on the ego and surround features. The difference column was added to the feature set to test hypothetical case when provided only the relative position and speed of the ego and surround vehicle could the DIRL model learn the driving behavior. The DIRL model was able to predict this feature quite accurately.

The list of features that are trained is large. It can be of advantage if the features set could be reduced. In such a case reduced data set must contain the most important features. Based on the results the most varying features are the longitudinal position and lateral velocity. These two features are of significant importance and must be retained in the reduced data set for prediction.

Initially, the model was set up to run the DIRL model until the generator and the discriminator converges. But the time it took was very high and ran into unprecedented executions glitches. The convergence of GAN wasn't achieved one of the reasons is that the varied time series trajectories. The generator of DIRL couldn't capture all the time series behaviour to fool the discriminator. This lead to reducing the training period of DIRL for a fixed number epoch mention earlier in this section. Even after reducing the training period DIRL model took about 12 - 15 hours to train each feature. Using more advanced computers with multi-core capabilities the DIRL model can easily predict the vehicle trajectories.

5

Conclusion and Future Work

In this section the project result are contextualized with the research questions and a possible future course of direction for the project development for better performance are suggested.

5.1 Conclusion

In the current project, a model based on inverse reinforcement learning was developed to predict human driving behavior during the lane change scenario. The DIRL model predicts the vehicle motion trajectory for 4.5 sec given the historical vehicle motion. The vehicle trajectory predicted, in turn, reflects the intentions of the driver during and after lane change. The DIRL model predicts the behaviour of both the ego vehicle and the surrounding vehicle during lane change scenarios. Though the aim was to predict the surrounding vehicle motion the model tries to predict a probabilistic trajectory of the lane change interactions all-together. This probabilistic time series prediction is very useful when provided to ADS.

The project aims to find answers to the research question stated in Section 1. From the results of the DIRL model, the answer to the research question is explained.

- **Research question 1:**How efficiently can Deep Inverse Reinforcement Learning algorithms can predict vehicles behaviour?

  The DIRL model predicts the lane change behaviour of the vehicle quite accurately with RMSE for the position is 2.68m and the RMSE for the velocity is 1.001m/s. The advantage of the DIRL model compared to the deep learning model is that it can predict a complete lane change trajectory of the vehicle behavior. DIRL model was able to learn such a complex driver behaviour with a comparatively small data set.

  The DIRL model was able to predict the longitudinal position, velocity and lateral position more accurately than the lateral velocity. The model was trained for relatively small-time, so to produce more accurate results DIRL model must be trained for a longer period or until the GAN converges with a more advanced

computer.

- **Research question 2**: How are the features of the surround vehicle mapped to get the optimal reward in inverse reinforcement learning?

  The reward functions are used to train the GAN using inverse reinforcement learning as described in Algorithim 5. The model maps the reward function to generate the predicted trajectory. So in the DIRL model, the rewards are not explicitly generated but are learned by the model, which influences the performance of the predictions.

  Optimal or the best reward generates the optimal policy according to Equation 2.33 [34]. In the current project, a policy is nothing but the predicted trajectory. The predicted trajectory of longitudinal and lateral features with the least error has the optimal reward mapped to the features set. The features, both longitudinal and lateral of ego vehicle has the optimal rewards. Since the predicted trajectory is very similar to the expert trajectory. The DIRL model could not learn the optimal rewards of the surround vehicle features but with more training, it would be possible to learn the best reward for the surrounding vehicle.

- **Research question 3**: Can the model predict future driving behaviour even after the lane change based on the output of the inverse reinforcement learning algorithm?

  The DIRL model is based on the Sequence to Sequence architecture that can predict time series of variable length. It is possible to predict the trajectory for more than 4.5s using the current prediction model. It is shown in the Tables 4.14.5 that the RMSE for the final interval of 3-4.5s for all the features is low. It seems that DIRL model may able to predict future trajectories beyond the 4.5 sec, which is after the lane change. But the accuracy of the predicted value must be evaluated and if the surround makes another lane change after the 4.5s it may be difficult for the model to predict this behaviour

5.2 Future Work

The project concentrated on discovering a new approach in predicting the driver intentions. Therefore there is a huge scope for improving the model. Future work on this model can be made for more accurate result, in the following direction that are listed below.

