PREDICTING HUMAN BEHAVIOR USING STATIC AND DYNAMIC MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

By
Berat Mert Albaba
August 2021

Predicting human behavior using static and dynamic models
By Berat Mert Albaba
August 2021

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Yıldıray Yıldız(Advisor)

Hamdi Dibeklioğlu

Mehmet Önder Efe

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

PREDICTING HUMAN BEHAVIOR USING STATIC AND DYNAMIC MODELS

Berat Mert Albaba
M.S. in Mechanical Engineering
Advisor: Yıldıray Yıldız
August 2021

Modeling human behavior is a challenging problem and it is necessary for the safe integration of autonomous systems into daily life. This thesis focuses on modeling human behavior through static and dynamic models. The first contribution of this thesis is a stochastic modeling framework, which is a synergistic combination of a static iterated reasoning approach and deep reinforcement learning. Using statistical goodness of fit tests, the proposed approach is shown to accurately predict human driver behavior in highway scenarios. Although human driver behavior are modeled successfully with the static model, the scope of interactions that can be modeled with this approach is limited to short duration interactions. For interactions that are long enough to induce adaptive behavior, we need models that incorporate learning. The second contribution of this thesis is a learning model for time extended human-human interactions. Through a hierarchical reasoning solution approach, equilibrium concepts are combined with Gaussian Processes to predict the learning behavior. As a result, a novel bounded rational learning model is proposed.

Keywords: Reinforcement Learning, Game Theory, Autonomous Vehicles.

ÖZET

STATIK VE DINAMIK MODELLER ILE INSAN DAVRANIŞININ TAHMINI

Berat Mert Albaba
Makine Mühendisliği, Yüksek Lisans
Tez Danışmanı: Yıldıray Yıldız
Ağustos 2021

İnsan davranışını modellemek, otonom sistemlerin günlük hayata güvenli entegrasyonu için gerekli olan zor bir problemdir. Bu tez, insan davranışlarını statik ve dinamik modeller aracılığıyla modellemeye odaklanmaktadır. Bu tezin ilk katkısı, statik yinelenen akıl yürütme yaklaşımı ile pekiştirmeli öğrenmenin sinerjik bir birleşimi olan stokastik bir modelleme çerçevesidir. Önerilen yaklaşım, otoyol senaryolarında insan sürücü davranışlarını başarılı bir şekilde modellemektedir ve otonom araçların entegrasyon senaryolarının doğrulanması için kullanılabilir. Her ne kadar insan sürücü davranışları statik bir modelle başarılı bir şekilde modellense de, statik modeller, ajanların etkileşimler sırasında inançlarını/varsayımlarını güncelleyebileceklerini hesaba katmamaktadır. Bu tezin ikinci katkısı, zamana yayılan insan-insan etkileşimleri için yeni bir öğrenme modelidir. Hiyerarşik bir akıl yürütme çözüm kavramı aracılığıyla, denge kavramları, Gaus süreçleri ile birleştirilirmiştir. Sonuç olarak, özgün ve sınırlı rasyonel bir öğrenme yaklaşımı önerilmiştir.

Anahtar sözcükler: Pekiştirmeli Öğrenme, Oyun Teorisi, Otonom Araçlar.

Acknowledgement

First and foremost, I am sincerely grateful to my adviser, Asst. Prof. Yildiray Yildiz for his continuous support, motivation, patience, and invaluable advice. This thesis and my research would not be possible without his support and guidance. His knowledge and experience encouraged me all the time of my life as an undergraduate and graduate student. It has been a great pleasure to work under his guidance.

I would like to thank my thesis committee, Asst. Prof. Hamdi Dibeklioglu and Prof. Mehmet Onder Efe for allocating their time to read and investigate my thesis.

I would like to express my gratitude to my dear friends Emre Mumcuoglu, Omer Ekmekcioglu, Efsun Kavaklioglu, Alihan Huyuk and Deniz Akkaya. Their friendship, support, and valuable advice changed my life in difficult times. I also thank all my colleagues that I could not list here for their support and great friendships.

Last but not least, I would like to thank my parents Neval and Metin, and my sister Nevra for their endless love, care, support, and for always believing in me in all stages of my life.

Contents

1 Introduction **1**

1.1 Contributions 5

1.1.1 Stochastic Modeling Approach 5

1.1.2 Learning Model for the Time Extended Human-Human Interactions 5

1.2 Organization 6

2 Modeling Driver Behaviors via Reinforcement Learning and Game Theory **7**

2.1 Methods 8

2.1.1 Level-k Reasoning 8

2.1.2 Deep Q-Learning 9

2.1.3 Combining Level-k Reasoning with Deep Q Learning 15

2.2 Traffic Scenario 18

2.2.1 Traffic Data Processing 18

2.2.2 Driver Observation Space 20

2.2.3 Driver Action Space 23

2.2.4 Physical Model of Vehicles 24

2.2.5 Vehicle Placements 25

2.2.6 Reward Function 25

2.3 Training and Simulation 26

2.3.1 Level-0 Policy 27

2.3.2 Training Performance 28

2.3.3 Simulation Performance 29

2.4 Validation with Traffic Data 30

2.4.1 Kolmogorov-Smirnov Test for Discontinious Distributions . . 31

2.4.2 Comparing game theoretical models with traffic data 34

2.4.3 Results 36

3 GP-k: Learning Model for Time-Extended Human-Human Interactions 55

3.1 Building Blocks 55

3.1.1 Level-k Reasoning 56

3.1.2 Reinforcement Learning 56

3.1.3 Combination of RL with Level-k 56

3.1.4 Gaussian Process 57

3.2 GP-k 57

3.3 Applications 62

3.3.1 Optimal Behavior via Level Inference 62

3.4 Sensitivity Analysis 63

4 Conclusion **65**

List of Figures

2.1 Level-k reasoning 9

2.2 Reinforcement learning process diagram 10

2.3 Neural network architecture 12

2.4 Combination of Level-k reasoning and reinforcement learning. 17

2.5 Space headway distribution 21

2.6 Traffic scenario 23

2.7 Acceleration distribution 24

2.8 Training rewards for DQN 28

2.9 Training rewards for c-DQN 28

2.10 Crash rates for DQN 29

2.11 Crash rates for c-DQN 30

2.12 Comparison results for $n_{limit} = 3$ and $RL_{method} = DQN$ (US101) . . . 38

2.13 Comparison results for $n_{limit} = 5$ and $RL_{method} = DQN$ (US101) . . . 40

2.14 Comparison results for $n_{limit} = 3$ and $RL_{method} = c-DQN$ (US101) . 42

2.15 Comparison results for $n_{limit} = 5$ and $RL_{method} = c-DQN$ (US101) . 44

2.16 Comparison results for $n_{limit} = 3$ and $RL_{method} = DQN$ (I80) 46

2.17 Comparison results for $n_{limit} = 5$ and $RL_{method} = DQN$ (I80) 48

2.18 Comparison results for $n_{limit} = 3$ and $RL_{method} = c-DQN$ (I80) . . . 49

2.19 Comparison results for $n_{limit} = 5$ and $RL_{method} = c-DQN$ (I80) . . . 51

3.1 Evaluation of a learning curve of an agent 58

3.2 Performance of the learning model with varying variance parameter. . 64

List of Tables

2.1 Driver modeling performances of the proposed GT policies and the existing approaches for US101 data. 53

2.2 Driver modeling performances of the proposed GT policies and the existing approaches for I80 data. 54

Chapter 1

Introduction

In multi-agent games, each agent aims to reach the optimal policy in order to optimize his/her payoff. Since there is more than one strategic agent in the environment, analyzing games in these environments is challenging. Game theory is mostly focused on the equilibrium analysis [1] [2], such as finding Nash equilibrium [3]. Equilibrium concepts forecast the converged state of the game; however, as explained in [4], such concepts fail to predict player behaviors in complex games and the early stages of games. Thus, in order to model initial actions in games, hierarchical reasoning solution concepts named Level-k reasoning and Cognitive Hierarchy Theory are proposed in [5], [6], and [7], where players move based on their initial assumptions about others during the game.

In this thesis, first of all, level-k reasoning is combined with deep reinforcement learning in order to develop driver models. High-fidelity driver models are required to accelerate the validation of autonomous vehicle (AV) technology. Safety concerns exist about the integration of AV into daily traffic, and these concerns should be addressed before their integration. Simulated traffic scenarios can be utilized to accelerate the validation of AV and introduce a rich variety of traffic scenarios which may take several million driving hours to encounter in real life [8], [9], [10], [11]. However, for reliable results, human driver models in simulations should have high-fidelity, i.e., should behave like human drivers.

Several approaches are proposed in the open literature to obtain high-fidelity human driver models. Markov Dynamic Models (MDMs) are utilized for the prediction and recognition of the driver maneuvers, such as steering or braking, in [12]. An approach named SITRAS (Simulation of Intelligent Transport Systems) is proposed in [13] for modeling lane changing and merging behaviors. Dynamic Bayesian Networks (DBNs) are presented in [14] for recognition of acceleration or lane changing actions. In [15] adaptive predictive control is proposed for modeling drivers' lateral actions. The cognitive architecture method, which is a framework that specifies computational behavioral models of human cognition, is presented in [16] for steering and lateral position modeling. A combination of cognitive psychology and artificial intelligence methods is utilized in [17] for predicting behaviors of human drivers at signalized and unsignalized intersections. In [18], a support vector machine (SVM) with a Bayesian filter is utilized for predicting lane change intentions of drivers. In [19], and [20], Gaussian Process Regression is employed to predict driver behaviors and detect behavioral patterns. For the estimation of driver decisions at intersections, a framework that models vehicle dynamics and driver behaviors as a hybrid-state system is proposed in [21]. In order to obtain driving styles from vehicle trajectories, inverse reinforcement learning is used in [22]. Hidden Markov Model-based driver models are derived in [23]. In [24], SVMs are used to model driving patterns. A neural network-based method, which utilizes recurrent neural networks, is presented in [25] for probabilistic driving modeling. An SVM-based model is given in [26] for predicting driving behavior at roundabouts. For modeling human highway driving behavior, Generative Adversarial Imitation Learning is proposed in [27]. Human drivers' stopping behaviors are modeled in [28]. Also, in [29], an optimal control theory-based approach is presented to model stopping behavior. Gaussian mixture model and hidden Markov model are combined in [30] for the prediction of lateral trajectories. A car-following driver model imitating human driving is proposed in [31], using a neural network-based modeling approach. In [32], interactions between an autonomous vehicle and a human driver are modeled. Apart from individual driver models, traffic flow models are also studied. Some examples can be found in [33], [34] and [35].

In the first part of this thesis, an iterated reasoning concept is combined with a deep reinforcement learning method. A stochastic modeling approach is proposed to model

human driver behaviors.

Although iterated reasoning concepts model initial human behaviors better, it misses the point that players may alter their assumptions during interactions, and it is unlikely to reach equilibrium by introspection [4]. Further, analyzing the process of going to equilibrium is avoided in most of the history [36].

Learning models try to explain how players learn. Learning models are widely used as alternatives to equilibrium models [37]. These models are categorized into two categories [37]: 1. reinforcement learning models and 2. belief-based learning models. In reinforcement learning models, strategies receive reinforcement based on their payoffs, and players adjust their strategies to the ones that offer higher payoffs [37]. Belief-based learning models are the models in which agents in a dynamic game optimize their actions based on their beliefs/predictions about others in the environment, using history of previous actions [38]. Best response dynamics proposed in [39] is one of the oldest approaches in belief learning. In [39], players assume that others would repeat their action in the previous time and respond best based on this assumption. One of the most popular belief learning approaches is fictitious play, which is first proposed in [40] and [41]. In modern fictitious play variants, as explained in [42], players first form a belief in other agents' action probabilities, observe the taken actions, and updates the initial probability with the frequency of actions. In the fictitious play, players best respond based on their beliefs. Because of the assumption that agents behaves optimally based on their beliefs, fictitious play may not be realistic [43]. Thus, a weighted fictitious play, or stochastic fictitious play, is proposed in [44]. In weighted fictitious play and its variants, past observations are weighted, i.e., considers the amount of time passed after taken actions. Further, based on this weighted belief, players choose the response probabilistically by calculating the effect of each action on the payoff based on the belief. Besides, Bayesian learning approaches also exist, in which beliefs are probability distributions over possible strategies of other agents [42]. As presented in [45], in naive Bayesian learning, beliefs over future action sequences are updated directly through observations with Bayes' theorem, and agents best respond to their beliefs. In sophisticated Bayesian learning, beliefs on future action sequences are derived from prior beliefs on other players' payoff functions [45]. Reinforcement learning and

belief learning approaches are combined in [46] to use the payoff guide of reinforcement learning in belief learning. Experimental studies also exist for belief learning. In [47], weighted fictitious play is studied empirically, and subjects are classified by their fictitious play type. In [48], beliefs about the environment and the previous ego action choices are combined to model human behavior. In [49], Bayesian learning is studied empirically. It is presented that when there is a unique equilibrium of pure strategies, equilibrium play is attained. However, when there are multiple equilibria, the play failed to converge the predicted equilibrium.

With the developments in single-agent reinforcement learning, multi-agent reinforcement learning (MARL) field is experiencing rapid progress [50]. However, in multi-agent settings, if the agent ignores how other agents behave, and optimizes its policy, the policy being trained may fail to converge [50], [51], [52]. Thus, introducing beliefs into MARL is widespread, which allows predicting the behaviors of other agents. In [53] and [54], hierarchical beliefs are utilized in multi-agent settings. In [55], an approach to respond well against sophisticated agents is built over [53] and uses Bayesian learning. Bayesian learning is also used in [56] for the cooperative partially observable multi-agent games. Interactive partially observable Markov decision process (I-POMPD), which extends POMDPs to multi-agent settings, is proposed in [57] and also utilizes belief learning. The work presented in [58] builds on I-POMDPs to predict other agents' intentions by using Bayesian learning. In [59], Bayesian learning is also used to infer the intents of other agents. In [60], a framework is presented for MARL which also includes belief learning.

Although belief learning approaches are generally used in multi-agent reinforcement learning today and improved the field significantly, belief learning models are computationally complex in multi-agent settings. There are also naive belief learning approaches that can be considered efficient, such as weighted fictitious play. However, in these models, players are myopic, i.e., do not consider future expected rewards.

In the second part of this thesis, a hierarchical solution concept, level-k thinking, belief learning, and equilibrium concepts are combined. A new learning model for the time extended human-human interactions is proposed.

1.1 Contributions

1.1.1 Stochastic Modeling Approach

What distinguishes our method from existing studies is that all the drivers in a multi-move scenario make strategic decisions simultaneously, instead of modeling the ego driver as a decision-maker and assuming predetermined actions for the rest of the drivers. This is achieved by combining a hierarchical game-theoretical concept named *level-k reasoning* with a reinforcement learning method called *Deep Q-learning* (DQN) [61]. There exist earlier studies that also use reinforcement learning and game theory in modeling driver behavior, such as [62], [63] [8], [64] and [65]. The contribution of this work over these earlier results can be listed as

1. The proposed method can successfully model a dramatically larger class of scenarios which was not possible earlier.
2. The driver crash rates in the simulations are reduced to realistic levels by eliminating the driver blind spots with an enlarged observation space;
3. Developed driver models are compared with two different sets of traffic data, and a significantly larger percentage of human driver behaviors are successfully modeled.
4. The proposed models are compared with the baseline models, IDM [66], MOBIL [67], and the models in the previous work [62]. Comparisons showed that the proposed models predict human driver behaviors better than the existing approaches.

1.1.2 Learning Model for the Time Extended Human-Human Interactions

In this model, agents in the environment behave according to continuous level-k policies [68]. The initial and the endpoint policies are assumed to be known by the player.

Through level-k reasoning, the level-k agent assumes that other players' level is (k-1) initially, which provides the initial point. Further, it is assumed that the endpoint (end of learning) is known, which may be converged equilibrium points such as Nash equilibrium. It is noted that the endpoint type is not important in the proposed approach since finding equilibrium is not the purpose of this work. Any converged point may be utilized. To exemplify, the converged point of a MARL approach may be utilized as the endpoint when a MARL agent is placed in a test environment. The proposed learning model utilizes Gaussian processes to form and update predictions about players during the game. The contribution of the proposed learning model over the existing literature is four-fold:

1. Efficient in complex environments since agents do not consider the effect of their actions on others' future actions
2. The initial and end-points make the model farsighted and prevents deviating from the existing policy based on some irrational observations,
3. The level-k component in the model offers bounded rationality during the whole game,
4. Gaussian processes allow introducing uncertainty to each observation easily.

