Generalized Reinforcement Learning for Automated Driving under Uncertainty

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

Danial Kamran, M.Sc.

geb. in Isfahan

Tag der mündlichen Prüfung: 29. January 2024

Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr. Matthijs Spaan

Foreword

This dissertation is the result of my research as a PhD student at the Institute of Measurement and Control Systems (MRT) at Karlsruhe Institute of Technology (KIT). I would like to express my sincere gratitude to Prof. Stiller for his guidance and mentorship throughout my PhD research projects. Special thanks also go to Martin, Sahin, Florian, and other colleagues at MRT for their helpful ideas, comments, and discussions about my work.

I would also like to express my special gratitude to Prof. Spaan for co-examining my thesis and for the insightful discussions and excellent collaboration with his research group (Thiago, Canmanie, and Qisong) at the Algorithmics Group at Delft University of Technology.

Last but not least, I would like to extend my heartfelt thanks to my wife, my parents, and my uncle Ali for their support during every moment of this journey.

Kurzfassung

Automatisiertes Fahren, auch als autonomes Fahren Technologie bekannt, hat eine tiefgreifende Bedeutung im modernen Verkehrswesen und darüber hinaus. Einer der überzeugendsten Gründe für die Bedeutung des automatisierten Fahrens ist sein Potenzial zur erheblichen Verbesserung der Verkehrssicherheit. Menschliches Versagen ist eine führende Ursache für Unfälle auf der Straße, und autonome Fahrzeuge haben das Potenzial, diese Vorfälle drastisch zu reduzieren, indem sie Faktoren wie abgelenktes Fahren, Müdigkeit und beeinträchtigtes Urteilsvermögen eliminieren. Darüber hinaus können automatisierte Systeme schneller auf unerwartete Situationen reagieren als Menschen, was die Wahrscheinlichkeit von Kollisionen weiter verringert. Eines der Hauptprobleme für vollautomatisiertes Fahren besteht darin, dass die Entscheidungsrichtlinie mit verschiedenen Unsicherheitsquellen umgehen muss, wie Wahrnehmungsfehler, verdeckte Fahrzeuge und unbekannte Absichten der Fahrer mit unterschiedlichen Fahrstilen. Diese Arbeit stellt einen Ansatz auf der Grundlage des Reinforcement Learning vor, um dieses komplexe Problem zu lösen.

Reinforcement Learning (RL) kann komplexe Entscheidungsprobleme lösen, indem es langfristig optimale Richtlinien lernt. Im modellfreien RL wird die nicht vollständige Beobachtbarkeit in der Regel durch die Annäherung des wahren Zustands mit einer Historie aktueller Beobachtungen (k-Markov-Annäherung) behandelt. Dies führt jedoch oft zu Richtlinien, die nicht verallgemeinert werden können auf Umgebungen mit leicht unterschiedlichen Dynamiken aufgrund verschiedener Rauschpegel oder unerwarteter menschlicher Verhaltensweisen mit neuen Beobachtungen. Darüber hinaus wird die Sicherheit in der Regel durch die Bestrafung des RL-Agenten für riskante Situationen behandelt, was entweder zu wenigen Fehlern zur Laufzeit oder zu sehr konservativen Richtlinien führt. Selbst *generalisten* Richtlinien, die mit verschiedenen Rauschpegeln und unterschiedlichem menschlichem Verhalten trainiert werden, sind nicht gegen neue Umgebungen mit neuen Fahrstilen robust.

Um die Verallgemeinerung in Bezug auf Sicherheit, Komfort und Nutzen anzugehen, schlagen wir vor, Verteilungs-RL mit einem proaktiven Sicherheitsverifikationsmodul zu kombinieren, um sichere und risikobewusste RL-Richtlinien bereitzustellen. Dank der proaktiven Sicherheitsschicht, die machbare notfallmäßige Ausweichmanöver basierend auf den schlimmsten Annahmen gewährleistet, werden unsichere Aktionen des RL herausgefiltert, was selbst bei höheren Rauschpegeln als den Trainingsbedingungen zu sicheren Richtlinien führt. Gleichzeitig wird die Richtlinie während des Trainings ermutigt, Sicherheitseingriffe zu minimieren und risikosensitive Entscheidungen mit Hilfe des Verteilungs-RL zu erlernen.

Um die Verallgemeinerung in Bezug auf unerwartete menschliche Verhaltensweisen anzugehen, modellieren wir unbekannte menschliche Absichten als versteckte Parameter im ursprünglich teilweise beobachtbaren Problem, was dazu führt, dass das Problem als Hidden Parameter Block Markov Decision Process (HiP-BMDP) gelöst wird. In dieser Formulierung haben Fahrer mit denselben Absichten dieselben abstrakten MDP-Zustände und unterscheiden sich nur in den Emissionsfunktionen. Dies führt zu einem modularen Rahmenwerk, das während des Trainings effizienter mit Beispielen umgeht und sich in neuen Umgebungen anpassen kann.

Unsere Bewertungen umfassen sowohl Simulationen als auch Experimente in der realen Welt. In den Simulationsexperimenten trainieren und bewerten wir RL-Richtlinien für das automatisierte Fahren an verdeckten Kreuzungen mit Sensorrauschen sowie interaktiven Einfädelungen mit neuen Fahrstilen. Unsere Bewertungen in der realen Welt basieren auf Open-Source-Beobachtungsdatensätzen, um die effiziente Offline-RL aus den Trajektorien beobachteter Fahrer für interaktives Einfädeln auf Autobahnen zu demonstrieren.

In Anbetracht unserer Experimente kann das vorgeschlagene Verteilungs-RL mit Sicherheitsverifikationsebene die Sicherheit garantieren und Notmanöver minimieren, wenn es in Umgebungen mit höherem Sensorsignalrauschen als im Training eingesetzt wird, basierend auf den zur Laufzeit bereitgestellten geeigneten Risikosensibilitätsdaten für die Richtlinie. Darüber hinaus kann das vorgeschlagene HiP-AP RL für interaktives automatisiertes Einfädeln in Umgebungen mit neuen Fahrstilen übertragen werden. Quantitativ kann der vorgeschlagene Verteilungs-RL-Agent die durchschnittliche Fahrzeit um etwa 40% bis 50% im Vergleich zur normalen DQN-Baseline reduzieren. Er erfordert

auch 50% bis 83% weniger Sicherheitseingriffe im Vergleich zur regelbasierten Richtlinie und erhöht die durchschnittliche Fahrzeit leicht.

In realen Einstellungen zeigt der vorgeschlagene Offline-RL-Agent die Fähigkeit, optimale interaktive Einfädelrichtlinien zu erlernen und mit neuen Fahrstilen umzugehen. Wir beobachten eine Erfolgsrate von 100% für diese Richtlinie, wenn sie mit dem proaktiven Sicherheitsmodul kombiniert wird, und zeigen ihre Robustheit gegen ungültige Absichtsvorhersagen gemäß unseren Ablationsversuchen.