- Due to the scope and time constrain the DIRL model was trained on individual features. It would be ideal to train the DIRL model on multiple features.

- The project concentrated on two agent lane changing scenario. The model can be extended to predict driver behaviour of multiple agents and multiple scenarios.

- Training the DIRL model can also include the training the hyper-parameter of the neural network
- The number of LSTM layer can be increased in the GAN network, especially increasing the LSTM in the Generator, improves the efficiency of the model.
- Experimenting with various architectures in the generator network and discriminator network, on of the possible is to make use of wavenet model, as it seems promising for sequential data.
- Advanced policy gradients such a Actor Critic, TRPO, DQN and PPO yields better result and may reduce the variance of the time series prediction.
- A comparative study of the DIRL model with already known models may help to gauge the superiority or the advantages of DIRL model

Bibliography

[1] Zhiyong Cui. Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. pages 1–12.

[2] João Pedro and Cardoso Pereira. Unsupervised Anomaly Detection in Time Series Data using Deep Learning. (November), 2018.

[3] Richard Sutton and Andrew Barto. Reinforcment Learning. *MIT Press*, 2018.

[4] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-Novem(November):2118–2125, 2018.

[5] Deloitte. 2019 consumer products outlook: Have consumer products companies reached a technology inflection point? 2019.

[6] Peg Young, Ken Notis, and Theresa Firestine. Bureau of Transportation Statistics Transportation Services Index and the Economy—Revisited. (July 1980), 2014.

[7] European Commission. Safe Mobility: A Europe that protects. (March 2018), 2018.

[8] S. Singh. Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey. *NHTSA's National Center for Statistics and Analysis This*, (August):1–2, 2015.

[9] C. J Kahane and J. N Dang. The Long-Term Effect of ABS in Passenger Cars and LTVs. *National Technical Information Service*, (Report No. DOT HS 811 182):89, 2009.

[10] I Goodfellow Courville and Y Bengio. *Deep learning*. MIT Press, 2016.

[11] Alex Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, (3):6645–6649, 2013.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. pages 1–9, 2014.

[13] D. E. Rumelhart and R. J. Williams Hinton, G. E. *Learning internal representations by error propagation.* MIT Press, 1986.

[14] Paul J. Werbos. Backpropagation Through Time:What it Does and How to do it. *Proceedings of the IEEE*, 78(10):1553–1560, 1990.

[15] G V Puskorius. Truncated Backpropagation Through Time and Kalman Filter ' Training for Neurocontrol. pages 2488–2493, 1994.

[16] Bengio Y, Simard P. 1994 Learning long-term dependencies with gradient descent is difficult.

[17] Sepp Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 2003.

[18] Hochreiter Sepp and Schmidhuber Jurgen. Long Short Term Memory. *Neural Computation*, 9(8):1–32, 1997.

[19] J. Schmidhuber and F. A. Gers. LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[20] Léon Bottou. Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, pages 177–186, 2010.

[21] Ian J Goodfellow, Jean Pouget-abadie, Mehdi Mirza, Bing Xu, and David Wardefarley. *Generative Adversarial Nets*. PhD thesis.

[22] Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:5967–5976, 2017.

[23] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:105–114, 2017.

[24] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. 2016.

[25] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. 2016.

[26] Sebastian Ruder. An overview of gradient descent optimization. pages 1–14, 2016.

[27] Dean A Pomerleau. ALVINN: an autonomous land vehicle in a neural network.

Advances in Neural Information Processing Systems 1, pages 305–313, 1989.