1.2 Organization

Stochastic modeling framework for modeling human driver behaviors is explained in Chapter II. Methods utilized to build the framework, details of the traffic scenario, and the simulations are described in detail. The proposed human driver models are validated with real data, which is discussed at the end of Chapter II. The learning model for modeling the time extended human-human interactions is explained in Chapter III. Sensitivity analysis of the learning model is presented at the end of Chapter III. In Chapter IV, a conclusion is made.

Chapter 2

Modeling Driver Behaviors via Reinforcement Learning and Game Theory

A synergistic combination of deep reinforcement learning and hierarchical game theory is proposed as a modeling framework for the behavioral modeling of drivers in highway driving scenarios. The primary motivation behind this work is the need for a modeling framework that can address multiple human-human and human-automation interactions, where all the agents can be modeled as decision-makers simultaneously. The modeling framework presented in this paper may be used in a high-fidelity traffic simulator consisting of multiple human decision-makers to reduce the time and effort spent for testing by allowing safe and quick assessment of self-driving algorithms. To demonstrate the fidelity of the proposed modeling framework, game-theoretical driver models are compared with real human driver behavior patterns extracted from two different sets of traffic data. Statistical goodness of fit tests are used for the comparisons.

2.1 Methods

2.1.1 Level-k Reasoning

In order to model strategic decision making process of human drivers, a game-theoretical concept named *level-k reasoning* is used [7], [6], [69]. The level-k approach is a hierarchical decision-making concept and presumes that different levels of reasoning exist for different humans. The lowest level of reasoning in this concept is called *level-0 reasoning*. A level-0 agent is a non-strategic/naive agent since his/her decisions are not based on other agents' possible actions but consist of predetermined moves, which may or may not be stochastic. A strategic level-1 agent exists one level higher, who determines his/her actions by assuming that the other agents' reasoning levels are level-0. Hence, the actions of a level-1 agent are the best responses to level-0 actions. Similarly, a level-2 agent considers other agents as level-1 and makes his/her decisions according to this prediction. This process continues following the same logic for higher levels. In Fig. 2.1 the general architecture of level-k thinking is presented. The figure shows a population consisting of level-0 (gray), level-1(blue), and level-2(yellow) agents. In this population, level-0's are non-strategic, meaning that they do not consider other agents' possible actions before making their move. Strategic level-1 and level-2 agents, on the other hand, assume that everyone else is level-0 and level-1, respectively, and act accordingly. In some experiments, humans are observed to have at most level-3 reasoning [70], which may, of course, depend on the type of the game being played. To generalize, all level-k agents, except level-0, presume that the rest of the agents are level-(k-1) and make their decisions based on this belief. Since this belief may not always hold true, the agents have bounded rationality.

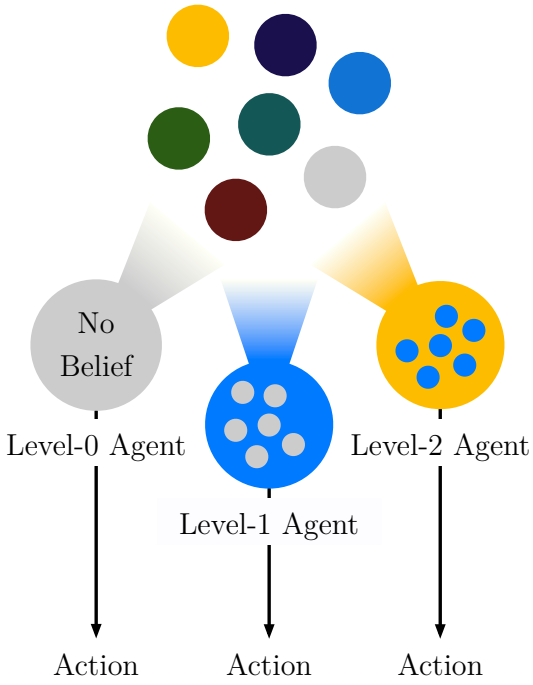


Figure 2.1: A level-0 agent (grey) does not consider other agents' possible moves before making a decision; a level-1 agent (blue) assumes that all other agents are level-0 and takes action based on this assumption; a level-2 agent (yellow) believes that other agents are level-1 and acts accordingly.

2.1.2 Deep Q-Learning

In the previous section, the exploitation of level-k reasoning to model strategic decision-making is explained. In time-extended scenarios, where the agents make a series of decisions before an episode is completed, such as the traffic scenarios focused on in this paper, level-k reasoning can not be used alone. To obtain driver models that provide the best responses to the other agents' alleged actions in a multi-move setting, we utilize Deep Q-Learning (DQN) together with level-k reasoning. The main reason for the employment of DQN is the large state space that becomes infeasible to handle with other reinforcement learning (RL) methods used in earlier studies [62], [8], [64], [65]. In this section, a brief description of DQN tailored for the task at hand is given. More detailed expositions of DQN can be found at [61], [71] and [72].

RL is a learning process through reward and punishment [73]. At each step of learning, or training, the trained agent observes a state $s_t \in S$ (state s at time t), selects an action $a_t \in A$ (action a at time t), and as a result, transitions into a new state $s_{t+1} \in S$ with a certain probability. The sets S and A represent observation and action spaces, respectively. After the transition, the agent receives a reward based on a reward function, which is a mathematical expression of the agent's preferences. This process, i.e., observation, action, transition, and reward accumulation, continues until the average reward converges to a certain value. The process is depicted in Fig. 2.2. The main goal of the agent is selecting actions that maximize the total obtained rewards, where future rewards may have decreasing levels of importance in the agent's decision making and thus, can be "discounted". In other words, the agent tries to find a policy $\pi : S \longrightarrow A$, a mapping from states to actions, which maximizes the expected discounted cumulative reward. This policy is called the optimal policy: π^*. In RL, value function $V(s)$ estimates the cumulative reward starting from state s. It is a measure of the value of being in state s. Action value function, $Q(s,a)$, estimates the cumulative reward obtained by the agent by starting with action a in state s. It represents how valuable selecting action a in state s is.

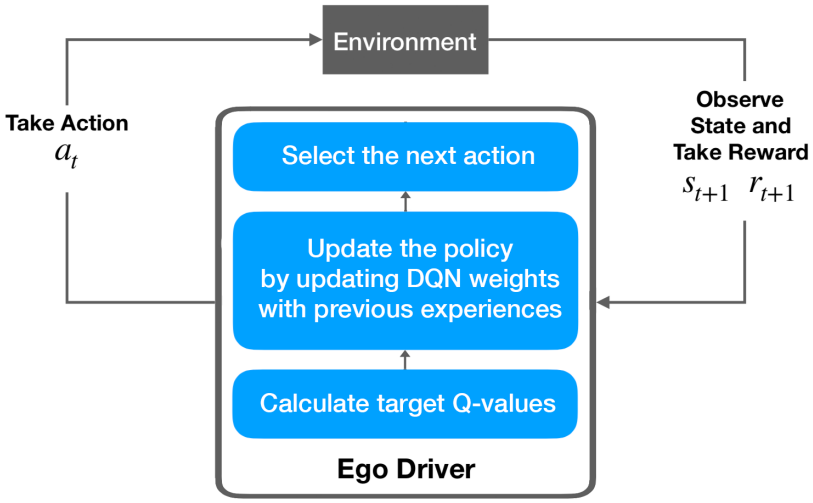


Figure 2.2: Reinforcement learning process diagram

An agent's value function, or expected cumulative discounted reward when starting from state s, and following a policy π is defined as

$$V^{\pi}(s) = E(\sum_{t \geq 0} \gamma^{t} r_t), \tag{2.1}$$

for all $s \in S$, where γ represents a discount factor, r_t represents the reward at time-step t, and s represents the observed state. The corresponding optimal value function is given as

$$V^{*}(s) = \max_{\pi} V^{\pi}(s). \tag{2.2}$$

The optimal value function is related to the optimal Q-function, Q^{*}, as

$$V^{*}(s) = \max_{a} Q^{*}(s,a). \tag{2.3}$$

Moreover, the optimal policy can be defined in terms of the optimal Q-function as

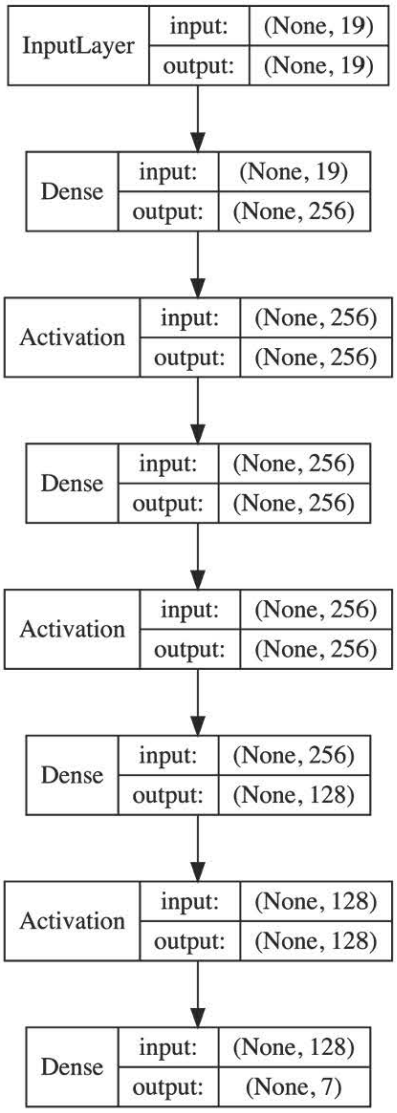
$$\pi^{*}(s) = arg\max_{a} Q^{*}(s,a). \tag{2.4}$$

Bellman equation provides a relationship between the value of a state-action pair *(s,a)* and its successor pairs through

$$Q(s,a) = E[r_{t+1} + \gamma \max_{a^{'}} Q(s_{t+1}, a^{'}) \mid s_t = s, a_t = a], \tag{2.5}$$

where E represents the expected value and $(s_{t+1}, a^{'})$ is the next state-action pair. *Q-learning* algorithm provides a method, based on the Bellman Equation (2.5), to update action values iteratively, which is given by

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_{t+1} + \\ \gamma \max_{a} Q_t(s_{t+1}, a) - Q_t(s_t, a_t)), \end{aligned} \tag{2.6}$$



InputLayer	Takes state representations as input
Dense	A fully connected layer, i.e., there is a connection between each input feature and each node of the layer
Activation	Implements activation functions, which introduce non-linearity
None	There are none values in each shape since the network accepts different batch sizes, i.e., none values are automatically replaced by the given batch size.

Figure 2.3: A deep neural network consisting of 4 layers is utilized to approximate the Q-function.

where α is a learning rate and γ is the discount factor. Q-learning is guaranteed to converge if each state-action pairs are visited infinitely many times during training [73].

Keeping a table containing action values for all state-action pairs may be infeasible for large state spaces. In these cases, a neural network function approximator with a weight vector θ can be used to approximate the action-value function $Q(s,a) \approx Q(s,a;\theta)$ [71]. In Deep Q-Learning (DQN), values of Q-function are approximated with a deep neural network, which proves useful, especially for large state spaces. The neural network architecture utilized in this work is given in Fig. 2.3.

DQN stores last N experiences in memory D. Each experience is a four-tuple: $(s,a,s^{'},r)$, where a is the action made in state s, $s^{'}$ is the transitioned state after taking the action a, and r is the obtained reward. In the first n_s steps of training, network weights are not updated. Starting from the n_{s+1}^{st} step, at each step, a mini-batch of the stored four-tuples is randomly sampled from memory D, and the Q-function is updated through network weight updates.

Algorithm 1 Deep Q-Learning

Initialize the memory D to capacity N

Initialize the main network and the target network, QN and QN^T, with weights sampled from a uniform distribution of range $[-\sqrt{\frac{6}{n_{input}+n_{output}}}, \sqrt{\frac{6}{n_{input}+n_{output}}}]$, where n_{input} and n_{output} are number of input and output neurons, respectively [74].

Set $T = 50$

for episode = 1 to M **do**

  for t = 1 to K **do**

    Sample action a_t using the probability values $P_t(a_i) = \frac{e^{QN_t(a_i)/T}}{\sum_{j=0}^{n-1} e^{QN_t(a_j)/T}}$, $i = 1, 2, ..., size(ActionSpace)$

    Execute action a_t and observe the reward r_t and the transitioned state s_{t+1}

    Store the experience (s_t, a_t, r_t, s_{t+1}) in D

    if size(D) $\geq n_s$ **then**

      Sample a random batch of experiences, consisting of P four-tuples (s_j, a_j, r_j, s_{j+1})

      for j = 1 to P **do**

        Set $y_j = r_j + \gamma \max_{a'} QN^T(s_{j+1}, a'; W)$

        if s_{j+1} is terminal, i.e. ego driver crashes, **then**

          Set $y_j = r_j$

        end if

        Perform a gradient descent step using the cost function$(y_j - NN(s_{j+1}, a_j; W))^2$ with respect to weight matrix W

      end for

    end if

    if s_{t+1} is terminal, i.e. ego driver crashes, **then**

      break

    end if

  end for

  if $T > 1$ **then**

    Update Boltzmann Temperature $T = Tc$, $c < 1$

  end if

end for

Boltzmann exploration, i.e., softmax, with an initial temperature of $T = 50$, which decreases exponentially to 1 during the training process, is used to obtain a stochastic neural network instead of a deterministic one. In Boltzmann exploration, during training, probabilities of actions are calculated with the Boltzmann distribution, and actions are taken probabilistically. At time-step t, probability of taking action a is

$$P_t(a) = \frac{e^{Q_t(a)/T}}{\sum_{i=0}^{n-1} e^{Q_t(a_i)/T}}, \tag{2.7}$$

where T is the temperature, and n represents the number of actions [75]. When the temperature is high, the probabilities of the actions are closer to each other. When the temperature is low, the probability of the action with the highest Q-value is higher than the rest of the actions.

Algorithm pseudo-code for DQN is given in Algorithm 1, where M is the total number of training episodes, K is the number of steps within an episode, P is the number of four-tuples in a mini-batch, and T is the temperature in Boltzmann distribution.

2.1.3 Combining Level-k Reasoning with Deep Q Learning

To generate agents with different levels of reasoning for modeling multi-move strategic decision-making in traffic scenarios, the learning capability offered by DQN is combined with the level-k reasoning approach.

In the proposed combined approach, the predetermined, non-strategic level-0 policy is the anchoring policy from which all the higher levels are derived using Deep Q-learning (DQN). For example, in order to obtain the level-1 policy, a traffic scenario is created where all drivers are level-0 agents except the *ego driver*, who is to be trained via DQN to learn how to best respond to the level-0 policy. The details of this training are given in Section II-B. Once the training is over, the ego driver becomes a level-1 agent. The procedure for obtaining the level-1 policy through the proposed combination of level-k reasoning and DQN is explained in Algorithm 2, where n_d is the number of drivers.

Algorithm 2 Obtaining the level-1 policy by combining DQN and Level-k Reasoning

1: Load the predetermined level-0 policy, π^0
2: Set the reasoning levels of all agents in the environment, p_i, to level-0: $\pi_{p_i} = \pi^0$, $i = 1,2,3,...,n_d$
3: Initialize the ego driver's policy to a uniform action probability distribution over all states: $\pi_{ego} = \pi^{uniform}$
4: Train the ego driver using DQN
5: At the end of the training, ego driver learns to best respond to π^0, and therefore the resulting policy is the level-1 policy, π^1.

The level-2 policy can be obtained similarly: A level-2 player assumes that all other players in the environment are level-1 and takes actions based on this assumption. Therefore his/her actions are the best responses to level-1 players. For the training of the level-2 agent, a traffic scenario where all the agents, except the ego driver, are assigned the previously obtained level-1 policy. Using DQN, the ego driver is trained to give the best responses to level-1 drivers. Hence the resulting policy becomes the level-2 policy. This process is given in Algorithm 3

Algorithm 3 Training of the level-2 agent by combining DQN and Level-k Reasoning

1: Load the previously obtained level-1 policy, π^1 (see Algorithm 2)
2: Set the agents in the environment, p_i, as level-1 agents: $\pi_{p_i} = \pi^1$, $i = 1,2,3,...,n_d$
3: Initialize the ego driver's policy to a uniform action probability distribution over all states: $\pi_{ego} = \pi^{uniform}$
4: Train the ego driver using DQN
5: At the end of the training, ego driver learns to best respond to level-1 policy, π^1. Thus, the resulting policy is the level-2 policy, π^2.