Abstract

Automated driving, also known as autonomous or self-driving technology, holds profound significance in modern transportation and beyond. One of the most compelling reasons for the importance of automated driving is its potential to significantly enhance road safety. Human error is a leading cause of accidents on the road, and autonomous vehicles have the potential to drastically reduce these incidents by eliminating factors such as distracted driving, fatigue, and impaired judgment. Moreover, automated systems can react faster than humans to unexpected situations, further reducing the likelihood of collisions. One of the major obstacles for fully automated driving is that the decision-making policy needs to deal with different sources of uncertainties such as perception errors, occluded vehicles and unknown drivers' intentions with different driving styles. This thesis presents an approach based on reinforcement learning to solve this complex issue.

Reinforcement learning (RL) can solve complex decision-making problems by learning long-term optimal policies. In model-free RL paradigm, partial observation is usually addressed by approximating the true state with a history of recent observations (k-Markov approximation). This will however result in policies that are not generalized to be used in environments with slightly different dynamics due to different noise levels or unexpected human behaviors resulting in novel observations. Moreover, safety is usually addressed by penalizing the RL agent for risky situations leading to either few failures at run-time or too much conservative policies. Even *generalist* policies which are trained with multiple noise levels and different human behaviors are not robust to new environments with novel driving styles.

In order to address generalization of RL policies regarding safety, comfort, and utility, we propose to combine distributional RL with a proactive safety verification module to provide safe and risk-aware RL policies. Thanks to the proactive safety layer which assures a feasible fail-safe emergency maneuver based on the worst-case assumptions, unsafe actions from the RL are filtered out

resulting in safe policies even at higher noise levels than the training settings. Concurrently, the policy is encouraged to minimize safety interference during training and learn risk-sensitive decisions using distributional RL.

In order to address generalization regarding unexpected human behaviors, we model unknown human intentions as hidden parameters in the original partially observable problem, leading to solve the problem as a Hidden Parameter Block Markov Decision Process (HiP-BMDP). In this formulation, drivers with same intentions have same high level MDP states and only differ in emission functions. This results in a modular framework which is more sample-efficient during training and adaptive when transferred into new environments.

Our evaluations include both simulation and real-world experiments. In the simulation experiments we train and evaluate RL policies for automated driving at occluded intersections with sensor noise and also interactive merging with novel driving styles. Our real-world evaluations are based on open-source observational datasets in order to demonstrate efficient offline RL from observed drivers' trajectories for interactive merging in highways.

In the light of our experiments, the proposed distributional RL with safety verification layer can guarantee safety and minimize emergency maneuvers when transferred to environments with higher sensor noise than training. Moreover, the proposed HiP-AP RL for interactive automated merging can transfer to environments with novel driver styles. Quantitatively, the proposed distributional RL agent can reduce the average driving time up to 50% compared to the normal DQN baseline. It also requires up to 83% less safety interference compared to the rule-based policy and slightly increases the average driving time.

In real-world settings, the proposed offline RL agent shows ability to learn optimal interactive merging policies from observed human trajectories. We observe 100% success rate for this policy when it is combined with the proactive safety module and show its robustness against invalid intention predictions according to our ablation experiments.

Table Of Contents

Abbreviations and Symbols

Abbreviations

RL	Reinforcement Learning
AI	Artificial Intelligence
DQN	Deep Q-Network
IQN	Implicit Quantile Network
PPO	Proximal Policy Optimization
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
HiP	Hidden Parameter
HiP-MDP	Hidden Parameter Markov Decision Process
BMDP	Block Markov Decision Process
HiP-BMDP	Hidden Parameter Block Markov Decision Process
HiP-AP	Hidden Parameter Adversarial Perturbation
BCQ	Batch-Constrained Deep Q-learning
PO-BCQ	Partially Observable BCQ
HiP-AP-BCQ	Hidden Parameter Adversarial Perturbation BCQ
VAE	Variational Auto-Encoder

IDM	Intelligent Driver Model
2D	Two Dimensional
PSS	Proactive Safe State
PSA	Proactive Safe Action
BNN	Bayesian Neural Network
MLP	Multi-layer Perceptron

Symbols

General

$\mathbb{R}$	real numbers
$\mathbb{E}$	expected value
U	continues uniform distribution
$\mathcal{N}$	normal distribution
t	time

Markov Decision Processes

$\mathcal{S}$	state space
$\mathcal{O}$	observation space (POMDP)
$\mathcal{X}$	context space (BMPD)
Θ	hidden parameter space (HiP-MPD)
$\mathcal{A}$	action space
R	reward function

T	environment transition distribution
Z	observation distribution (POMDP)
θ	list of hidden parameters for one task (episode) (HiP-MDP)
P_θ	distribution over hidden parameters (HiP-MDP)
T_θ	environment transition distribution for hidden parameter θ (HiP-MDP)
q	emission distribution (BMDP)
γ	discount factor

Decision-Making

d_k	longitudinal distance of vehicle k (or ego) on its path to conflict zone
v_k	longitudinal velocity of vehicle k (or ego) on its path
θ_k	driver intention of vehicle k
l_k	lane k at intersection
d_{l_k}	longitudinal distance of closest occluded grid to the conflict zone at line k
v_{l_k}	speed limit on lane k
a_{ego}	longitudinal acceleration of the ego vehicle on its path
$a_{\text{ego}}^{\max}$, $a_{\text{ego}}^{\min}$	maximum and minimum acceleration of the ego vehicle
$a^{\max}$, $a^{\min}$	maximum and minimum acceleration of other vehicles
$\mathcal{H}$	set of worst-case assumptions
$\hat{s}^a_{\mathcal{H},o}$	simulated worst-case state by observation o and action a with assumptions in $\mathcal{H}$
$d_{\text{ego}}^{\text{FS}}$	ego distance to conflict zone after a full stop maneuver

t_i^{HW}	time headway between ego vehicle and earliest vehicle on lane i
w_{CZ}	conflict zone width
d_{SG}	safety gap distance (emergency stop maneuver)
t_{SG}	safety gap time (emergency leave maneuver)
$A_{\mathrm{PSA}}(o)$	set of safe actions for observation o
TX_i	time for the vehicle i to reach the conflict zone with constant velocity
TTC	time for the vehicle i to reach the conflict zone with constant velocity
t_{AP}	threshold on TTC to apply adversarial perturbation on true value of driver intention

Reinforcement Learning

$Q^\pi(s,a)$	expected value of return for policy π when choosing action a at state s
$\mathcal{Z}^\pi(s,a)$	distribution of return for policy π
$\mathcal{T}$	Bellman optimality operator
α	risk-sensitivity parameter in IQN
a_{emg}	emergency action from safety layer
λ	collision penalty
w^Q	neural network parameters for Q network

1 Introduction

Several real-world problems can be modeled as sequential decision-making problems where a decision at different time sequences needs to be made in order to fulfill some long term goals. This can be the treatment for a patient given the current patient characteristics or the future trajectory of an automated vehicle while driving on the highway. Most real-world sequential decision-making problems are not easy to solve with a simple solution. It requires predicting the future state in the environment for different possible actions, and selecting the action which results in the best outcome.