[28] Sing Bing Kang and Katsushi Ikeuchi. Toward automatic robot instruction from perception - Mapping human grasps to manipulator grasps. *IEEE Transactions on Robotics and Automation*, 13(1):81–95, 1997.

[29] Aude Billard and Daniel Grollman. Robot learning by demonstration. *Scholarpedia*, 8(12):3824, 2014.

[30] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Autonomous Helicopter Flight Using Reinforcement Learning. *Encyclopedia of Machine Learning and Data Mining*, pages 75–85, 2004.

[31] Nathan Ratliff and Joel Chestnutt. Boosting Structured Prediction for Imitation Learning. *Advances in Neural Information Processing Systems 19*, (January 2006), 2018.

[32] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, 2009.

[33] R Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering,*, 21(2):125–147, 1960.

[34] Andrew Y. Ng and S Russel. Algorithm for Inverse Reinforcement learning. *in Proc. 17th International Conf. on Machine Learning*, pages 663–670, 2000.

[35] Richard S Sutton and Andrew G Barto. Time-Derivative Models of Pavlovian Reinforcement. *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, (Mowrer 1960):497–537, 1990.

[36] Prasad Tadepalli and Dokyeong Ok. H-learning : A Reinforcement Learning Method to Optimize Undiscounted Average Reward 1 Introduction. *Most*, (March 1996):1–17, 1994.

[37] Sridhar Mahadevan. An Average-Reward Reinforcement. *AAAI-96 Proceedings.*, pages 875–880, 1996.

[38] Brian Ziebart, Andrew Bagnell, and Andrew Maas. Maximum Entropy Inverse Reinforcement Learning Brian. *Twenty-Third AAAI Conference on Artificial Intelligence (2008)*, pages 1433–1438, 2008.

[39] B.D. Ziebart, J.A. Bagnell, and A.K. Dey. Modeling interaction via the principle of maximum causal entropy. *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010.

[40] E.T Jaynes. Inforamation Theory and Statistical Mechanics. Technical report, 1986.

[41] Miroslav Dudík and Robert E. Schapire. Maximum Entropy Distribution Estimation with Generalized Regularization. pages 123–138, 2006.

[42] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum Entropy Deep Inverse Reinforcement Learning. 2015.

[43] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative Entropy Inverse Reinforcement Learning Abdeslam. *I4th International Conference on Artificial Intelligence and Statistics*, 15:182– 189, 2011.

[44] Ming Jin, Andreas Damianou, Pieter Abbeel, and Costas Spanos. Inverse Reinforcement Learning via Deep Gaussian Process. 2015.

[45] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. 48, 2016.

[46] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. (Nips), 2016.

[47] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. 2015.

[48] Market Assessment, Web Site, and Project Infrastructure. NGSIM Overview. (December):2–3, 2006.

[49] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[50] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. *The International Journal of Robotics Research*, (October):1–6, 2013.

[51] Federal Highway Administration. Interstate 80 Freeway Dataset Factsheet. *Fhwa-Hrt-06-137*, (December):1–2, 2006.

[52] FHWA. NGSIM US Highway 101 Dataset. page 3, 2007.

[53] Coifman Benjamin and Lizhie Li. A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset. *Transportation Research Part B: Methodological*, 105:362–377, 2017.

[54] Tomer Toledo and David Zohar. Modeling Duration of Lane Changes. *Transportation Research Record: Journal of the Transportation Research Board*, 1999(1):71–78, 2007.

[55] William Fedus, Ian Goodfellow, and Andrew M. Dai. MaskGAN: Better Text Generation via Filling in the_____. 2018.

[56] Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph Lim. Multi-Modal Imitation Learning from Unstructured Demonstrations using Generative Adversarial Nets. 2017.

[57] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. (Goodfellow), 2016.

[58] Seong Hyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018-June:1672–1678, 2018.

[59] Mohamed Akram Zaytar and Chaker El Amrani. Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks. *International Journal of Computer Applications*, 143(11):7–11, 2016.