Algorithm 4 Training of the level-k agent by combining RL and Level-k Reasoning

1: Load the level-(k-1) policy, π_{k-1}
2: Assign level-(k-1) policy to the agents in the environment, p_i, as: $\pi_{p_i} = \pi^{k-1}$
3: Place level-(k-1) agents in the environment
4: Set empty policy to the learning agent: $\pi_{ego} = \pi^{empty}$
5: Train the learning agent with DQN
6: At the end trained agent learns best responses to the level-(k-1), so save the resulting policy as the level-k policy, π_k

Higher-level policies can also be obtained in a similar way if desired. The training process for obtaining a level-k policy is described in Algorithm 4. It is noted that the hierarchical learning process explained above decreases the computational cost since, at each stage of learning, the agents other than the ego agent use previously trained policies and hence become parts of the environment. This helps to obtain traffic scenarios containing a mixture of different levels where all the agents simultaneously make strategic decisions. This framework sharply contrasts conventional driver decision-making approaches in heavy traffic, where one or two drivers are strategic decision-makers, and the rest are assigned predefined policies that satisfy certain kinematic constraints. In this work, the highest level is set to level-3, following [70]. A visual representation of the process of combining level-k reasoning and DQN is given in Fig. 2.4

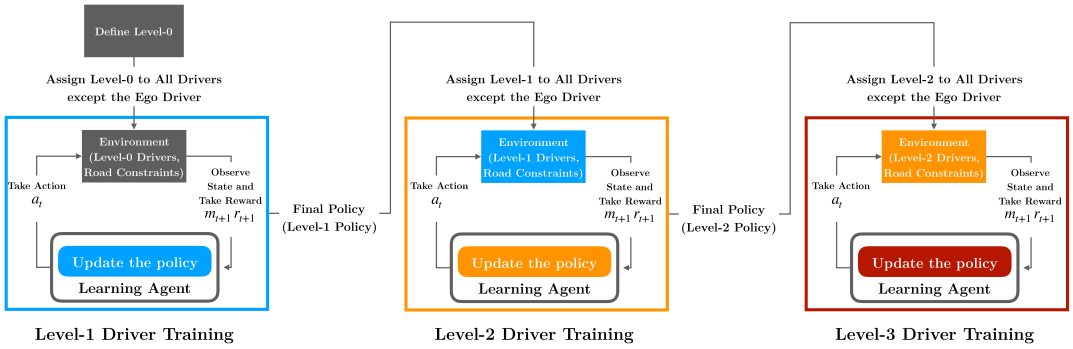


Figure 2.4: Combination of Level-k reasoning and reinforcement learning.

2.2 Traffic Scenario

The traffic scenario used to train and test driving policies comprises a five-lane highway and multiple vehicles. The lane width of the highway is 3.7m, and the size of each vehicle is 5m x 2m. The vehicles have continuous dynamics. In the following subsections, the elements of the traffic scenario are explained. Specific numerical values needed to create the traffic scenario, such as observation and action space parameters, are determined based on one of the two sets of traffic data. Therefore, we first explain how data is processed before providing the scenario details. It is noted that in this section, we only describe data processing for use in determining specific traffic scenario parameters. Processing of the data to obtain real human driver policies is explained later in Section IV.

2.2.1 Traffic Data Processing

In this work, two sets of traffic data, collected on US101 and I80 highways [76], [77], are used for model validation. Among these two, the US101 set is employed to determine the observation and action space parameter values. These data sets consist of time and vehicle ID stamped position, velocity and acceleration values. Before employing the data, firstly, inconsistencies in the acceleration and velocity values are addressed: A careful analysis demonstrates some contradictions in acceleration values and large velocity jumps over consecutive time steps in the traffic data. The problem of large velocity jumps is solved by applying a linear curve fitting. To exemplify, if among the velocity values v_{i-5}, v_{i-4}, ..., v_i, v_{i+1}, v_{i+2}, ..., v_{i+5}, where the subscripts denote the time steps, the values v_{i+1} and v_{i+2} showed impossible jumps, these values are replaced with appropriate values $v_{i+1} = v_i + \frac{v_{i+3} - v_i}{3}$ and $v_{i+2} = v_i + \frac{2(v_{i+3} - v_i)}{3}$. Once unreasonable velocity jumps were eliminated, acceleration values are obtained by using the five-point stencil method [78], [79] given as

$$a_i = \frac{-v_{i+2\delta i} + 8v_{i+\delta i}}{12\delta i} + \frac{-8v_{i-\delta i} + v_{i-2\delta i}}{12\delta i}. \tag{2.8}$$

Since $v_{i+2\delta i}$, $v_{i+\delta i}$, $v_{i-\delta i}$ and $v_{i-2\delta i}$ do not exist for the first and the last two time-steps for each car, 5-point finite difference method is used to calculate the acceleration values for these points. Using this method, the first and last time step accelerations are calculated as

$$a_i = \frac{-25v_i + 48v_{i+\delta i} - 36v_{i+2\delta i}}{12\delta i} + \frac{16v_{i+3\delta i} - 3v_{i+4\delta i}}{12\delta i} \tag{2.9}$$

$$a_i = \frac{3v_{i-4\delta i} - 16v_{i-3\delta i} + 36v_{i-2\delta i}}{12\delta i} + \frac{-48v_{i-\delta i} + 25v_i}{12\delta i}, \tag{2.10}$$

respectively. Similarly, for the second and second-to-last time steps, the accelerations are calculated as

$$a_i = \frac{-3v_{i-\delta i} - 10v_i + 18v_{i+\delta i}}{12\delta i} + \frac{-6v_{i+2\delta i} + v_{i+3\delta i}}{12\delta i} \tag{2.11}$$

$$a_i = \frac{-v_{i-3\delta i} + 6v_{i-2\delta i} - 18v_{i-\delta i}}{12\delta i} + \frac{10v_i + 3v_{i+\delta i}}{12\delta i}, \tag{2.12}$$

respectively.

Both US101 Data and I80 Data consist of more than five lanes. Because of that, in order to make them comparable to our model, which is for a five-lane road, we have decreased the number of lanes by considering cars on lanes, which are right to the 5^{th}, as on 5^{th} lane.

2.2.2 Driver Observation Space

In traffic, drivers cannot observe all the cars on the road but observe the cars around their vicinity. This work assumes that a driver on lane l observes the closest front and rear cars on lanes $l-2, l-1, l+1$ and $l+2$ along with the front car on lane l. Therefore, up to 9 surrounding cars are observable by the driver. Observations are coded as relative positions and velocities. Specifically, a driver on lane l can detect

- Relative position and velocity of the car in front on the same lane (lane l).
- Relative position and velocity of the car in front on the left lane (lane $l+1$).
- Relative position and velocity of the car in rear on the left lane (lane $l+1$).
- Relative position and velocity of the car in front on the right lane (lane $l-1$).
- Relative position and velocity of the car in rear on the right lane (lane $l-1$).
- Relative position and velocity of the car in front on two left lane (lane $l+2$).
- Relative position and velocity of the car in rear on two left lane of (lane $l+2$).
- Relative position and velocity of the car in front on two right lane (lane $l-2$).
- Relative position and velocity of the car in rear on two right lane (lane $l-2$).
- Own lane number (l).

In this work, both continuous and discrete observation spaces are used for modeling, separately. When we use the discrete observation space, we name the method as regular DQN. When we use continuous observation space we name the method c-DQN. We

provide data validation results for both of the cases. It is noted that regular DQN is computationally less expensive, but c-DQN is more accurate than DQN. Below we provide the details of both DQN and c-DQN.

2.2.2.1 DQN

Human driver observations, consisting of relative positions, δx, and velocities, δv, of the surrounding vehicles, are quantized into different sets: Relative positions are binned as *close*, *nominal* and *far*, while the bins used for relative velocities are *stable*, *approaching* and *moving away*. In order to determine the contents of these bins, the raw US101 traffic data [76] is processed. Below, we explain how the relative position and relative velocity bins are determined.

The distribution of distances between vehicles, obtained by processing US101 traffic data, is shown in Fig. 2.5. It is seen that around 50% of the time, the distance between vehicles stays between 11m and 27m. Based on this observation, the driver relative position observations are binned as *close*, if $\delta x < 11$m, *nominal* if $11\text{m} < \delta x < 27\text{m}$, and *far* if $\delta x > 27$m.

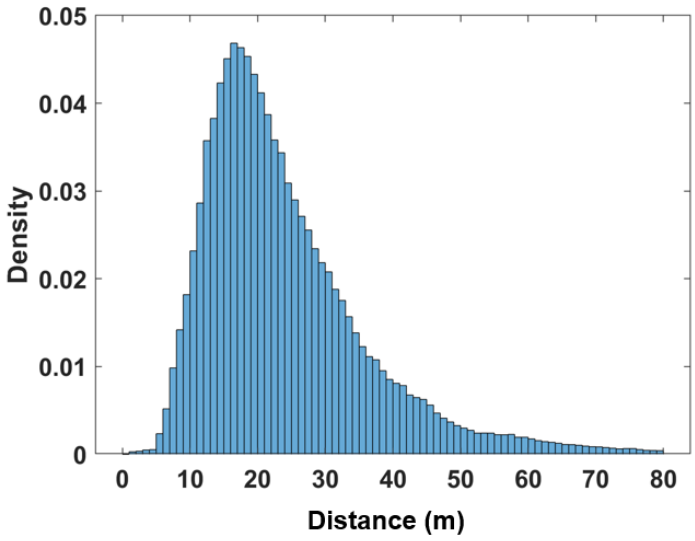


Figure 2.5: Distribution of distances between consecutive vehicles presented as a probability density function.

Drivers also observe the changes in distances to cars around them, which can be thought in terms of relative velocities, δv. In DQN, we bin the relative velocities as approaching if $\delta v < -0.1$m/s, stable if -0.1m/s $< \delta v < 0.1$m/s and moving away if $\delta v > 0.1$m/s.

As a result, the ego driver observation bins for a single surrounding vehicle consist of 3 relative position bins and three relative velocity bins. Furthermore, he/she can observe his/her lane number (1-5). Considering nine surrounding vehicles, the size of the observation space equals $3^9 3^9 5 = 3^{18} 5 = 1937102445$, which approximately equals 2 billion. Hence, even after binning the observations, the resulting observation space size is quite large.

It is noted that in earlier studies [62], [8], [64], [65], at most 5 surrounding vehicles were included in the observation space, which made its size at most 295245. In this work, thanks to the DQN method, which was not employed earlier, a dramatically larger observation space can be handled.

2.2.2.2 c-DQN

The main motivation behind using binned observations, which is the case for regular DQN explained above, is simplifying the learning process without introducing unreasonable assumptions. However, reduced observation resolution may result in inaccurate decisions in critical conditions. To address this problem, we tested the c-DQN approach, where continuous observations are used without any binning. This approach provides an accurate view of the surrounding vehicles to the learning process but makes the observation space infinitely large.

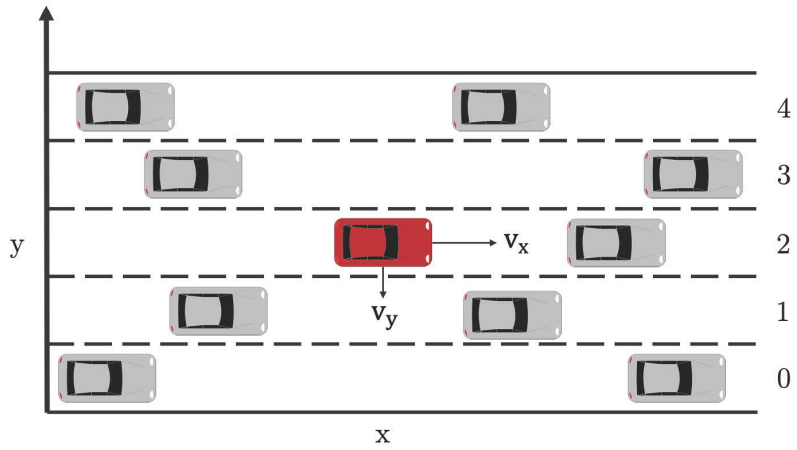


Figure 2.6: The ego vehicle (red, center) and the vehicles the ego driver can observe. Lane numbers are shown on the right.

2.2.3 Driver Action Space

Drivers have two action types: *changing lane* and *changing acceleration*. For lane change, two actions are defined: *moving to the left* lane and *moving to the right* lane. In order to determine *acceleration changing* actions, the distribution of vehicle accelerations, obtained by processing the US101 data, is used. Figure 2.7 presents the acceleration distribution. In the figure, five regions are identified and approximated by known continuous distributions shown in red color and superimposed on the original figure. Based on this acceleration data analysis, the driver actions in terms of accelerations are defined as

1. *Maintain*: acceleration is sampled from normal distribution with $\mu = 0, \sigma = 0.0075\text{m/s}^2$.

2. *Accelerate*: acceleration is sampled from a uniform distribution between $0.5\text{m/s}^2, 2.5\text{m/s}^2$.

3. *Decelerate*: acceleration is sampled from a uniform distribution between $-0.5\text{m/s}^2, -2.5\text{m/s}^2$.

4. *Hard Accelerate*: acceleration is sampled from a inverse half normal distribution with $\mu = 3.5\text{m/s}^2, \sigma = 0.3\text{m/s}^2$.

5. *Hard Decelerate*: acceleration is sampled from a half normal distribution with $\mu = -3.5\text{m/s}^2, \sigma = 0.3\text{m/s}^2$.

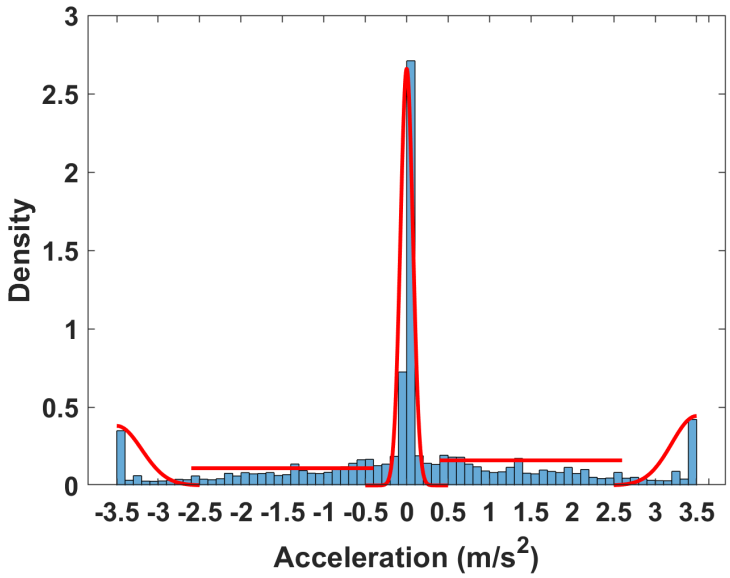


Figure 2.7: Acceleration distribution is approximated with five different distributions: a normal distribution with 0 mean and 0.075 standard deviation, a uniform distribution between 0.5 m/s and 2.5 m/s, a uniform distribution between -0.5 m/s and -2.5 m/s, a half normal distribution with 3.5 mean and 0.3 standard deviation and a half normal distribution with -3.5 mean and 0.3 standard deviation. x-axis shows the accelerations in m/s^2 and y-axis presents the values of the probability density function.

Distributions superimposed on the histogram in Fig. 2.7 are continuous. Therefore, all driver actions are sampled from continuous distributions.

2.2.4 Physical Model of Vehicles

As explained before, drivers can take lane changing or acceleration changing actions. It is assumed that changing the lane takes 1 second.

In Fig. 2.6, the variable x is used to represent the longitudinal position and y represents the lateral position. Similarly, v_x and v_y represent the longitudinal and lateral velocities, respectively. The equations of motion for the vehicles in the traffic are given by

$$x(t_0+t) = x(t_0) + v_x(t_0) + \frac{1}{2}a(t_0)(t-t_0)^2 \tag{2.13}$$

$$y(t_0+t) = y(t_0) + v_y(t_0) \tag{2.14}$$

$$v_x(t_0+t) = v_x(t_0) + a(t_0)(t-t_0), \tag{2.15}$$

where t_0 is the initial time-step, and a is the acceleration.