The main challenge is that the outcome not only depends on the current, previous, and future decision sequences, but also depends on several environmental factors. Moreover, good or bad decisions are not always identifiable from spontaneous environmental feedback. A poor decision may result in an adverse outcome after several future decisions, making it difficult to determine the main cause of unwanted outcomes.

Suppose that a physician wants to find the optimal treatment for a patient. They should consider several factors that represent the current state of the patient, as well as previous treatments the patient may have received. Based on the patient's state, the physician should study the effectiveness of different treatment options. One treatment may result in different outcomes when applied to patients with slightly different states. The effectiveness of a treatment can depend on several factors, such as genetics, previous treatments, and other patient characteristics. For a young patient, a strong treatment with certain side effects may be the best option, while for an old patient, such treatment can be life-threatening.

Artificial Intelligence (AI) can help to address these complex decision-making problems by training deep neural networks which can learn the complex relationships between observations and long-term outcomes for different actions. Reinforcement Learning (RL) [SB18] is a machine-learning paradigm that can address sequential decision-making problems in different domains such as health

[Fat+21] and automated driving [Ise+18; INF18; Tra+18; KZL19; Kam+20; Mir+18; Hue+19; Kam+22]. RL learns the best decision-making solution from several interactions with the environment and maximizing numeric rewards that it receives from the environment.

Learning from previous interactions is a natural mechanism that we all experience and benefit from, even from the beginning of our lives. A child learns how to stand up and walk for the first time through several trials and errors, and by reasoning about causal relationships between observations, actions, and outcomes. In health or robotics, several existing datasets about different treatments or actions applied in the real world can be an important source of information to learn optimal decisions from previous experiences with RL.

1.1 Motivation

Several previous works have leveraged the high computational power of deep RL methodologies as the main decision-making module in automated driving across various challenging scenarios. Although these approaches can address the complex problem of sequential decision-making in automated driving, they are usually not generalized solutions for different real-world applications. The motivations for this thesis regarding generalized RL for automated driving are three-fold, as explained in the following sections.

1.1.1 Safe Reinforcement Learning for Automated Driving

Decision-making policies for automated driving do not have access to accurate and complete information about the environment. This information comes from perception systems such as cameras, which are prone to estimation noise and sensor occlusions. Therefore, an RL policy should be robust to these challenges, which is usually not the case in previous works, where the agent is trained in an ideal simulation environment assuming perfect perception.

Another issue regarding safety of RL policies at deployment time is related to their deep learning-based architectures. These architectures are usually the main computational power of RL policies in order to extract useful features from high dimensional and noisy input data in order to estimate the value of

different decisions using deep neural networks. Due to their black-box nature, deep neural networks are usually hard to verify [Liu+19]. Moreover, they are vulnerable to different problems related to their training procedure such as overfitting and sample inefficiency. Due to overfitting, they may perform well only for specific variation of the data which is usually the part similar to the training and validation datasets. This results in poor generalization of RL policies in situations where test data is not similar to the validation data, and therefore the model may not be safe in those corner case scenarios.

In order to learn generalized RL policies, one can design deeper neural networks and train the policy using different possible scenarios and corner cases in order to make the policy optimal for all situations. The main issue related to that solution is the requirement for huge amount of training data in order to cover all possible scenarios. This also requires huge and complex neural networks which can result in catastrophic forgetting, a general challenge in sequential decision learning and deep neural networks [MC89; Goo+13].

Another issue regarding generalized and safe RL for automated driving is that not all corner case scenarios can be provided during training. This is inherently a problem when humans are part of the system, as they generally act as non-stationary elements in an environment [Xie+20]. In critical applications like automated driving, factors such as local infrastructure, traffic density, and weather conditions may lead to varying behaviors of human-controlled agents. Therefore, it can always be possible for the trained policy to observe novel scenarios and confront out-of-distribution dynamics in the test environment, which can result in unsafe outcomes.

1.1.2 Interactive Reinforcement Learning under Uncertainty

Another main challenge in RL is uncertainty about important features that impact the likelihood of outcomes. These features can play a key role on the outcome of decisions but not directly observable, such as genetic information of a patient which can relate to the efficacy of the treatments on that patient. In automated driving context, these important features can be the intention of other drivers in the interactive scenarios. In merging scenarios, accelerating to merge before a cooperative driver is a safe decision, while the same action

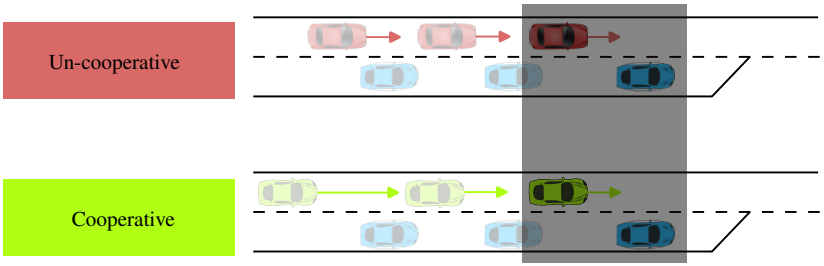


Figure 1.1: Two different interactive automated driving scenarios with different driver intentions that have exactly same observations (shaded region) at the current time. In order to predict drivers' intention and make optimal decision, a history of recent observations is necessary.

for having interaction with a non-cooperative driver may result in unsuccessful merging.

RL theoretically models the decision-making problem as Markov Decision Process (MDP) [How60], assuming it has access to all required information for optimal decision-making represented as the current state of the environment. However, this is not a realistic assumption, as important genetic information of the patient may not be always in access or the main intention of drivers is not always predictable. Fig. 1.1 shows two interactive merging scenarios with different driver intentions. In the bottom scenario, the other vehicle has a cooperative intention. Although it has a higher velocity (longer arrow in the figure) at the beginning, it decelerates to open a merging gap for the ego vehicle. In the top scenario, the other vehicle is non-cooperative and drives with constant velocity. Although these are two different scenarios with different MDP states, the current observations for both of them are identical due to partial observation about drivers' intentions. Therefore, standard RL models can not easily be applied in such interactive scenarios in real-life, where humans' intention play an important role but are not directly observable.

One way to address intention uncertainty in RL is to provide history of recent observations as an approximation of the current state [MS13; Bou+18; Kam+20]. The goal is to enable RL to predict the intention of drivers implicitly through its previous interactions with them. However, this requires high-dimensional observations containing long enough history of previous interactions. The

policy then needs to consider multiple MDPs for multiple tasks (e.g. different driving styles) which results in *super-MDPs* [Zha+20a].

In order to learn optimal policies for such complex super-MDPs, a huge amount of data is required to capture all possible behaviors and observations [Sod+22; Du+19], which is not realistic and prone to different issues explained in previous section such as catastrophic forgetting and corner cases.