2.2.5 Vehicle Placements

At the beginning of the training and simulations, vehicles are randomly placed on a 600m endless circular road segment. Initial distances between vehicles are constrained to be larger than, or equal to, 11m, the upper limit of *close*. Initial velocities are selected to prevent impossible-to-handle cases at the beginning of the training or simulation: A driver who is in close proximity to the vehicle in front should be able to prevent a crash using the *hard decelerate* action.

2.2.6 Reward Function

Rewards are collected based on a reward function, which represents the goals of the driver. In traffic, drivers try to avoid crashing and getting too close to other cars. Furthermore, minimizing the travel time with minimal effort is desired. In the reward function, a variable is defined for each of these goals, and the weights are assigned to these variables to emphasize their relative importance. The reward function is defined as

$$R = w_1 c + w_2 s + w_3 d + w_4 e, \tag{2.16}$$

where w_i are the weights. The terms of the reward function are defined below.

c: Equals to -1 if a crash occurs and 0, otherwise. Penalizes the driver if an accident occurs. Getting out of the road boundaries is also considered a crash.

s: Equals to the difference between the speed of the driver and the mean speed normalized by the maximum speed. Thus, higher velocities are rewarded to improve performance. The formula to calculate this term is

$$s = \frac{v(t) - \frac{v_{max}+v_{min}}{2}}{v_{max}}. \tag{2.17}$$

d: Equals to -1 if the distance to the car in front is *close*, 0 if the distance to the car in front is *nominal*, and 1, otherwise. Rewards keeping the headway large and penalizes small headways.

e: Equals to 0 if the action of the driver is *maintain*, -0.25 if the action is *accelerate* or *decelerate*, -0.5 if the action is *hardaccelerate* or *harddecelerate* and -1 if the action is *moveleft* or *moveright*. Penalizes the effort consumed for the action taken.

It is noted that the nominal velocity is selected as 12.29m/s (27.49mph), and the maximum allowed velocity is selected as two times of the nominal velocity, 24.59m/s (55mph), which is the speed limit at US101 for the selected road section. Nominal velocity is not the desired velocity. In fact, as shown in (17), drivers take positive rewards as they approach the maximum velocity and be penalized for velocities smaller than the nominal velocity.

2.3 Training and Simulation

For the training of the driver policies, two separate reinforcement learning (RL) methods, Deep Q-Learning (DQN) and its continuous version, c-DQN, are used together with the level-k reasoning approach. The advantages of these RL methods over each other are discussed in the previous section. The training environment is a five-lane road, where a training episode is defined by a fixed number of simulation steps. When a crash occurs, the existing episode ends a new one is initialized.

During the training of a level-k driver, 125 level-(k-1) vehicles are placed on the road, together with the ego vehicle. This number of vehicles makes the density of the cars approximately equal to the average car density in the US101 data [76]. The number of cars is decreased to 100 at the end of the 1300_{th} episode and increased to 125 again at the end of the 3800_{th} episode, to increase the number of states that the drivers are exposed to during training.

Both DQN and c-DQN algorithms are implemented using the Phyton library Keras [80], together with the stochastic optimizer Adam [81]. The initial learning rate is selected as 0.005, the discount factor γ is selected as 0.975, and the memory capacity N is selected as 2000.

2.3.1 Level-0 Policy

The non-strategic level-0 policy must be determined first before obtaining other levels. A level-0 policy can be defined by using several different approaches. For instance, a uniformly random selection of actions can be defined as level-0 policy [82]. In earlier studies, where approaches similar to the one proposed in this paper, level-0 policies are set as a single persisting action regardless of the state being observed [83], [84], [85], or as a conditional logic based on experience [86]. The level-0 policy used in this study is defined as

1. *hard decelerate* if the car in front is *close* and *approaching*;
2. *decelerate* if the car in front is *close* and *stable* or *nominal* and *approaching*;
3. *accelerate* if the car in front is *nominal* and *movingaway* or *far* and
4. *maintain* otherwise.

2.3.2 Training Performance

Fig. 2.8 and Fig. 2.9, show the evolution of the average rewards during training for DQN and c-DQN methods, respectively. The rewards monotonically increase and eventually converge for both of the methods. c-DQN rewards tend to have a more uniform structure within different levels compared to the regular DQN. The main reason for this behavior is that c-DQN learns and converges faster, and reward curves look more uniform when plotted with the same scale.

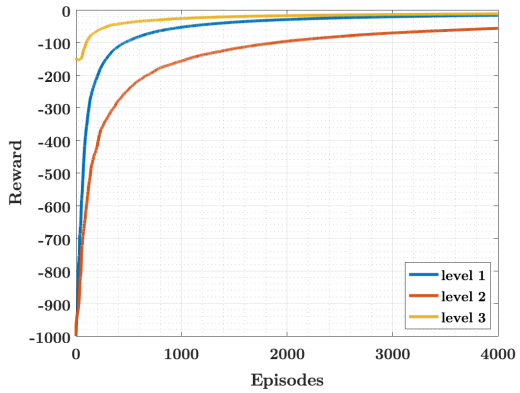


Figure 2.8: Average rewards during level-1, level-2 and level-3 policy training for Deep Q-Learning.

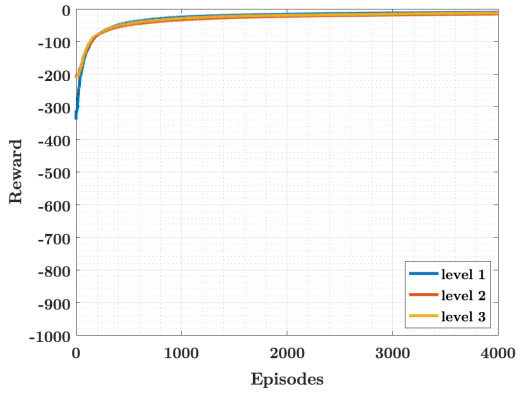


Figure 2.9: Average rewards during level-1, level-2 and level-3 policy training for continuous Deep Q-Learning.

2.3.3 Simulation Performance

The following scenarios are simulated

- Level-1 driver is placed on a traffic environment consisting $n_d - 1$ level-0 drivers on a 600m road segment
- Level-2 driver is placed on a traffic environment consisting $n_d - 1$ level-1 drivers on a 600m road segment
- Level-3 driver is placed on a traffic environment consisting $n_d - 1$ level-2 drivers on a 600m road segment,

where n_d corresponds to the total number of drivers on the road. Simulations are performed for $n_d = 75, 80, 85, 90, 95, 100, 105, 110, 115, 120$ and 125, for each scenario. In all of the above scenarios, simulations are run for 100 episodes, each covering a 100s simulation. Simulation results, in terms of crash rates, for DQN and c-DQN are presented in Fig. 2.10 and 2.11, respectively.

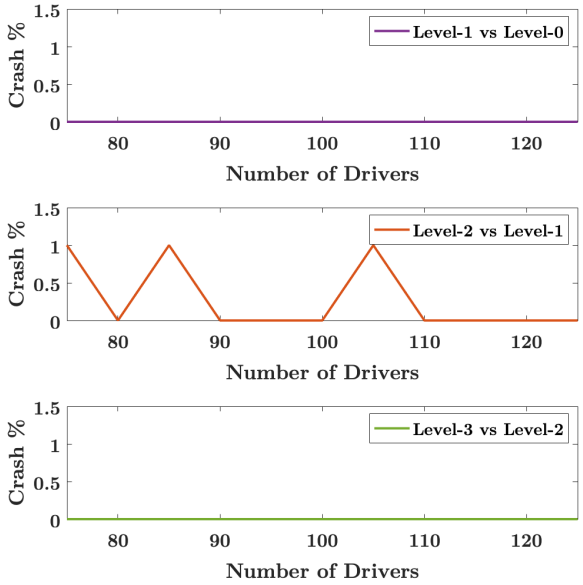


Figure 2.10: Crash rates for level-k vs level-(k-1) scenarios for different number of cars on a circular 600m road for policies trained with DQN.

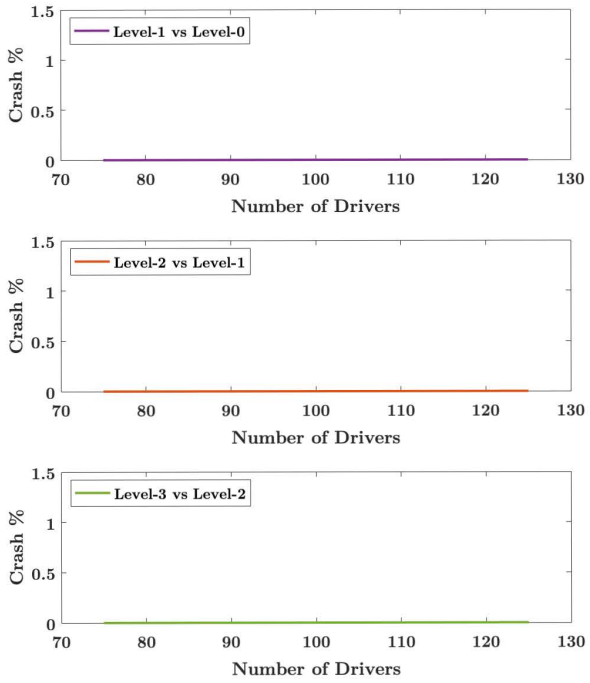


Figure 2.11: Crash rates for level-k vs level-(k-1) scenarios for different number of cars on a circular 600m road for policies trained with c-DQN.

When compared with previous studies, [8] and [63], in terms of crash rates, policies proposed in this paper show more realistic driving behavior, since the average crash rate is 2 per million miles driven nationally [87].

2.4 Validation with Traffic Data

In order to compare the proposed policies, i.e., driver models, with the policies obtained by processing the real traffic data, Kolmogorov Smirnov Goodness of Fit Test (K-S Test) is employed. This test is one of the most commonly used non-parametric goodness of fit tests [53]. Since the policies consist of discrete probability distributions, K-S Test for Discontinuous Distributions [88] is used. The test is explained briefly in the following section, and a more detailed description can be found in [88].

2.4.1 Kolmogorov-Smirnov Test for Discontinious Distributions

For an unknown discrete probability distribution function (pdf) $F(x)$ and a hypothesized pdf $H(x)$, the null hypothesis of the K-S Test is

$$H_0 : F(x) = H(x) \text{ for all x.} \tag{2.18}$$

To test the null hypothesis, first, empirical cumulative pdf of observed data, $S_n(x)$, and hypothesized cumulative pdf, $H_c(x)$, are calculated. Secondly, the test statistics, which are measures of the difference between $S_n(x)$ and $H_c(x)$, are calculated as

$$D = sup_x|H_c(x) - S_n(x)| \tag{2.19}$$

$$D^- = sup_x(H_c(x) - S_n(x)) \tag{2.20}$$

$$D^+ = sup_x(S_n(x) - H_c(x)), \tag{2.21}$$

where D is the two sided, and D^- and D^+ are one sided test statistics. Thirdly, *critical level*s of D^- and D^+, $P(D^- \leq d^-)$ and $P(D^+ \leq d^+)$, are calculated using Algorithm 5-6, respectively, where n denotes the sample size, and d^- and d^+ denote the observed values of D^- and D^+.

Algorithm 5 Calculation of the critical value of D^-

for i = 1 to $n(1-d^-)$ **do**
  On the graph of $H_c(x)$, draw a horizontal line with ordinate $d^- + i/n$
  if this line intersects with the graph of $H_c(x)$ at a jump (a discontinuity), **then**
    Set $c_i = 1 - H_c(x)$. If the intersection occurs exactly at the left limit of the discontinuity, use the left limit value for $H_c(x)$. Otherwise use the right limit.
  else
    Set $c_i = 1 - d^- - i/n$
  end if
end for
Set $b_0 = 1$
for i = 1 to $n(1-d^-)$ **do**
  if $c_i > 0$ **then**
    Set $b_i = 1 - \sum_{j=0}^{i-1} C(i,j) c_j^{i-j} b_i$
  end if
end for
Calculate the critical level as:
$P(D^- \geq d^-) = \sum_{i=0}^{n(1-d^-)} C(n,i) c_i^{n-i} b_i$

Algorithm 6 Calculation of the critical value of D^+

for i = 1 to $n(1-d^+)$ **do**
  On the graph of $H_c(x)$, draw a horizontal line with ordinate $1-d^+-i/n$
  if This line intersects with the graph of $H_c(x)$ at a jump (a discontinuity), **then**
    Set $c_i = 1-H_c(x)$. If the intersection occurs exactly at the right limit of the discontinuity, use the right limit value for $H_c(x)$. Otherwise use the left limit.
  else
    Set $f_i = 1-d^+-i/n$
  end if
end for
Set $e_0 = 1$
for i = 1 to $n(1-d^+)$ **do**
  if $f_i > 0$ **then**
    Set $e_i = 1-\sum_{j=0}^{i-1} C(i,j) f_j^{i-j} e_i$
  end if
end for
Calculate the critical level as:
$P(D^+ \geq d^+) = \sum_{i=0}^{n(1-d^+)} C(n,i) f_i^{n-i} e_i$

Finally, the critical value for the two-sided test statistic is determined as

$$P(D \geq d) \doteq P(D^+ \geq d) + P(D^- \geq d), \tag{2.22}$$

where d is the observed value of D. It is noted that this critical value describes the percentage of data samples, whose test statistics are larger than or equal to d, given that the null hypothesis is true. Thus, the probability of observation (data point) being sampled from the hypothesized model, $H(x)$, or, equivalently, the probability of the null hypothesis being true, increases with the increase in the critical value. The null hypothesis is rejected if the critical value is smaller than a certain threshold called the *significance value* α, which is selected as 0.05 and 0.10 in this work.

Remark 1 *When the null hypothesis can not be rejected, it means that there is not enough data-based evidence that the investigated model is not representative of the*

real data. This leads to retaining the null hypothesis. Therefore, in the rest of the paper, we call a data-modal comparison "successful" when the null hypothesis is not rejected.

2.4.2 Comparing game theoretical models with traffic data

Obtaining game-theoretical (GT) policies, which are stochastic maps from observations to actions, is explained in previous sections. To obtain the data-based driver policies, the traffic data, [77] and [76], are processed, and for each vehicle, action probability distributions over all visited states are generated. The probabilities are calculated by the frequencies of the actions taken by the drivers for a given state. Action probabilities that are lower than 0.01 are replaced with 0.01 with re-normalizations in order to eliminate close-to-zero probabilities for both the GT policies and the ones obtained from the data.

GT and the data-based policies are compared for each driver: For a given vehicle, whose states and actions are captured in the traffic data, first, the vehicle driver's frequency of actions for each visited state is calculated. These frequencies are converted to probability distributions over actions for each state. The probability distributions are called the policies of the driver. Once the driver policies are determined from the data, these policies are compared with the GT policies, using the K-S test for each state. Finally, *success rate* of the GT policies, for the individual driver being investigated, is defined as the ratio of the states whose corresponding policies are successfully modeled by the GT policies over all the visited states. For example, the result may state that "70% of the states visited by Driver-1 are successfully modeled by the GT policies, therefore the success rate is 70%". The process of comparison for each driver is given in Algorithm 7, where $nstate$ defines the number of states that are visited by the driver, $n^i_{visit-driver}$ defines the number of times the state i is visited by the driver. Furthermore, $n_{comparisons}$ and $n_{success}$ are the total number of states whose policies are compared and the number of successful comparisons, respectively. Since the K-S test works best for large sample sizes, the comparisons are conducted for states that are visited more than a certain number, which we call n_{limit}.

Algorithm 7 Comparing GT models with traffic data, for an individual driver

for i = 1 to $nstate$ **do**
  if $n^i_{visit-driver} \geq n_{limit}$ **then**
    $n_{comparisons} += 1$
    Set p_i to the GT policy (pdf).
    Set k_i to the data-based policy (pdf).
    Set H_c to the cumulative pdf obtained from p_i.
    Set S_n to the cumulative pdf obtained from k_i.
    Test the null hypothesis (18) using K-S test.
    if Null hypothesis is not rejected **then**
      $n_{success} += 1$
    end if
  end if
end for
Percentage of successfully modeled states (for this specific driver) = $100\frac{n_{success}}{n_{comparisons}}$

Data-based policies of each individual driver are compared with the proposed GT policies using Algorithm 7 and the *success rate*s for each driver are found and plotted.

It is noted that the proposed GT models are pdfs over the action space defined in Chapter III. To compare the performance of these models with an alternative model, the alternative model should have the same stochastic map structure where the pdfs are given over the same action space. If this requirement is not satisfied, i.e., the alternative model is not stochastic or does not have the same action space, it becomes unclear how to conduct a systematic comparison. Furthermore, in the case that the domain of the stochastic map (action space) is not the same, then the traffic data needs to be reprocessed to obtain policies that have the same structure of the alternative model, which is a nontrivial task. A commonly used method in these circumstances is creating a benchmark model, which is used a minimum performance threshold for the tested model. In this study, we used a model that has uniform pdf over the action space as the benchmark, and the *success rate*s (see Algorithm 7) of the benchmark model are also provided for comparison purposes.

2.4.3 Results

For the validation of the proposed game theoretical (GT) driver models, two different sets of traffic data, obtained from the highways US101 [76], and I80 [77], are used.

The following definitions are employed when reporting the validation results.

Definition 1 *Given two discrete probability distribution functions (pdf) p and q, the Mean Absolute Error (MAE) between p and q is defined as*

$$MAE = 1/n \sum_{i=1}^{n} |p(x_i) - q(x_i)|, \tag{2.23}$$

where x_is are random variables.

Definition 2 *aMAE is the average of the MAE_js between the GT policies and the data-based policies, for which the null hypothesis is not rejected. Therefore, aMAE is calculated as*

$$aMAE = 1/M \sum_{j=1}^{m} MAE_j, \tag{2.24}$$

where M is the number of comparisons for which the null hypothesis is not rejected.

Definition 3 *rMAE is the average of the MAE_ks between the GT policies and the data-based policies, for which the null hypothesis is rejected. Therefore, rMAE is calculated as*

$$rMAE = 1/K \sum_{k=1}^{m} MAE_k, \tag{2.25}$$

where K is the number of comparisons for which the null hypothesis is rejected.

To compare the proposed models with existing approaches, which are policies in the previous work [62] (pGT), IDM [66], and MOBIL [67] , first of all, the existing approaches are reformatted as the same stochastic map structure of the proposed models. The rear vehicle on the lane of the ego car is included in the observation space of MOBIL. However, since the observation space utilized in this work does not include

this vehicle, the rear vehicle on the ego lane is omitted. Two different MOBIL policies, $M-0$ and $M-1$, are generated for politeness values (p) 0 and 1, respectively. After generating the policies of MOBIL and IDM, to obtain the stochastic map structure of these policies, 100 random samples are created for each state, and action probability distributions of IDM and MOBIL are generated.

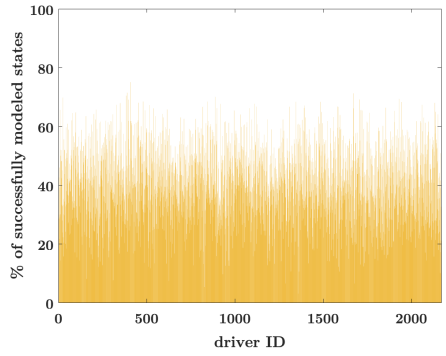
Model vs data comparisons are made for two different n_{limit} values, specifically for $n_{limit} = 3$ and $n_{limit} = 5$. As explained earlier, n_{limit} is the minimum number of state visits in the traffic data for the corresponding policy to be considered in the K-S test. When a state is visited a small number of times, the probability of the resulting driver policy for this state being sampled from a Uniform action-probability-Distribution (UD) increases, which means that the policy for the given state does not have a "structure" that is distinguishable from a UD. It is observed that the minimum number of state visits is approximately equal to 3 for the K-S test to acknowledge that the policy is sampled from a non-uniform distribution, with a significance value of 0.05. Therefore we report the results for $n_limit = 3$. Moreover, we also report the results for $n_{limit} = 5$ to demonstrate the effect of this variable on the test outcomes. Finally, we state the RL_{method} in the results, which is the reinforcement learning method, either DQN or c-DQN, used in the tests.

2.4.3.1 Model validation using US-101 Data

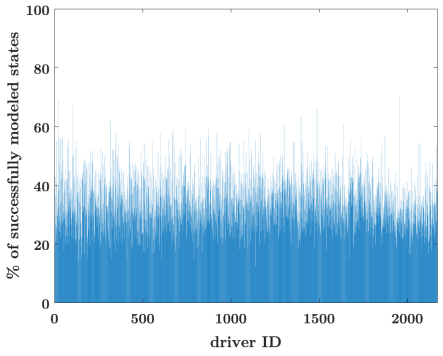
In this section, we give comparison results between the policies obtained by processing the raw US-101 Data and the GT policies. The data are collected between 7.50-8.05 AM, and consists of 2168 different drivers [76].

a)RL_{method} *:DQN,* $n_{limit} = 3$

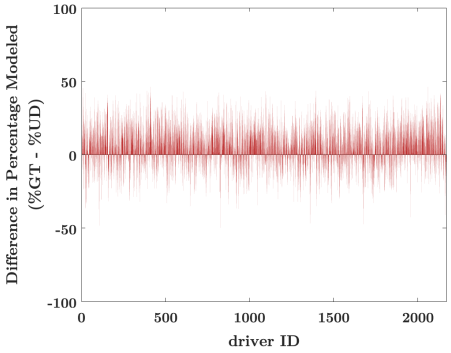
For this model-data comparison, $aMAE = 0.07$ and $rMAE = 0.22$.



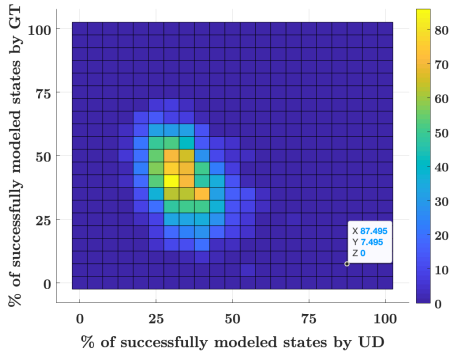
(a) Percentages of successfully modeled states by the GT policies obtained through DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

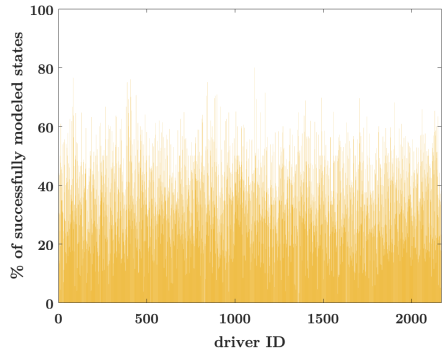
Figure 2.12: Comparison results for $n_{limit} = 3$ and $RL_{method} = DQN$ (US101)

Fig. 2.12a and 2.12b show the performances of the proposed GT policies and the uniform distribution (UD) policy in terms of modeling human driver behaviors. In these figures, the x-axis shows driver IDs, which start from 1 and end at 2168; and the y-axis shows the percentages of the successfully modeled states for each driver. The differences between the percentages of successfully modeled states, for each driver, by

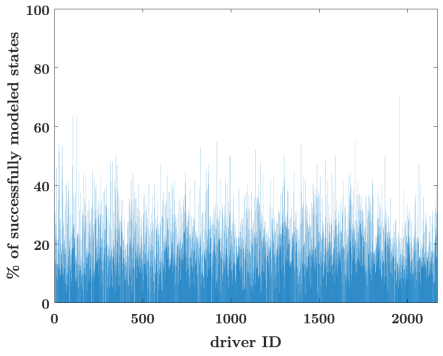
the GT policies and the UD policy are presented in 2.12c. Overall, Figs. 2.12a-c show that the performance (success rate) of the GT policies are better than the UD policy in general. However, the difference is not large. Fig. 2.12d shows the difference between the performances of the GT policies and the UD policy using a different visualization method: In the figure, the x and y axes (horizontal and vertical) show the percentages of the successfully modeled states by the UD and GT policies, respectively. The colors on the figure represent the number of drivers. For example, the figure shows that there are around 75 drivers, whose 50% of the states' policies could be successfully modeled by the GT policy while only 30% could be modeled by the UD policy. The colored cluster being above the x=y line shows that GT performs better than the UD policy, in general.

b)RL_{method} :*DQN,* $n_{limit} = 5$

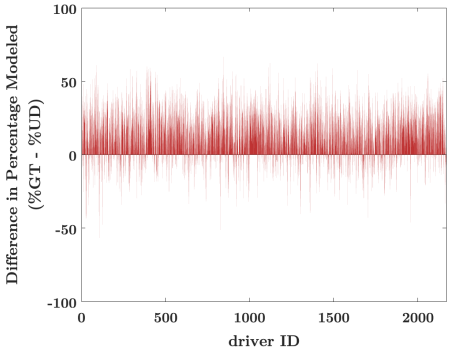
For this model-data comparison, $aMAE = 0.06$ and $rMAE = 0.22$.



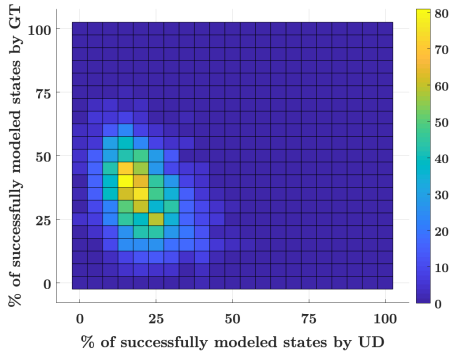
(a) Percentages of successfully modeled states by the GT policies obtained through DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

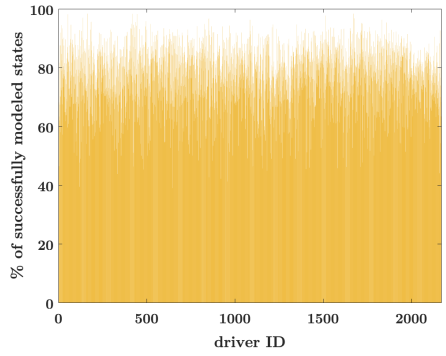
Figure 2.13: Comparison results for $n_{limit} = 5$ and $RL_{method} = DQN$ (US101)

Performance of the GT and UD policies are shown in Figs. 2.13a and 2.13b, respectively. The difference between these policies, in terms of successfully modeled state percentages, is given in Fig 2.13c. When compared with Fig. 2.12c, Fig. 2.13c shows that the positive performance difference between the GT policies and the UD policy increases with the increase in n_{limit} value. The main reason behind this is that with

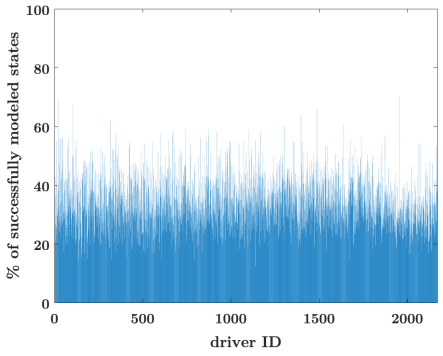
the increase of n_{limit}, the K-S test power increases. Therefore the test can make better decisions in terms of determining whether or not the observed data is sampled from the hypothesized probability distribution function (pdf). The color map created earlier and presented in Fig. 2.12d is also created here but this time for $n_{limit} = 5$, which is given in Fig. 2.13d. Compared to Fig. 2.12d, it is seen that the color cluster's distance from the x=y line is increased, corresponding to increased performance improvement of the GT policies over UD policy.

c) RL_{method} *:c-DQN,* $n_{limit} = 3$

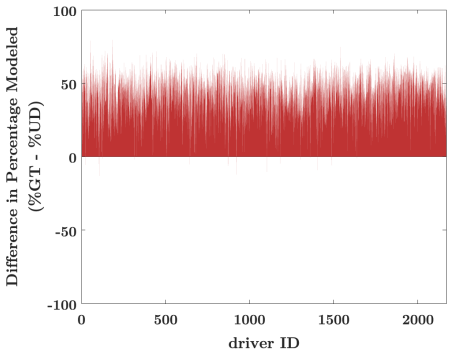
$aMAE = 0.10$ and $rMAE = 0.22$, for this model-data comparison.



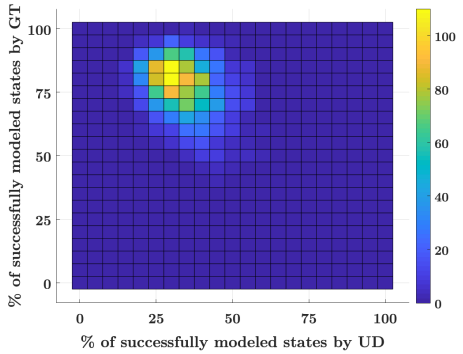
(a) Percentages of successfully modeled states by the GT policies obtained through c-DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the c-DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the c-DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

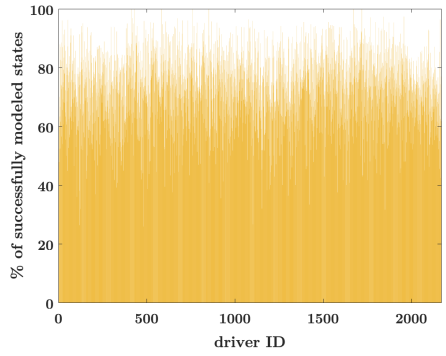
Figure 2.14: Comparison results for $n_{limit} = 3$ and $RL_{method} = c-DQN$ (US101)

Performance of the GT policies, based on c-DQN, in terms of modeling drive behavior, and the difference between the GT and UD policy performances are given in Fig. 2.14a and 2.14c, respectively. The dramatic improvement in modeling percentages compared to the previous two cases, presented in subsections *a)* and *b)*, is a result

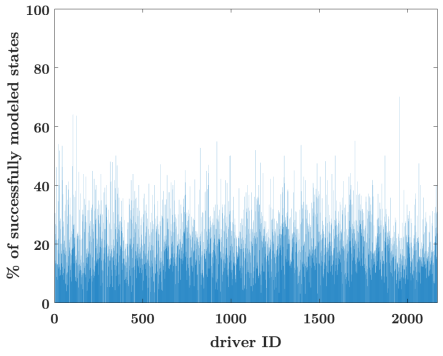
of continuous, instead of discrete, observations. A color map similar to the one presented in Fig. 2.12d, but for the case where c-DQN is employed, instead of DQN, is also created and shown in Fig. 2.14d. Compared to Fig. 2.12d, the color cluster is further away from the x=y line, which also emphasizes the dramatic improvement over the positive performance difference of GT policies over the UD policy.

d) RL_{method} *:c-DQN,* $n_{limit} = 5$

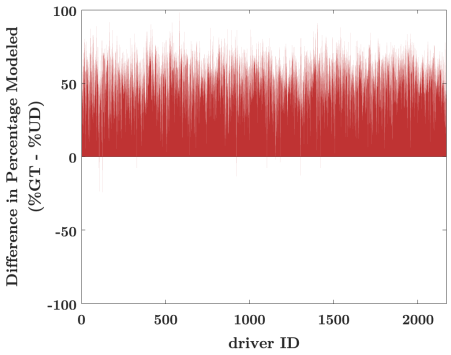
$aMAE = 0.08$ and $rMAE = 0.20$, for this model-data comparison.