Moreover, RL policies that are trained based on long history observations have a poor performance when transferred into new environments with different transition dynamics. Suppose that an RL policy is capable to identify cooperative drivers by observing their previous trajectories trained by datasets from one country and now transferred into new country. Although the policy can generate optimal decisions in scenarios very similar to its training environment, it may generate suboptimal decisions in the new environment when the drivers have totally different behaviors compared with the training environment. Therefore, generalization is an important requirement for intelligent policies used for automated driving.

1.1.3 Offline Reinforcement Learning using Real-World Observational Datasets

Although RL is a powerful methodology to solve different decision-making tasks, it usually requires online interactions with the environment by its current policy in order to improve it. This makes majority of standard RL algorithms not applicable in crucial real-world settings were data gathering is very expensive, difficult or even not possible. The amount of randomized controlled trial data is much lower than real-world data in clinical research because we cannot easily compare the effectiveness of random treatments on different patients. Similarly, in automated driving, it can be very dangerous to train an RL model directly on highways to learn how to prevent risky situations.

Recently offline RL [Lev+20; FMP19; ASN20], as new paradigm in RL has gained a huge attention to solve decision-making problems by observing other policies' interactions in the environment. This methodology does not require new interactions with the environment and therefore can benefit from already existing interactions from other policies called *behavior policies*. The behavior

policies can be either suboptimal or expert, and the RL algorithm should be able to learn the optimal policy in both cases. This makes RL to be used in crucial real-world applications such as health or interactive automated driving. Moreover, with the help of technology advances, gathering, management, and access to several observational datasets in these applications are becoming more and more possible. For example, in [Zha+19], several hours of vehicles' trajectories driving in highways and urban areas are provided which can be used to train automated driving RL policies. This motivates to concentrate on methodologies that can mitigate from this immense amount of available data and learn the decision-making solutions from those observed behaviors. Therefore, in this dissertation, we demonstrate training and evaluating offline RL algorithm for interactive automated driving in the merging scenario using this observational datasets. We show how we train our offline RL policy without requiring to gather new interactions in the environment and utilize the available observational dataset for our scenario.

1.2 Contributions

We split the main contributions of this thesis into three main aspects and explain them in the following sections.

1.2.1 Safe Reinforcement Learning for Automated Driving under Uncertainty

We address safety, scalability and comfort as intertwined challenges for RL under uncertainty. Similar to [SSS16], we consider safety as a hard constraint in the decision-making problem and therefore utilize RL to optimize other targets (utility and comfort) as soft constraints. For that, we utilize a proactive safety layer similar to [Nau+19] which evaluates if an emergency fail-safe maneuver by the worst-case assumption exists in order to verify the actions generated from the RL policy. We will show how the proposed concept helps to learn verified safe RL policies that are not prone to high sensor noise or corner case scenarios and always perform safe under specific circumstances for maximum perception noise and compliant behaviors for other participants.

Although safety verification is done by the worst-case proactive module, the RL policy still needs to minimize the amount of emergency safety interruptions to generate comfortable behaviors. In order to make the policy adaptive in terms of choosing more/less conservative behaviors, we apply distributional RL which learns distributions of action returns instead of their expected values. This feature makes the learned policy generic and applicable to high uncertainty environments, or corner case scenarios that may happen in real-world.

1.2.2 Interactive and Transferrable RL Policies under Intention Uncertainty

Our contribution to address interactive automated driving under intention uncertainty is twofold. First, we model interactive automated driving under intention uncertainty with Hidden Parameter Markov Decision Process (HiP-MDP) [Zha+20a] where human strategies are modeled as behavioral latent variables with different emission functions for different drivers. Different cooperative drivers all decelerate in the merging scenario to open a merging gap for our vehicle, while they may have different deceleration and speed profiles (different emissions for the same hidden parameter). In contrast to history-based approaches which address intention uncertainty by *implicit* intention prediction, in our framework, the RL policy is directly conditioned on the output of drivers' intention predictors which are trained separately as supervised learning models. This results in a more compact state vector and a more sample-efficient RL comparing to the history-based RL solutions.

In order to learn robust RL policies which can use plug-in intention predictors, we propose Hidden Parameter Adversarial Perturbation (HiP-AP) RL agents that are trained in the oracle environment with adversarial perturbations [Zha+20b] on the true hidden parameters. We will show that HiP-AP policies can directly transfer to new environments with different emission functions by just updating their online hidden parameter estimators or using an updated external intention predictor without any requirement to update the reinforcement policy itself.

1.2.3 Safe and Interactive RL with Real-World Datasets

Finally, we demonstrate efficiency of safe HiP-AP RL as our proposal to learn interactive policies under intention uncertainty using real-world datasets. We show how our solution can address safety and sample efficiency of RL policies when they are trained by limited observational datasets instead of simulators.

1.3 Outline

The remaining parts of this dissertation are structured as following: In Chapter 2, first fundamentals about MDP and partially observable MDPs which we use for automated driving problem formulation are explained. Then we explain fundamentals of RL which we apply to provide decision-making for automated driving.

In Chapter 3, our approach for addressing safety in RL for automated driving is presented. We focus on safety and comfort of RL policies for automated driving at occluded intersections with noisy observations.

In Chapter 4, first we present our approach addressing interactive and generalized RL for automated driving under drivers' intention uncertainty. Then we explain our simulations and real-world experiments, and evaluate the efficiency of our framework compared with different baselines in terms of safety, sample efficiency, and generalization.

Finally, Chapter 5 will conclude this dissertation and provides insights and directions for potential future works.

2 Fundamentals

Sequential decision-making for robotics and specifically self-driving vehicles can be modeled as a Markov Decision Process (MDP). Since usually in reality not all required information for optimal decision-making is available, most of the problems should be modelled with Partially Observable Markov Decision Process (POMDP). In this dissertation, we mainly focus on benefitting from high computational power of RL in order to solve POMDP problems in automated driving.

For that, in this section, we briefly explain fundamentals about MDP, POMDP, and its other subtypes which later will be used in our approaches. After explaining important models for our decision-making problem formulations, we will have an introduction about RL fundamentals as machine-learning-based decision-making solution for them. Recently several varieties of RL methodologies have been proposed which we briefly review some well-known algorithms related to this dissertation.

2.1 Markov Decision Process (MDP)

MDP [How60] is a well known model for formulating decision-making problems in different real-world applications. An MDP is represented with a tuple $< \mathcal{S}, \mathcal{A}, R, T, \gamma >$ All the required information about the surrounding environment at time t is described by a state $s_t \in \mathcal{S}$. The action space $\mathcal{A}$ identifies all possible decision options, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, $T : \mathcal{S} \times \mathcal{A} \rightarrow Dist(\mathcal{S})$ is the environment transition probability function, and $\gamma \in [0, 1)$ is the discount factor.

The action can be discrete or continues based on the application. The action selection process is described by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which is a mapping from states to actions (or to distributions over actions). The distribution of the

successor state s_{t+1} is defined by the transition model $s_{t+1} \sim T(s_t, a_t)$ based on the current state and chosen action. In every step, depending on the state and applied action, a reward is provided from the reward function $r_t = R(s_t, a_t)$. An optimal policy in MDP will maximize the cumulative discounted reward (also called return):

$$\sum_{k=0}^{\infty} \gamma^t r_{t+k} \tag{2.1}$$

Where future rewards are discounted by the factor $\gamma \in [0, 1)$.

2.2 Partially Observable Markov Decision Process (POMDP)

In many real-world applications, the agent does not have full knowledge of the environment's state but instead receives only partial observations of the state. These problems can be formulated by a Partially Observable Markov Decision Process (POMDP) and represented by the tuple $< \mathcal{S}, \mathcal{A}, R, \mathcal{O}, T, Z, \gamma >$. Similar to MDP, $\mathcal{S}$ is the state space, $\mathcal{A}$ action space, R reward function, and T is the environment transition probability function. In addition, $\mathcal{O}$ is the observation space and Z is the observation model. At each step, the agent receives an observation $o_t \in \mathcal{O}$ which depends on the current state and chosen action and is distributed according to the observation model $o_t \sim Z(s_t, a_t)$. The agent's goal in POMDP problems is still to maximize the future return, but with the additional challenge of uncertainty about the environment state.

2.3 Hidden Parameter Markov Decision Process (HiP-MDP)

Several real-world problems that are originally POMDP, can be described by a finite set of MDPs which are identified with a set of finite hidden parameters. As an example, learning a policy for a robot to swing a bat with unknown length and mass is originally a POMDP but for fixed values of length and mass, it will become an MDP. This problem can be formulated as a family of bat swinging tasks which differ in the hidden parameters as length and mass of the

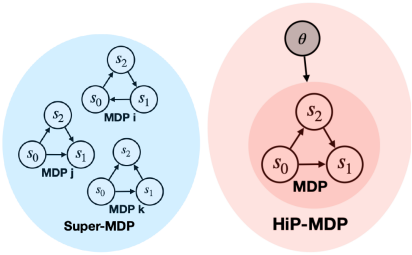


Figure 2.1: Left: a Supper-MDP problem which can be represented as HiP-MDP in right (graphic from [Zha+20a]).

bat. The main advantage of this formulation is the efficiency in the training and generalization. After training a policy for specific tasks, the policy does not need to be trained from scratch for a new task.

Hidden Parameter Markov Decision Process (HiP-MDP) [DK16] was proposed to model such family of tasks which differ in the underlying dynamics that are captured by a set of hidden parameters. HiP-MDP is represented by a tuple: $< \mathcal{S}, \mathcal{A}, R, T_\theta, \Theta, \gamma, P_\theta >$. Similar to MDP, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space. T_θ describes the transition dynamics of the environment for specific task with hidden parameter (probably a set) θ. In HiP-MDP, the reward function R is shared across all task instances. Hidden parameters are drawn at the beginning of each task instance by probability P_Θ from possible hidden parameters set Θ and are fixed until the end of the task.

HiP-MDP has these important assumptions:

- Considering a fixed (but unknown) task parameters (θ), the transition function is also fixed (only depends on the state).
- The hidden parameters for each task are constant and have no dynamics until the end of the task.
- The hidden parameters are the only information required to specify the task and the underlying MDP.

These features of HiP-MDP allow its usage in several applications where a family of tasks are required to solved together, happening arbitrarily in a shared environment. Although the hidden parameters are not observable directly, they can be identified during the interaction with the environment and once they are identified, the whole transition function is captured and the corresponding policy can be applied for that particular task.

2.3.1 Block Markov Decision Process (BMDP)

Block Markov Decision Process (BMDP) [Du+19] is described by a tuple: $< \mathcal{S}, \mathcal{A}, R, \mathcal{X}, T, q, \gamma >$. Here the finite state space ($\mathcal{S}$) is unobservable but the context space $\mathcal{X}$ is observable. $\mathcal{A}$ is action space and T latent state transition distribution function. q is the (possibly stochastic) context emission mapping function which provides the conditional probability $q(x|s)$ for all $s \in \mathcal{S}$ and $x \in \mathcal{X}$. Similar to the POMDP, in the BMDP the state is not directly observable. However, the main difference is that in the BMDP, the whole context space is partitioned into disjoint blocks $\mathcal{X}_s$ each referring to the underlying unobservable state based on the conditional distribution $q(x|s)$.

2.3.2 Hidden Parameter Block Markov Decision Process (HiP-BMDP)

Hidden Parameter Block Markov Decision Process (HiP-BMDP) [Zha+20a] combines HiP-MDP and BMDP settings, both of which describe partially-observable systems and build upon the definition of a MDP. Similar to HiP-MDP, it models a family of tasks which have different hidden parameters. In addition, it assumes that the true states are not observable, and we only have access to the observable context that refers to unique state (assumption in Block-MDP).

A HiP-BMDP is described by a tuple $\{\mathcal{S}, \mathcal{A}, R, T_\theta, \Theta, P_\theta, \mathcal{X}, q, \gamma\}$ where $\mathcal{A}$ is action space, $\mathcal{S}$ describes the unobservable state space and $\mathcal{X}$ the observable context. In a HiP-BMDP, the transition function T_θ is conditioned on the value of a task parameter $\theta : \theta \in \Theta$ (HiP-MDP property). This hidden parameter is sampled from a probability distribution $\theta \sim P_\Theta$. Additionally, q is the emission

mapping that describes the probability of observing x given the true state s: $q(x|s)$ for $x \in \mathcal{X}, s \in \mathcal{S}$.

2.4 Reinforcement Learning

Reinforcement Learning (RL) [SB18] is a type of machine learning which is usually used for solving decision-making problems that are modelled by MDPs. RL learns the optimal decision-making policy through several trial and error interactions with its environment and adapting its actions based on the reward it receives from the environment. By employing techniques such as exploration and exploitation, RL algorithms aim to discover optimal strategies that lead to the most favorable long-term outcomes through maximizing the discounted cumulative reward (equation 2.1).

Several algorithms for RL have been proposed which can be categorized based on their learning paradigm into model-based and model free groups. Model-free RL involves the agent directly learning a policy or value function from its interactions with the environment, without building an explicit model of the environment. This includes methods like Q-learning [WD92], where the agent learns action-values, and policy gradient methods like Proximal Policy Optimization (PPO) [Sch+17], which optimize policy parameters directly.

On the other hand, model-based RL entails the agent constructing a model of the environment, enabling it to simulate possible outcomes and plan accordingly. This category includes techniques like Monte Carlo Tree Search (MCTS) [Sil+16], where the agent combines real experience with simulated experience generated from its model. Both model-free and model-based approaches have their strengths and weaknesses, and their efficiency for specific task depends on factors such as the complexity of the environment and the availability of computational resources.

RL algorithms can also be categorized based on the policy that is used for interaction with the environment for gathering training data. If the policy for training data collection is exactly same as the RL policy, it is an on-policy RL type (like PPO). On-policy RL potentially leads to slower learning due to lack of reusing old collected data.