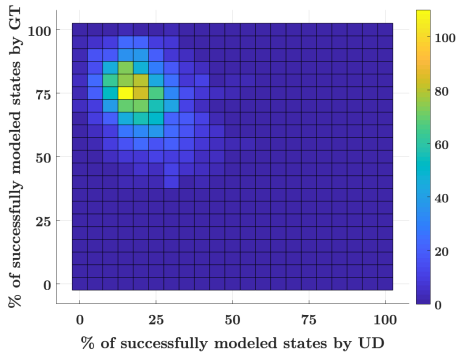
(a) Percentages of successfully modeled states by the GT policies obtained through c-DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the c-DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the c-DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

Figure 2.15: Comparison results for $n_{limit} = 5$ and $RL_{method} = c-DQN$ (US101)

Fig. 2.15a shows the percentages of successfully modeled driver behavior, for each of the 2168 drivers, by the c-DQN-based GT policies. Moreover, Fig. 2.15c shows the difference between the successfully modeled visited state percentages of each driver by the GT policies and the UD policy. Fig. 2.15d presents a color map similar to the one given in Fig. 2.14d, but this time for the case of $n_{limit} = 5$. Compared to the previous

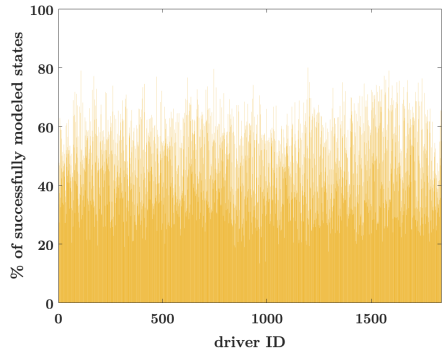
case, presented in subsection *c)*, these figures, Figs. 2.15a-c, show that the positive performance difference between the GT and UD policies improved. The main reason for this improvement is the increase n_{limit} value, which corresponds to an increased K-S test power.

2.4.3.2 Model Validation with I-80 Data

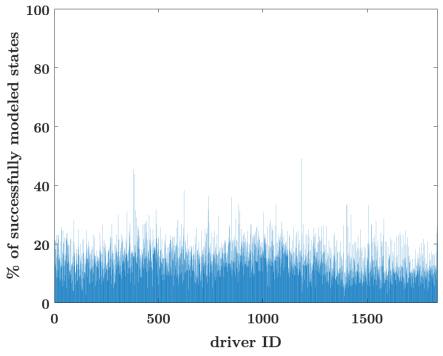
In addition to US101, highway I-80 data [77] is also used to test the validity of the proposed GT policies. For this test, I-80 data collected between 5.00-5.15 PM is used, which contains 1835 drivers.

a)RL_{method} :*DQN,* $n_{limit} = 3$

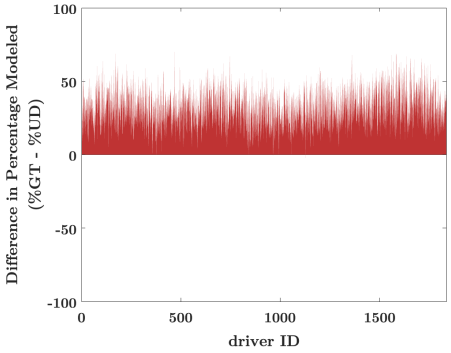
For this model-data comparison, $aMAE = 0.04$ and $rMAE = 0.23$.



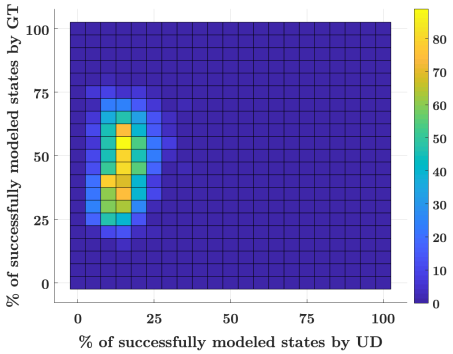
(a) Percentages of successfully modeled states by the GT policies obtained through DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

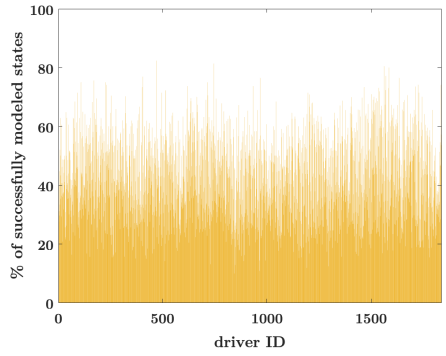
Figure 2.16: Comparison results for $n_{limit} = 3$ and $RL_{method} = DQN$ (I80)

For every 1835 human drivers in the dataset, percentages of visited states whose policies are successfully modeled by GT and UD policies are presented in Figs. 2.16a and 2.16b, respectively. These figures show that the GT policies model human drivers considerably better than the UD policies. The difference between the percentages of the successfully modeled policies by the GT policies and the UD policy is given in

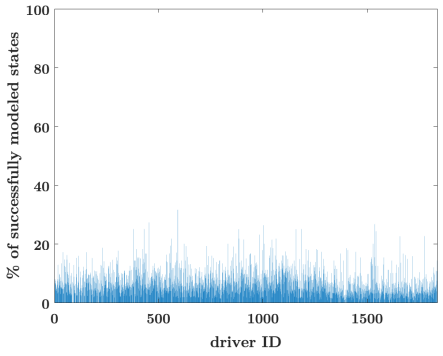
Fig. 2.16c. When compared with Fig. 2.12c, Fig. 2.16c shows that the proposed GT policies have a higher percentage of success rate in modeling drivers in I80 compared to the ones in US101. In Fig. 2.16d x and y axes (horizontal and vertical) show the percentages of the successfully modeled states by the UD and GT policies, respectively. The color cluster in Fig. 2.16d being above the x=y line shows that the GT policies perform better than the UD policy. Compared to Fig. 2.12d, Fig. 2.16d shows that the positive performance difference between the GT and UD policies is larger for I80 data compared to that of US101.

b)RL_{method} :*DQN,* $n_{limit} = 5$

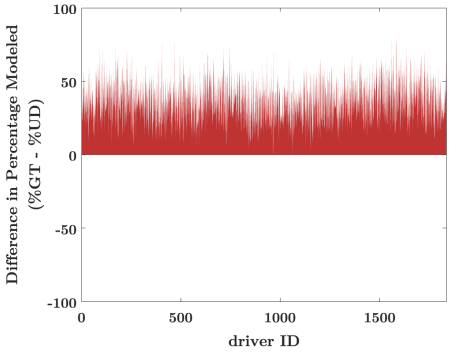
$aMAE = 0.03$ and $rMAE = 0.23$, for this model-data comparison.



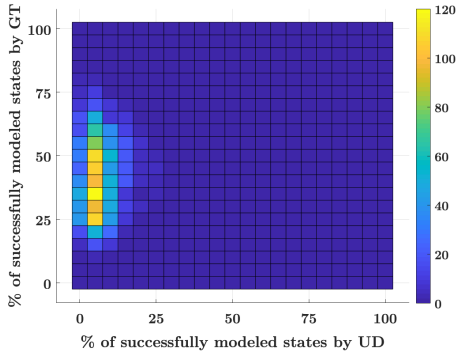
(a) Percentages of successfully modeled states by the GT policies obtained through DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

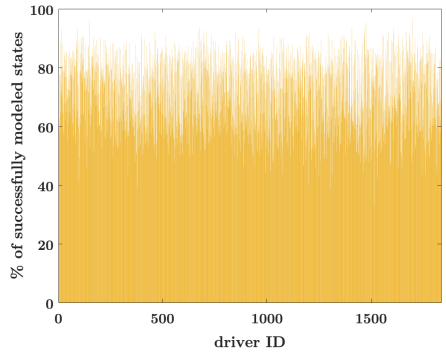
Figure 2.17: Comparison results for $n_{limit} = 5$ and $RL_{method} = DQN$ (I80)

Driver behavior modeling performance of the GT and UD policies are given in Fig. 2.17a and 2.17b, respectively. The difference between these policies, in terms of successfully modeled state percentages, is given in Fig 2.17c. A color map, similar to Fig. 2.13d, is also shown in Fig. 2.17d. Compared to Figs. 2.16a-d, Figs. 2.17a-d show that with the increase in the n_{limit} value, i.e., with the increase in the test power,

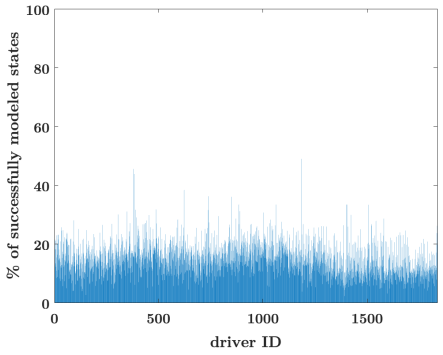
the difference between the GT and the UD policies becomes more clear in terms of human driver behavior modeling performance.

c) RL_{method} *:c-DQN,* $n_{limit} = 3$

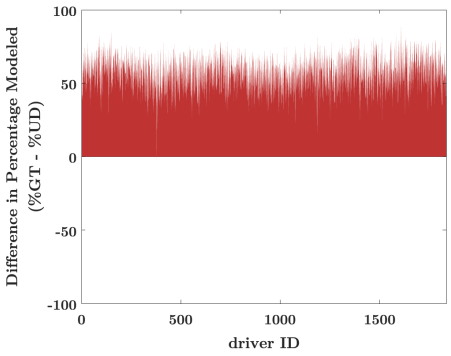
$aMAE = 0.09$ and $rMAE = 0.22$, for this model-data comparison.



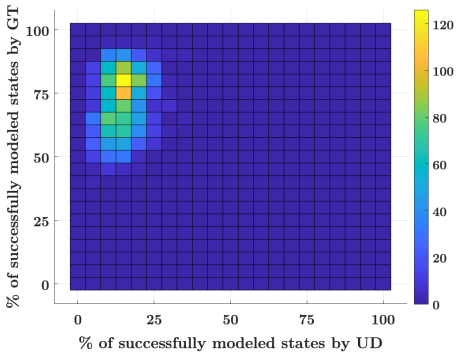
(a) Percentages of successfully modeled states by the GT policies obtained through c-DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the c-DQN-based GT policies and the UD policy.



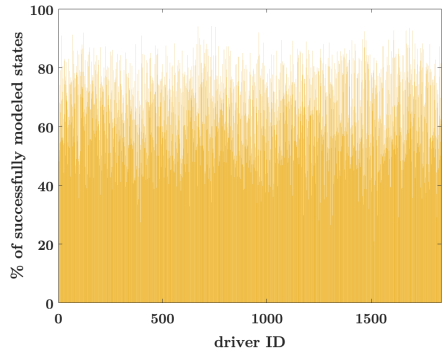
(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the c-DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

Figure 2.18: Comparison results for $n_{limit} = 3$ and $RL_{method} = c - DQN$ (I80)

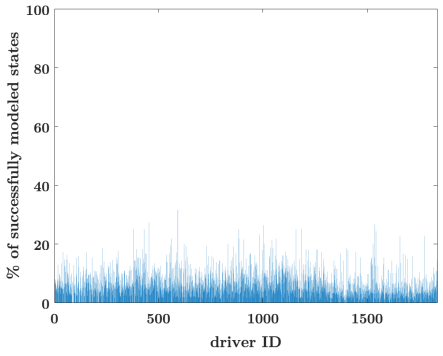
In Fig. 2.18a, the performance of the GT policies in terms of modeling driver performance is shown, and in Fig. 2.18c, the difference between the percentage of states whose policies are successfully modeled by the GT policies and by the UD policy is given. Fig. 2.18d shows a color map similar to the one presented in Fig. 2.16d, but for the case where c-DQN, instead of DQN, is employed. Compared to the previous two cases, the results of which are provided in Figs. 2.16a-d and 2.17a-d, Figs. 2.18a-c show a dramatic improvement in the modeling capability of the GT policies, thanks to the continuous observation space. Furthermore, a comparison between Fig. 2.14a-c and Fig. 2.18a-c shows that proposed GT policies' performance advantage over UD is more pronounced for I80 compared to US101.

d) RL_{method} *:c-DQN,* $n_{limit} = 5$

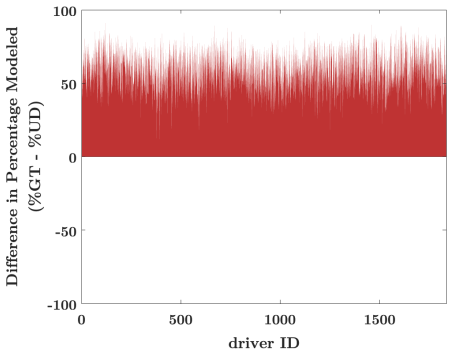
For this model-data comparison, $aMAE = 0.07$ and $rMAE = 0.21$.



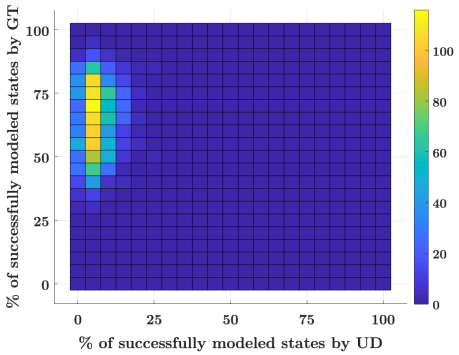
(a) Percentages of successfully modeled states by the GT policies obtained through c-DQN, for each driver. Each vertical line belongs to an individual driver.



(b) Percentages of successfully modeled states by the UD policy, for each driver. Each vertical line belongs to an individual driver..



(c) Differences in the percentages of the successfully modeled states of each driver, between the c-DQN-based GT policies and the UD policy.



(d) Color map showing the number of drivers whose x% of the visited states are successfully modeled by the UD policy and y% by the c-DQN-based GT policy. x and y percentages are given in the horizontal and vertical axes, respectively.

Figure 2.19: Comparison results for $n_{limit} = 5$ and $RL_{method} = c-DQN$ (I80)

Performance of the GT policies in terms of modeling human driver behavior is presented in Fig. 2.19a. Furthermore, for each of the 1835 drivers, the difference between percentages of successfully modeled states by the GT policies and the UD policy is shown in Fig. 2.19c. A color map, similar to the one presented in Fig. 2.18d is given in Fig. 2.19d. Compared to Figs. 2.18a-c, Figs. 2.19a-c show that

the increase in n_{limit}, which increases the K-S test power, demonstrates the positive difference between the modeling capability of the GT and UD policies more clearly.

Remark 2 *US101 data is used only to determine the observation and action set boundaries. It is not used to train the GT driver models. Therefore, the GT policies are not obtained by fitting the model parameters to the data. However, since this data is used to set the observation-action space boundaries, it still affected, albeit indirectly, the obtained models. To test the resulting GT policies with data that is not used in any way to obtain these policies, additional model-validation tests are conducted with the I80 data. To summarize, although the US101 data is not used to train the models, and therefore overfitting is not a concern, additional validation tests are conducted with the I80 data for further assurance of the validity of the GT models.*

2.4.3.3 Summary of Results

Table I and Table II are given for the clear presentation of the statistical analysis results, which provide a summary of each K-S test conducted with different parameters. c-DQN policies are referred to as cGT, and DQN policies are referred to as dGT in the results.

Table I presents the results obtained using US101 data for the proposed GT policies and the existing approaches. Average mean errors for the policies that passed the K-S test, aMAE, and for the ones that failed to pass the test, rMAE, are provided in the table. Results present that the proposed GT policies, cGT, model human behaviors better than the existing approaches. It is noted that in the previous work [62], data filtering is utilized since a traditional RL approach is used. Thus, comparisons are made with a limited portion of the data in [62]. On the other hand, no data filtration is done in this work. Although much larger data is utilized, cGT performs significantly better than pGT policies.

Table 3 presents the performances of the proposed GT policies and baseline models for I80 data. This table also shows that proposed GT policies overperformed baseline methods, IDM and MOBIL, in terms of modeling human drivers. Besides, again, the

Table 2.1: Driver modeling performances of the proposed GT policies and the existing approaches for US101 data.

		n_{state} = 3	n_{state}= 5
α = 0.05	Mean % modeled by cGT policies	**76.73%**	**72.69%**
	Mean % modeled by pGT policies	60.92%	54.34%
	Mean % modeled by dGT policies	40.26%	34.76%
	Mean % modeled by IDM	17.74%	8.76%
	Mean % modeled by M-0	6.09%	2.80%
	Mean % modeled by M-1	1.34%	0.66%
	Mean % difference: %cGT - %pGT	15.81%	18.35%
	Mean % difference: %cGT - %dGT	36.47%	37.93%
	Mean % difference: %cGT - %IDM	58.99%	63.93%
	Mean % difference: %cGT - %M-0	70.64%	69.89%
	Mean % difference: %cGT - %M-1	75.39%	72.03%
	aMAE	0.10	0.08
	rMAE	0.22	0.20
α = 0.10	Mean % modeled by cGT policies	**68.22%**	**64.28%**
	Mean % modeled by pGT policies	52.48%	45.73%
	Mean % modeled by dGT policies	31.88%	30.10%
	Mean % modeled by IDM	11.73%	5.31%
	Mean % modeled by M-0	3.77%	1.23%
	Mean % modeled by M-1	0.89%	0.41%
	Mean % difference: %cGT - %pGT	15.74%	18.55%
	Mean % difference: %cGT - %dGT	36.34%	34.18%
	Mean % difference: %cGT - %IDM	56.49%	58.97%
	Mean % difference: %cGT - %M-0	64.45%	63.05%
	Mean % difference: %cGT - %M-1	67.33%	63.87%
	aMAE	0.09	0.08
	rMAE	0.21	0.19

performances of the proposed policies are significantly better than the policies in the previous work.