On the other hand, off-policy RL involves training an agent's policy using data collected by a different policy, which is usually the old version of actual RL policy and therefore allowing efficient reuse of previous experiences (such as Q-learning [WD92]).

As a new type of RL, Offline RL, or batch learning (e.g. [Lev+20; FMP19; ASN20]) has recently gained a huge attention, which is based on training from a fixed dataset of experiences, without further interaction with the environment to gather new training data. These type of RL methods address the challenges posed by safety concerns, high-cost interactions, or restricted access to live data collection, making it applicable in scenarios like robotics, autonomous systems, and healthcare. However, ensuring robust generalization from historical data and effectively handling issues like distributional shift remain active research areas regarding optimal offline RL methodologies.

In the remaining parts of this section, we explain a few well-known RL algorithms as fundamentals for our baselines and proposed models in the next chapters.

2.4.1 Deep Q-Network (DQN)

A famous approach to find an optimal policy with RL is Q-learning [WD92] which tries to train a policy that maximizes the expected future return, defined as state, action value, or Q value for a policy π:

$$Q^{\pi}(s,a) = \mathbb{E}\left[R(s,a) + \sum_{k=1}^{\infty} \gamma^k R(s_k, \pi(s_k))\right], \\ s_i \sim T(.|s_{i-1}, a_{i-1}), s_0 = s, a_0 = a$$

after choosing action a in state s and following policy π thereafter.

Using the Bellman equation [Bel66], the policy ϕ Q function can be represented as:

$$Q^{\pi}(s,a) = \mathbb{E}_{s' \sim T, a' \sim \pi}\left[R(s,a) + \gamma Q^{\pi}(s',a')\right].$$

Similarly, the optimal Q function can be represented as:

$$Q^*(s,a) = \mathbb{E}_{s' \sim T}\left[R(s,a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a')\right].$$

In DQN [MS13], deep neural networks are utilized to learn the optimal Q values for each state and action over samples from a replay buffer. For that, in every training interaction, the target Q network parameters ($w^{Q'}$) are updated through minimizing the loss function for B random samples of (s_i, a_i, r_i, s_{i+1}) tuples (state, action, reward, next state) from the replay buffer:

$$L(w^{Q'}) = \sum_{i=1}^{B} (y_i - Q(s_i, a_i | w^{Q'}))^2 \tag{2.2}$$

where y_i is the network training target based on the current Q network (w^{Q}):

$$y_i = r_i + \gamma \max_{a'} Q(s_{i+1}, a' | w^{Q}) \tag{2.3}$$

The parameters of the target network $w^{Q'}$ are updated to the current network parameters w^{Q} after a fixed number of iterations (hard update), or by averaging (soft update) $w^{Q} = \beta w^{Q'} + (1-\beta) w^{Q}$ for some small β ([Lil+15]).

The use of max operator in the network training target can result in large overestimation of the action values. In order to resolve this bias, Double DQN algorithm [VGS16] uses two separate Q-networks. During each update, one Q-network is used to select the best action, while the other Q-network evaluates the chosen action's value which prevent the same Q-values from being selected and evaluated at the same time. This technique is usually helpful for more accurate and stable learning particularly in environments where reward or state transitions are stochastic.

2.4.2 Distributional Reinforcement Learning

In normal RL algorithms like DQN, the expected values of actions are being learned. However, more information about the distribution of the learned values are vital for risk-aware decision-making.

Distributional RL [BDM17] tries to learn a return distribution $\mathcal{Z}^\pi(s_t, a_t) \overset{D}{=} R(s_t, a_t) + \sum_{k=1}^{\infty} \gamma^k R(s_{t+k}, \pi(s_{t+k}))$ instead of its expected value. Note that here $\overset{D}{=}$ indicates equality in distribution. The value distribution can be computed using dynamic programming based on the *distributional Bellman equation* [BDM17] :

$$\begin{aligned} \mathcal{Z}^\pi(s,a) &\overset{D}{=} R(s,a) + \gamma \mathcal{Z}^\pi(S', A') \\ S' &\sim T(\cdot|s,a), A' \sim \pi(\cdot|s') \end{aligned} \tag{2.4}$$

This equation is shown in [BDM17] to be a contraction in the Wasserstein metric.

Similar to regular RL, the *distributional Bellman optimality operator* ($\mathcal{T}^*$)can be applied to the distributional RL:

$$\begin{aligned} \mathcal{T}^*\mathcal{Z}(s,a) &\overset{D}{=} R(s,a) + \gamma \mathcal{Z}(S', \arg\max_{a' \in \mathcal{A}} \mathbb{E}\left[\mathcal{Z}(S', a')\right]) \\ S' &\sim T(\cdot|s,a) \end{aligned} \tag{2.5}$$

In order to learn the optimal return distribution, Bellemare et al. parameterized it as a categorical distribution over a fixed set of equidistant points [BDM17]. Their approach, called *C51*, minimizes the Kullback-Leibler divergence [KL51] to the distributional Bellman targets, which, however, was not a contraction in the Wasserstein metric.

Later Dabney et al. proposed to learn return distributions through *Quantile Regression* (QR-DQN) on a fixed set of quantiles, minimizing the Wasserstein distance to the distributional Bellman targets [Dab+18b]. In a newer approach, Dabney et al. introduced Implicit Quantile Network (IQN) [Dab+18a] to approximate the quantile function $F_{\mathcal{Z}}^{-1}(\tau)$ for the random variable $\mathcal{Z}$. Assuming $\tau \sim U([0,1])$, the return distribution can then be sampled from $F_{\mathcal{Z}}^{-1}(\tau)$ as samples from the implicitly defined return distribution. The main advantage of IQN is that any distortion risk measure $\beta : [0,1] \to [0,1]$ for different types of desired risk-aware behaviors can be incorporated to compute distorted expectations of Z:

$$Q_\beta(s_t, a_t) = \mathbb{E}_{\tau \sim U([0,1])}\left[\mathcal{Z}_{\beta(\tau)}(s_t, a_t)\right],$$

and the risk-sensitive greedy policy:

$$\pi_\beta(s_t) = \arg\max_{a \in \mathcal{A}} Q_\beta(s_t, a).$$

In this dissertation, we utilize the IQN implementation of distributional RL since it allows exploring risk-sensitive policies π_β during training which helps us to learn a family of risk-aware policies using only one neural network.

2.4.3 Batch-Constrained Deep Q-learning (BCQ)

Batch-Constrained Deep Q-learning (BCQ) is an offline type of RL proposed by Fujimoto et al. in [FMP19]. It combines both imitation learning and RL paradigms at the same time, and therefore it can benefit from both expert and suboptimal policy observations in the training dataset. It should be noted that pure imitation learning methods can only learn optimal policies from pure expert datasets and therefore are not capable of learning from suboptimal behaviors.

Extrapolation error is one of the main challenges in offline or batch RL which refers to the difference between the true state action visitation in the current policy and the dataset. The main reason for extrapolation error can be incomplete datasets, biased batches, or the mismatch between the distribution of data and the current policy.