Statistical analysis summary for the I80 data is provided similarly in Table 2. Similar conclusions can be drawn from the results shown in Table 2. The proposed GT policies overperformed the existing approaches. The main difference is that the power of the proposed GT policies is more pronounced here compared to the US101 data. The difference does not stem from a dramatic improvement in the success of the GT policies but a large drop in the predictive power of the existing policies for this dataset.

Table 2.2: Driver modeling performances of the proposed GT policies and the existing approaches for I80 data.

		n_{state} = 3	n_{state}= 5
α = 0.05	Mean % modeled by cGT policies	**71.42%**	**64.37%**
	Mean % modeled by pGT policies	35.97%	30.66%
	Mean % modeled by dGT policies	46.43%	41.83%
	Mean % modeled by IDM	4.56%	2.46%
	Mean % modeled by M-0	1.86%	0.96%
	Mean % modeled by M-1	0.15%	0.05%
	Mean % difference: %cGT - %pGT	35.45%	33.71%
	Mean % difference: %cGT - %dGT	24.99%	22.54%
	Mean % difference: %cGT - %IDM	66.86%	61.91%
	Mean % difference: %cGT - %M-0	69.56%	63.41%
	Mean % difference: %cGT - %M-1	71.27%	64.32%
	aMAE	0.09	0.07
	rMAE	0.22	0.21
α = 0.10	Mean % modeled by cGT policies	**63.04%**	**56.37%**
	Mean % modeled by pGT policies	28.73%	24.24%
	Mean % modeled by dGT policies	41.15%	38.70%
	Mean % modeled by IDM	3.31%	1.43%
	Mean % modeled by M-0	1.15%	0.44%
	Mean % modeled by M-1	0.08%	0.03%
	Mean % difference: %cGT - %pGT	34.31%	32.13%
	Mean % difference: %cGT - %dGT	21.89%	17.67%
	Mean % difference: %cGT - %IDM	59.73%	54.94%
	Mean % difference: %cGT - %M-0	61.89%	55.93%
	Mean % difference: %cGT - %M-1	62.96%	56.34%
	aMAE	0.08	0.06
	rMAE	0.21	0.20

One conclusion is that the driver reactions given in I80 dataset are much harder to model than US101.

Chapter 3

GP-k: Learning Model for Time-Extended Human-Human Interactions

A new learning model for time extended human-human interactions is proposed in this work. Game theory is mainly focused on equilibrium concepts, which fails to infer human behavior in some cases, especially when how to reach the equilibrium becomes important. Learning models imitate the human learning process without any equilibrium assumptions. In this work, through a hierarchical reasoning solution concept, equilibrium concepts are combined with Gaussian Processes to predict the learning behavior. As a result, a novel bounded rational learning approach is developed.

3.1 Building Blocks

The building blocks of the GP-k are presented in this section. Level-k reasoning, reinforcement learning, and Gaussian processes are these building blocks. A brief explanation for each is given below.

3.1.1 Level-k Reasoning

Level-k reasoning is a hierarchical solution concept first proposed in [5], [6] and [89]. In *level-k reasoning*, *level-0* agents do not hold any belief about others and behave non-strategically. *Level-1* players best responds based on their assumption that all other players in the environment are *level-0*. At one step higher, in a similar manner to *level-1*, *level-2* agents believes that other players are *level-1* and responds best to this belief. In short, *level-k* responds best based on the assumption that all other agents are *level-(k-1)*. Thus, *level-k* reasoning is a iterated best response approach. More detailed explanations of level-k are presented in [62].

3.1.2 Reinforcement Learning

Reinforcement learning is a learning representation through reward and punishment. In short, agents observe the environment, take action, transition into a new environment setting, and receive a reward. The goal of the agents are to maximize the weighted cumulative reward. More detailed explanation of reinforcement learning can be found in [90] and [62]. In this work, an approximate reinforcement learning approach, deep Q-learning (DQN), presented in [91] is utilized.

3.1.3 Combination of RL with Level-k

Level-k reasoning and reinforcement learning is combined in order to obtain level-k policies. The main idea is to train the ego player's level-k policy using reinforcement learning, while assigning level-(k-1) policies to the other players. A detailed explanation of this approach is given in the previous chapter, and [62]. In this work, instead of discrete level-k policies, i.e. $\{level-0, level-1, \ldots, level-k\}$, continuous level-k policies, , where k is a non-negative real number, are utilized.

3.1.4 Gaussian Process

Gaussian process is a stochastic process, where any finite number of random variables follow a multivariate Gaussian distribution. A function is expressed by a Gaussian process as

$$f(x) \sim GP(m(x), k(x,x')) \tag{3.1}$$

where $m(x)$ is the mean function and $k(x,x')$ is the kernel, i.e. covariance function. Mean function and kernel are defined as

$$m(x) = E[f(x)] \tag{3.2}$$

$$k(x,x') = E[(f(x) - m(x))(f(x') - m(x'))] \tag{3.3}$$

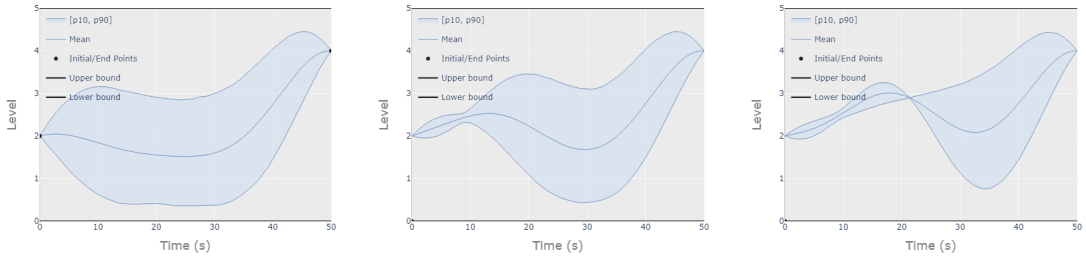
A detailed explanation of Gaussian processes can be found in [92].

3.2 GP-k

In an environment consisting of two players, the learning curve of a player is represented by a function $f(t) : t \to L$, where t is the time, and L is the level space, i.e., consists of all possible levels. It is predicted that a player starts with a policy of a prior level, π_p, and evolves to an end policy, i.e., the reasoning level, π_f, at the converged game at t_f.

For the learning curve of an agent, a Gaussian process is defined as distributions over $f(t)$ such that $GP(m(t), k(t,t'))$, where $m(t)$ is the mean function and $k(t,t')$ is the kernel as defined previously. In the proposed approach, for levels from 0 to the maximum level x, $l_0, \ldots, l_x$, L contains $l_0, l_1, \ldots, l_{x-1}, l_x$. In this work, the kernel is selected as

$$k(t_i, t_j) = \frac{1}{\sigma_f^2} e^{\frac{-1}{l^2}(t_i - t_j)^T (t_i - t_j)} \tag{3.4}$$



(a) Learning curve at the initial game.

(b) Learning curve after 5 observations.

(c) Learning curve after 10 observations.

Figure 3.1: Evaluation of a learning curve of an agent

where σ_f and l are the design parameters.

At the beginning of the game, only two points are known.

$$f(0) = \pi_p \text{ and } f(t_f) = \pi_f \tag{3.5}$$

Thus,

$$T^0 = \begin{bmatrix} t_0 \\ t_f \end{bmatrix}, f^0 = \begin{bmatrix} \pi_p \\ \pi_f \end{bmatrix} \text{ and } K(T,T) = \begin{bmatrix} k(t_0,t_0) & k(t_0,t_f) \\ k(t_f,t_0) & k(t_f,t_f) \end{bmatrix} \tag{3.6}$$

At time-step 1, t_1, the action taken by the agent is observed as a_1. The probability density function of possible levels for this is calculated as $F(l)$, $l \in [l_0, l_x,]$ and the parameters of the closest Gaussian distribution over levels is obtained from optimizing 1-Wasserstein distance as

$$\underset{\alpha_1,\beta_1}{\text{minimize}} \int_L | F(l) - \mathcal{N}(\alpha_1, \beta_1^2) | \, dl \tag{3.7}$$

This observation, $T_*^1 = [t_1]$ and $f_*^1 = [\pi_{\alpha_1}]$ with Gaussian noise $\mathcal{N}(0, \beta_1^2)$, is utilized to update the learning curve by using the relations

$$\begin{bmatrix} f^0 \\ f_*^1 \end{bmatrix} = N\left(0 \quad \begin{bmatrix} K(T^0,T^0) & K(T^0,T_*^1) \\ K(T_*^1,T^0) & K(T_*^1,T_*^1) \end{bmatrix}\right) \tag{3.8}$$

$$K(T_*^1,T^0) = K(T^0,T_*^1)^T = \begin{bmatrix} k(t_1,t_0) \\ k(t_1,t_f) \end{bmatrix}. \tag{3.9}$$

From this, the curve is updated as

$$\mu_*^1 = K(T_*^1,T^0)K(T^0,T^0)f^0 \tag{3.10}$$

$$\Sigma_*^1 = K(T_*^1,T_*^1) - K(T_*^1,T^0)K(T^0,T^0)^{-1}K(T^0,T_*^1) \tag{3.11}$$

$$T^1 = \begin{bmatrix} t_0 \\ t_1 \\ t_f \end{bmatrix} \tag{3.12}$$

$$f^1 = \begin{bmatrix} \pi_0 \\ \pi_{\alpha_1} \\ \pi_f \end{bmatrix} \tag{3.13}$$

Similarly, at time-step i, initially, it is known that

$$T^{i-1} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{i-1} \\ t_f \end{bmatrix}, f^{i-1} = \begin{bmatrix} \pi_p \\ \pi_{\alpha_1} \\ \vdots \\ \pi_{\alpha_{i-1}} \\ \pi_f \end{bmatrix} \tag{3.14}$$

$$K(T^{i-1},T^{i-1}) = \begin{bmatrix} k(t_0,t_0) & k(t_0,t_1) & \cdots & k(t_0,t_{i-1}) & k(t_0,t_f) \\ k(t_1,t_0) & k(t_1,t_1)+\beta_1^2 & \cdots & k(t_1,t_{i-1}) & k(t_1,t_f) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k(t_{i-1},t_0) & k(t_{i-1},t_1) & \cdots & k(t_{i-1},t_{i-1})+\beta_{i-1}^2 & k(t_{i-1},t_f) \\ k(t_f,t_0) & k(t_f,t_1) & \cdots & k(t_f,t_{i-1}) & k(t_f,t_f) \end{bmatrix} \tag{3.15}$$

The probability density function of levels corresponding to the observed action is estimated as $F_2(l)$, $l \in [l_0, l_x]$, and the closest Gaussian distribution parameters are found as

$$\underset{\alpha_i,\beta_i}{\text{minimize}} \int_L \mid F_i(l) - \mathcal{N}(\alpha_i, \beta_i^2) \mid dl \tag{3.16}$$

From this observation, $T_*^i = \left[t_i\right]$ and $f_*^i = \left[\pi_{\alpha_i}\right]$ with Gaussian noise $\mathcal{N}(0, \beta_i^2)$, the learning curve is updated by using the relations

$$\begin{bmatrix} f^{i-1} \\ f_*^1 \end{bmatrix} = N\left(0 \quad \begin{bmatrix} K(T^{i-1}, T^{i-1}) & K(T^{i-1}, T_*^i) \\ K(T_*^i, T^{i-1}) & K(T_*^i, T_*^i) \end{bmatrix}\right) \tag{3.17}$$

$$K(T_*^i, T^{i-1}) = K(T^{i-1}, T_*^i)^T = \begin{bmatrix} k(t_i, t_0) \\ k(t_i, t_1) \\ \vdots \\ k(t_i, t_{i-1}) \\ k(t_i, t_f) \end{bmatrix}. \tag{3.18}$$

as

$$\mu_*^i = K(T_*^i, T^{i-1})K(T^{i-1}, T^{i-1})f^{i-1}$$
$$\Sigma_*^i = K(T_*^i, T_*^i) - K(T_*^i, T^{i-1})K(T^{i-1}, T^{i-1})^{-1}K(T^{i-1}, T_*^i)$$
$$T^i = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_i \\ t_f \end{bmatrix}$$
$$f^1 = \begin{bmatrix} \pi_0 \\ \pi_{o_1} \\ \vdots \\ \pi_{o_i} \\ \pi_f \end{bmatrix}.$$

This process shapes the learning curve as time passes and continues until t_f, where $\pi_{\alpha_f} = \pi_f$.

In this work, the constrained Gaussian processes approach presented in [93] is utilized in order to set an upper and lower limit for the reasoning levels.

The proposed approach is explained visually in Fig. 1-a, 1-b, and 1-c. Although these are the design parameters, σ_f is selected as $x/2$ to cover all the level range, and l is selected as 10. The learning curve of an agent at the initial game is presented in Fig. 1-a. After five observations, the evolved learning curve is given in Fig. 1-b. After five more observations, the obtained learning curve is given in Fig. 1-c.

3.3 Applications

3.3.1 Optimal Behavior via Level Inference

At the beginning, for each l_i, $i \in Z^*$, learning curves are formed and trained. For instance, GP_i presents the distribution of learning curve functions if the initial assumption on the level of the opponent is i. A level-k agent utilizes the GP_{k-1} for the predictions about the opponent.

When a level-k agent is placed in an environment, first of all, from the first two action observations of the opponent a_0, a_1, the agent predicts the corresponding observed level set such that

$$(o_0, o_1) = (\arg\max_i(p(a_0|\pi_i), \arg\max_j(p(a_1|\pi_j)), \quad \text{such that } \pi_i, \pi_j \in \Pi \tag{3.19}$$

where Π is the policy set contains all possible policies corresponding to all levels, i.e., $\pi_0, \ldots, \pi_x$. In addition, the second possible set is obtained as

$$(o_0^{'}, o_1^{'}) = (\arg\max_i(p(a_0|\pi_i), \arg\max_j(p(a_1|\pi_j)), \quad \text{such that} \pi_i\pi_j \in \{\Pi^{'} - \pi_{o_0} - \pi_{o_1}\}. \tag{3.20}$$

Then, for (o_0, o_1) corresponding time sets are calculated as

$$(t_0, t_1) = (\arg\min_i(K(i,T)K(T,T)f - o_0), \arg\min_j(K(j,T)K(T,T)f - o_1). \tag{3.21}$$

Similarly, time sets $(t_0^{'}, t_1)$, $(t_0, t_1^{'})$ and $(t_0^{'}, t_1^{'})$ are calculated for each combined set, $(o_0^{'}, o_1)$, $(o_0, o_1^{'})$ and $(o_0^{'}, o_1^{'})$. Then, the most possible level observation is assumed to

be the one for which the difference between the predicted time-steps are minimum. In other words

$$(t_0^*, t_1^*) = \arg\min_i d(i) \quad \text{such that} i \in H. \tag{3.22}$$

where $H : \{(t_0, t_1), (t_0^{'}, t_1), (t_0, t_1^{'}), (t_0^{'}, t_1^{'})\}$ and $d(u) = u_1 - u_0$.

As a result, the level of the opponent in each future time t_i may be calculated as the mean of the Gaussian process at $t_0^* + t_i(t_1^* - t_0^*)$ as

$$l_{t_i} = K(t_0^* + t_i(t_1^* - t_0^*), T)K(T,T)f + \frac{x}{2}. \tag{3.23}$$

3.4 Sensitivity Analysis

In order to show the effects of the model parameters on the performance of the model, a sensitivity analysis is conducted. The analyzed parameter is the variance of the kernel, σ_f. For three different values of σ_f, 0.5, 1, and 1.5, the modeling performance of the learning model is analyzed.