In order to address extrapolation error in offline RL, BCQ proposes to enforce the current policy have similar state-action visitation to the training batch. Therefore, in BCQ, in addition to optimizing value function, a constraint limits the amount of distance between the selected actions from the policy to the dataset, and therefore the method is called *batch-Constrained* Q-learning. For that, a state conditioned generative model based on conditional Variational Auto-Encoder (VAE) [KW13; RMW14] is used to produce actions similar to the dataset for a give state. The generated actions from this model are then perturbed with another neural network for optimal policy training. The final

policy then samples n actions from the state conditioned generative model (G_ω) and selects the action with the highest value estimate (Q^w):

$$\pi(s) = \arg \max_{a_i + \mathcal{E}_\phi(s, a_i, \Phi)} Q_w(s, a_i + \mathcal{E}_\phi(s, a_i, \Phi)), \quad \{a_i \sim G_\omega(s)\}_{i=1}^n. \tag{2.6}$$

Here $\mathcal{E}_\phi(s, a, \Phi)$ represents the perturbation model which outputs a constrained adjustment for action a in range $[-\Phi, \Phi]$. Parameters n and Φ can tune the amount of similarity between RL actions to dataset actions and therefore $n = 1$ and $\Phi = 0$ results in complete behavior cloning instead of RL.

3 Safe and Risk-Aware RL

Safety is one of important challenges in real-world applications of AI. Specially when AI is being used for decision-making in automated driving, wrong decisions can result in unwanted and catastrophic outcomes. Black-box nature of AI makes it very hard to understand and debug the main logic behind the AI model. Moreover, it is very difficult to make sure that a trained AI model always has the same performance and safety guarantee measured before deployment.

Similarly, a RL policy may have undesired outcomes at deployment time due to observing out-of-distribution inputs or paying attention to different criteria such as utility rather than safety as the highest priority for optimal decision-making. Most of previous works, address safety of RL in automated driving by providing big negative rewards when the RL results in collisions during training in simulations. However, these *Penalty-Based* solutions can not be considered as the final solution for providing safe-RL policies as we will explore their drawbacks through some experiments in this chapter.

In this chapter, we focus on approaches for providing safe and risk-aware RL policies in automated driving based on worst-case safety verification. We first review some of the previous works which address safety in RL and then explain our approach for tackling this issue. Finally, we evaluate our approach in different experiments and compare its performance with other baselines.

3.1 Related Work: Safe Reinforcement Learning

In this section we briefly summarize previous works that address safety of RL agents in automated driving or other real-world applications in robotics. We categorize state-of-the-art safety related approaches into three major groups: 1) Penalty-based safety encouragement, 2) Shielding mechanism and 3) Decoupling safety from RL.

3.1.1 Penalty-Based Safety Encouragement

In order to encourage learning safe policies which prevent the agent to be in unsafe states, one solution is to apply penalties in the reward function as part of the MDP model. As proposed in [Ise+18; Tra+18], for encouraging safe behaviors, the reward function will usually have this form:

$$R = \begin{cases} -\lambda, & \text{if collision} \\ 1.0, & \text{if success} \\ 0.0, & \text{otherwise} \end{cases} \tag{3.1}$$

Applying collision penalties in the reward function will automatically assign lower values for unsafe actions and therefore the RL agent learns to select actions that result in safer trajectories.

Although the learned policies are prevented to generate risky behavior during training by receiving large penalties for having collisions [Ise+18; Tra+18; Kam+22] or being in risky situations [INF18; Kam+20], there is no guarantee that the learned policy is always safe after training. This can be the result of different issues related to improper policy training such as catastrophic forgetting [MC89] or permutation sensitive input representation for the RL neural network [Zah+17]. Unsafe behaviors can also happen during out of distribution observations in the test environment, such as edge case configurations which rarely or never happened during training and therefore the RL agent does not generate suitable safe behavior when it occurs.

One can try to learn a policy which reduces the amount of risk to be still safe in rare but challenging scenarios like [Kam+20], however, such a policy then becomes too conservative in normal scenarios. This dilemma between more conservative but slower policies and risky but faster policies is thoroughly elaborated by Isele et al. in [INF18] where higher timeout penalties as part of the reward function resulted in a faster but more risky policy. The origin of this problem lies in the difficulty of shaping the reward to guarantee having safe policies in gradient based learning approaches as discussed in [SSS16].

We also investigate the result of setting different crash penalties on the behavior of the trained RL policies for automated driving at simulated intersection scenarios. For that, we trained different RL policies based on Proximal

Policy Optimization (PPO) [Sch+17] RL model that have penalty-based reward functions as Equation 3.1 with different crash penalties (λ). Figs. 3.1 to 3.4 depict average collision rates (shown as average cost) and average durations to finish the scenario (episode length until a success or a crash) for these policies. Note that some policies could not completely converge during the training time fixed across all experiments. We observed that having higher collision penalties will result in safer policies as expected. But those policies become much slower as the result of too much conservative decisions. Moreover, even the policy with the highest collision penalty ($\lambda = 0.2$) is not completely collision free. This proves the fact that penalty-based RL methods suffer from two important challenges: 1) Finding a good trade-off between safety and utility, and 2) Sensitivity of the final trained behavior to the reward parameters.

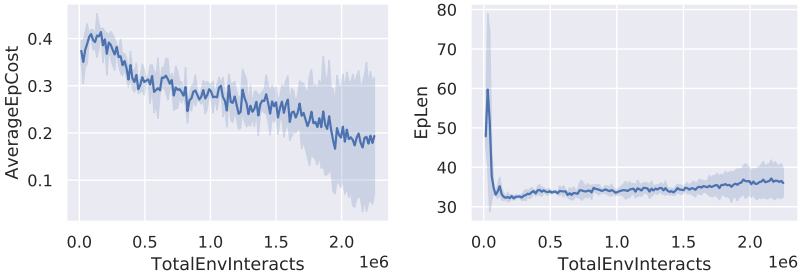


Figure 3.1: Average episode cost and length for penalty-based RL agents with λ=0.10.

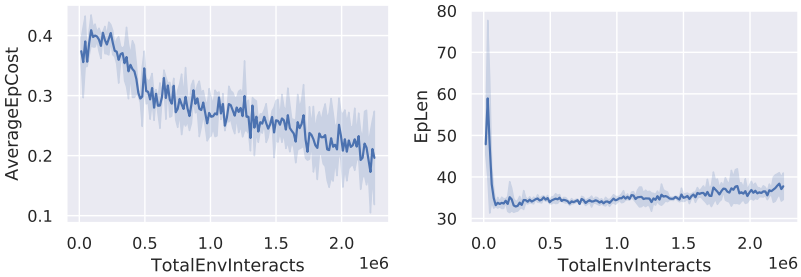


Figure 3.2: Average episode cost and length for penalty-based RL agents with λ=0.11.