In order to analyze the modeling performance, two merging scenarios are extracted from reconstructed I-80 data [77]. In these cases, the level of the non-merging car is predicted. First of all, in order to obtain the predicted equilibrium point, two agents are trained in a merging scenario following a multi-agent learning approach, friend Q-learning [94]. The training environment is the same as the one explained in [95]. At the end of the training, two policies are obtained: merging car policy, $\pi_{merging}^{FoF}$, and non-merging car policy, π_{normal}^{FoF}. The level of the non-merging car policy is estimated as

$$l = \arg\max_i KS(\pi_{normal}^{FoF}, \pi_i) \tag{3.24}$$

where $KS(\pi_j, \pi_k)$ is the function that calculates the mean critical value of the K-S test explained in Chapter 2 over all states. The initial point is also found from the first

observation following the Eqn. 3.7. Then for each time-step, the action of the driver is predicted. For this, first, the level is predicted as explained in Chapter 3.2, and the action in that state is taken as the most probable action of the predicted level at that state. After the prediction, the driver's action is observed for the next prediction, and GP-k is updated. This process is followed until the end of the scenario. After predicting actions in each step, the predicted trajectory is generated as the resulting trajectory if the driver starts from his/her initial point and follows the predicted actions.

The mean percentage of the correctly predicted actions and the mean absolute trajectory deviations are calculated for each variable set to evaluate the performance. The results are presented in Figure 3.2. With the increase in the variance of the kernel, the modeling performance increases, and the trajectory deviation decreases. The main reason behind this is that higher variance allows more model elasticity and better fit to the observations.

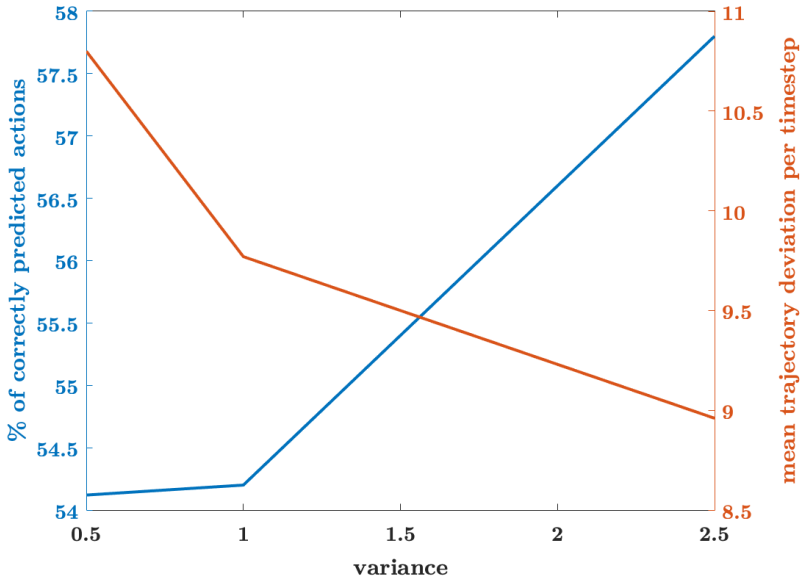


Figure 3.2: Performance of the learning model with varying variance parameter.

Chapter 4

Conclusion

This thesis proposes a stochastic modeling framework for modeling human driver behaviors and a learning model for modeling the time extended human-human interactions. The modeling framework is built by combining a static iterated reasoning approach, level-k reasoning, and a deep reinforcement learning method, DQN. The proposed approach covers a dramatically larger class of scenarios compared to similar approaches in the literature. Proposed driver models are validated with real data, and the modeling performance is compared with the existing approaches. It is presented that the proposed policies perform significantly better than the existing approaches in terms of modeling human behavior.

Secondly, a dynamic learning model for time-extended human-human interactions is proposed by combining reinforcement learning, level-k reasoning, and Gaussian processes. The proposed approach offers a novel bounded rational learning model. A sensitivity analysis of the proposed approach is presented in order to show the effects of model parameters on the modeling performance.

Bibliography

[1] D. Fudenberg, F. Drew, D. K. Levine, and D. K. Levine, *The theory of learning in games*, vol. 2. MIT press, 1998.

[2] D. Fudenberg and D. K. Levine, "Learning and equilibrium," *Annu. Rev. Econ.*, vol. 1, no. 1, pp. 385–420, 2009.

[3] J. F. Nash *et al.*, "Equilibrium points in n-person games," *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[4] S. N. Durlauf and L. E. Blume, "Learning and evolution in games: Belief learning," in *Game Theory*, pp. 29–37, Springer, 2010.

[5] D. O. Stahl II and P. W. Wilson, "Experimental evidence on players' models of other players," *Journal of economic behavior & organization*, vol. 25, no. 3, pp. 309–327, 1994.

[6] R. Nagel, "Unraveling in guessing games: An experimental study," *The American Economic Review*, vol. 85, no. 5, pp. 1313–1326, 1995.

[7] C. F. Camerer, T.-H. Ho, and J.-K. Chong, "A cognitive hierarchy model of games," *The Quarterly Journal of Economics*, vol. 119, no. 3, pp. 861–898, 2004.

[8] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on control systems technology*, 2017.

[9] T. Wongpiromsarn, S. Mitra, R. M. Murray, and A. Lamperski, "Periodically controlled hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 396–410, Springer, 2009.

[10] J. Lygeros, D. N. Godbole, and S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE transactions on automatic control*, vol. 43, no. 4, pp. 522–539, 1998.

[11] T. Wongpiromsarn and R. M. Murray, "Formal verification of an autonomous vehicle system," in *Conference on Decision and Control*, 2008.

[12] A. Liu and D. Salvucci, "Modeling and prediction of human driver behavior," in *Intl. Conference on HCI*, 2001.

[13] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5-6, pp. 351–371, 2002.

[14] I. Dagli, M. Brost, and G. Breuel, "Action recognition and prediction for driver assistance systems using dynamic belief networks," in *Net. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*, pp. 179–194, Springer, 2002.

[15] A. Y. Ungoren and H. Peng, "An adaptive lateral preview driver model," *Vehicle system dynamics*, vol. 43, no. 4, pp. 245–259, 2005.

[16] D. D. Salvucci, "Modeling driver behavior in a cognitive architecture," *Human factors*, vol. 48, no. 2, pp. 362–380, 2006.

[17] Y. Liu and U. Ozguner, "Human driver model and driver decision making for intersection driving," in *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 642–647, IEEE, 2007.

[18] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 797–802, IEEE, 2013.

[19] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[20] Q. Tran and J. Firl, "Modelling of traffic situations at urban intersections with probabilistic non-parametric regression," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 334–339, IEEE, 2013.

[21] V. Gadepally, A. Krishnamurthy, and U. Ozguner, "A framework for estimating driver decisions near intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 637–646, 2014.

[22] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 2641–2646, IEEE, 2015.

[23] S. Lefevre, A. Carvalho, and F. Borrelli, "Autonomous car following: A learning-based approach," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 920–926, IEEE, 2015.

[24] A. Burton, T. Parikh, S. Mascarenhas, J. Zhang, J. Voris, N. S. Artan, and W. Li, "Driver identification and authentication with active behavior modeling," in *2016 12th International Conference on Network and Service Management (CNSM)*, pp. 388–393, IEEE, 2016.

[25] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of recurrent neural networks for probabilistic modeling of driver behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1289–1298, 2016.

[26] M. Zhao, D. Kathner, M. Jipp, D. Soffker, and K. Lemmer, "Modeling driver behavior at roundabouts: Results from a field study," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 908–913, IEEE, 2017.

[27] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 204–211, IEEE, 2017.

[28] M. Da Lio, A. Mazzalai, K. Gurney, and A. Saroldi, "Biologically guided driver modeling: The stop behavior of human car drivers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2454–2469, 2018.

[29] J. Han, D. Karbowski, N. Kim, and A. Rousseau, "Human driver modeling based on analytical optimal solutions: Stopping behaviors at the intersections," in *Dynamic Systems and Control Conference*, vol. 59162, p. V003T18A010, American Society of Mechanical Engineers, 2019.

[30] W. Wang, D. Zhao, W. Han, and J. Xi, "A learning-based approach for lane departure warning systems with a personalized driver model," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9145–9157, 2018.

[31] J. Hu and S. Luo, "A car-following driver model capable of retaining naturalistic driving styles," *Journal of Advanced Transportation*, vol. 2020, 2020.

[32] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions.," in *Robotics: Science and Systems*, vol. 2, Ann Arbor, MI, USA, 2016.

[33] K. Li and P. Ioannou, "Modeling of traffic flow of automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 2, pp. 99–113, 2004.

[34] Y. Wang and P. Ioannou, "New model for variable speed limits," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2249, pp. 38–43, 2011.

[35] A. Abadi, T. Rajabioun, P. A. Ioannou, *et al.*, "Traffic flow prediction for road transportation networks with limited traffic data.," *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 2, pp. 653–662, 2015.

[36] C. F. Camerer, *Behavioral game theory: Experiments in strategic interaction*. Princeton university press, 2011.

[37] N. Feltovich, "Reinforcement-based vs. belief-based learning models in experimental asymmetric-information games," *Econometrica*, vol. 68, no. 3, pp. 605–641, 2000.

[38] S. N. Durlauf and L. E. Blume, "Learning and evolution in games: Belief learning," in *Game Theory*, pp. 191–198, Springer, 2010.

[39] A. A. Cournot, *Researches into the Mathematical Principles of the Theory of Wealth*. Macmillan, 1897.

[40] G. W. Brown, "Iterative solution of games by fictitious play," *Activity analysis of production and allocation*, vol. 13, no. 1, pp. 374–376, 1951.

[41] J. Robinson, "An iterative method of solving a game," *Annals of mathematics*, pp. 296–301, 1951.

[42] J. Nachbar, *Learning in Games*, pp. 1695–1705. New York, NY: Springer New York, 2012.

[43] N. Feltovich, *Belief-Based Learning Models*, pp. 444–447. Boston, MA: Springer US, 2012.

[44] D. Fudenberg and D. M. Kreps, "Learning mixed equilibria," *Games and economic behavior*, vol. 5, no. 3, pp. 320–367, 1993.

[45] J. Jordan, *Bayesian Learning in Games*. American Cancer Society, 2006.

[46] C. Camerer and T. Hua Ho, "Experience-weighted attraction learning in normal form games," *Econometrica*, vol. 67, no. 4, pp. 827–874, 1999.

[47] Y.-W. Cheung and D. Friedman, "Individual learning in normal form games: Some laboratory results," *Games and economic behavior*, vol. 19, no. 1, pp. 46–76, 1997.

[48] V. P. Crawford, "Adaptive dynamics in coordination games," *Econometrica: Journal of the Econometric Society*, pp. 103–143, 1995.

[49] J. C. Cox, J. Shachat, and M. Walker, "An experiment to evaluate bayesian learning of nash equilibrium play," *Games and Economic Behavior*, vol. 34, no. 1, pp. 11–33, 2001.

[50] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[51] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

[52] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[53] H. De Weerd, R. Verbrugge, and B. Verheij, "How much does it help to know what she knows you know? an agent-based simulation study," *Artificial Intelligence*, vol. 199, pp. 67–92, 2013.

[54] F. B. Von Der Osten, M. Kirley, and T. Miller, "The minds of many: Opponent modeling in a stochastic game.," in *IJCAI*, pp. 3845–3851, 2017.

[55] T. Yang, J. Hao, Z. Meng, C. Zhang, Y. Zheng, and Z. Zheng, "Towards efficient detection and optimal response against sophisticated opponents," in *IJCAI*, 2019.

[56] J. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling, "Bayesian action decoder for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, pp. 1942–1951, PMLR, 2019.

[57] P. J. Gmytrasiewicz and P. Doshi, "A framework for sequential planning in multi-agent settings," *Journal of Artificial Intelligence Research*, vol. 24, pp. 49–79, 2005.

[58] Y. Han and P. Gmytrasiewicz, "Learning others' intentional models in multi-agent settings using interactive pomdps," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5639–5647, 2018.

[59] S. Qi and S.-C. Zhu, "Intent-aware multi-agent reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7533–7540, IEEE, 2018.

[60] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *arXiv preprint arXiv:1707.09183*, 2017.

[61] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[62] B. M. Albaba and Y. Yildiz, "Modeling cyber-physical human systems via an interplay between reinforcement learning and game theory," *Annual Reviews in Control*, vol. 48, pp. 1–21, 2019.

[63] M. Albaba, Y. Yildiz, N. Li, I. Kolmanovsky, and A. Girard, "Stochastic driver modeling and validation with traffic data," in *2019 American Control Conference (ACC)*, pp. 4198–4203, IEEE, 2019.

[64] N. Li, D. Oyler, M. Zhang, Y. Yildiz, A. Girard, and I. Kolmanovsky, "Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 727–733, IEEE, 2016.

[65] D. W. Oyler, Y. Yildiz, A. R. Girard, N. I. Li, and I. V. Kolmanovsky, "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," in *Proceedings of the American Control Conference*, vol. 2016, pp. 1705–1710, Institute of Electrical and Electronics Engineers Inc., 2016.

[66] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[67] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[68] C. O. Yaldiz and Y. Yildiz, "Modeling human driver interactions using an infinite policy space through gaussian processes," *arXiv preprint*, 2021.

[69] D. O. Stahl and P. W. Wilson, "On players? models of other players: Theory and experimental evidence," *Games and Economic Behavior*, vol. 10, no. 1, pp. 218–254, 1995.

[70] M. A. Costa-Gomes, V. P. Crawford, and N. Iriberri, "Comparing models of strategic thinking in van huyck, battalio, and beil's coordination games," *Journal of the European Economic Association*, vol. 7, no. 2-3, pp. 365–376, 2009.

[71] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[72] M. Roderick, J. MacGlashan, and S. Tellex, "Implementing the deep q-network," *arXiv preprint arXiv:1711.07478*, 2017.

[73] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[74] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[75] S. Ravichandiran, *Hands-on Reinforcement Learning with Python: Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow*. Packt Publishing Ltd, 2018.

[76] U. F. H. Administration, "Us101 dataset."

[77] U. F. H. Administration, "I80 dataset."

[78] T. Sauer, *Numerical Analysis*. Pearson, 3 ed., 2017.

[79] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, vol. 55. US Government printing office, 1948.

[80] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

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

[82] D. Shapiro, X. Shi, and A. Zillante, "Level-k reasoning in a generalized beauty contest," *Games and Economic Behavior*, vol. 86, pp. 308–329, 2014.

[83] N. Musavi, D. Onural, K. Gunes, and Y. Yildiz, "Unmanned aircraft systems airspace integration: A game theoretical framework for concept evaluations," *Journal of Guidance, Control, and Dynamics*, pp. 96–109, 2016.

[84] N. Musavi, K. B. Tekelioğlu, Y. Yildiz, K. Gunes, and D. Onural, "A game theoretical modeling and simulation framework for the integration of unmanned aircraft systems in to the national airspace," in *AIAA Infotech@ Aerospace*, p. 1001, 2016.

[85] Y. Yildiz, A. Agogino, and G. Brat, "Predicting pilot behavior in medium scale scenarios using game theory and reinforcement learning," in *AIAA Modeling and Simulation Technologies (MST) Conference*, p. 4908, 2013.

[86] S. Backhaus, R. Bent, J. Bono, R. Lee, B. Tracey, D. Wolpert, D. Xie, and Y. Yildiz, "Cyber-physical security: A game theory model of humans interacting over control systems," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2320–2327, 2013.

[87] N. H. T. S. Administration, "Traffic safety facts 2017: A compilation of motor vehicle crash data," *Retrieved*, 2019.

[88] W. J. Conover, "A kolmogorov goodness-of-fit test for discontinuous distributions," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 591–596, 1972.

[89] D. O. Stahl, "Boundedly rational rule learning in a guessing game," *Games and Economic Behavior*, vol. 16, no. 2, pp. 303–330, 1996.

[90] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[91] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[92] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT press, 2005.

[93] C. Agrell, "Gaussian processes with linear operator inequality constraints," *Journal of Machine Learning Research*, vol. 20, pp. 1–36, 2019.

[94] M. L. Littman, "Friend-or-foe q-learning in general-sum games," in *ICML*, vol. 1, pp. 322–328, 2001.

[95] C. Köprülü and Y. Yıldız, "Act to reason: A dynamic game theoretical model of driving," *arXiv preprint arXiv:2101.05399*, 2021.