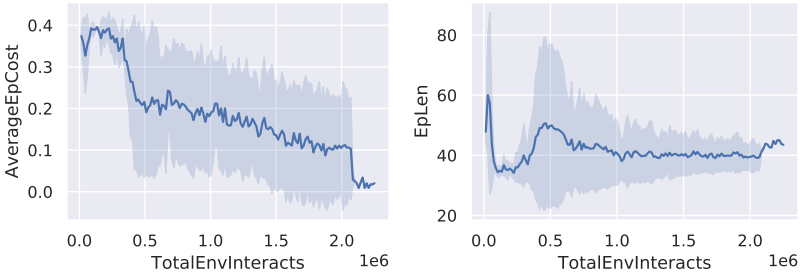


Figure 3.3: Average episode cost and length for penalty-based RL agents with λ=0.14.

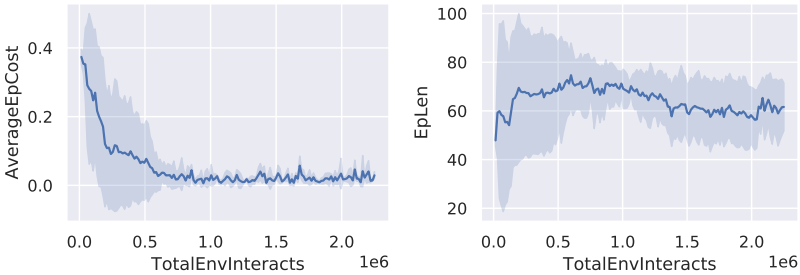


Figure 3.4: Average episode cost and length for penalty-based RL agents with λ=0.20.

Policy sensitivity to the collision penalty can also prevent the policy to be transferred into new environments with slightly different dynamics and observation models (e.g. with more dense traffic, less cooperative drivers or higher sensor noise). Therefore, for a new environment, we should either retrain a new policy from scratch in that new environment or a more conservative policy with bigger safety penalty in the old environment.

3.1.2 Safety Assessment via Shielding

Another approach to address safety for RL policies is to utilize a safety layer which is capable of distinguishing safe and unsafe actions using formal methods. Based on a *shielding* mechanism [Als+17] the safety layer robustly verifies

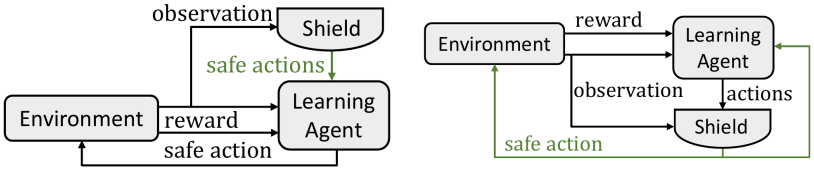


Figure 3.5: Preemptive (left) and post-posed (right) shielding mechanisms for safe RL (graphic from [Als+17]).

safety of generated actions from the RL policy. Authors in [Als+17] studied the effect of applying the safety shield before or after the RL agent as *Preemptive* and *Post-Posed Shielding* mechanisms (Fig. 3.5). In preemptive shielding, the list of safe actions are first provided as the input for the RL agent which then selects one of the safe actions, while in post-posed shielding the safety layer monitors the RL output and corrects the generated action if it violates safety specifications. The authors showed that safety verification mechanisms not only address safety issues, but they can also be helpful for faster convergence of RL.

The post-posed shielding mechanism has been recently used for safety verification of RL policies used for lane change decision-making of automated vehicles in [Mir+18; KWA20]. Authors in [Mir+18] used formal methods to verify safety of a lane change action based on the required constraints for a safe lane change maneuver. They deployed DQN in order to approximate $Q(s,a)$ for every lane change action. Using the safety layer, possible safe lane change actions are generated for every state and then the safe action with the highest Q value from the DQN is selected as the final action. As the fail-safe action, the automated vehicle will keep its current lane whenever both right and left lane change actions are unsafe. Therefore, the agent always performs safe and does not require any collision penalty in its reward function.

Similarly, in [KWA20] authors proposed a safety based masking framework which excludes unsafe lane change actions. However, the authors added collision-based penalties into the reward function in order to train a policy without applying safety-filter during training and only deploying that in the test phase.

Similar to the shielding perspective, authors in [SSS16] proposed to provide safe RL policies by decoupling safety from the RL in the final framework. In

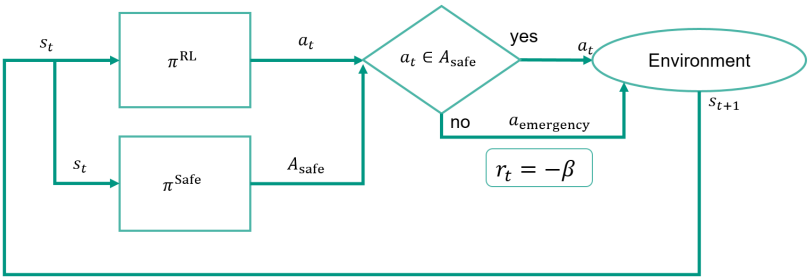


Figure 3.6: General structure of the proposed safe RL framework for automated driving at occluded intersections.

this way, the RL agent is only responsible to generate *soft-constraints* which are then considered in an optimization problem which costs depend on the soft-constraints from RL and also *hard-constraints* related to safety. Our proposed approach to address safety in automated driving can be categorized as part of this general approach.

3.2 Approach

In the context of automated driving, uncertainties due to sensor occlusions, perception noise or ambiguous behavior of other participants need to be considered inside the safety verification constraints. However, there is a trade-off between the level of safety that is guaranteed and efficiency [Sha+01]. Assuming the worst-case perception noise and worst-case occluded vehicles in the safety verification layer may result in unnecessary safety interruptions. On the other hand, one can increase the capability of intervention (e.g. deceleration with -10 m/s^2) and only prevent unsafe actions just before a hazardous event is going to happen. This will result in rare interventions but with extremely uncomfortable maneuvers for the vehicle passengers.

In order to address safety and comfort, we propose safe risk-aware policies by combining the shielding mechanism with distributional RL. Fig. 3.6 depicts the general structure of the proposed safe distributional RL framework. During training, the RL agent is penalized for every safety intervention similar to

[Als+17; LLK20]. However, the main difference is that it learns distributions instead of expected values for each action return which helps to provide risk-aware policies that are able to adapt their conservative behavior according to the existing uncertainty in the environment.

In the remaining parts of this section, we first explain our problem formulation for automated driving at intersections under uncertainties. Then we explain our safety verification approach which makes sure the safety is guaranteed based on assumptions about maximum velocity of other vehicles. Finally, we will explain our safe RL agent which is based on distributional RL in order to provide adaptive safe policies for automated driving at intersections with different uncertainty levels.

3.2.1 Problem Formulation

We consider the problem of automated driving at occluded intersections where the ego vehicle has uncertainties about the state of occluded vehicles similar to [Ise+18; Tra+18]. Here the RL agent is only responsible for longitudinal control of the automated vehicle assuming that no lane change decisions are needed, and the automated vehicle is following a predefined path by a lateral controller (e.g. [WG08]). In addition to the sensor occlusion, we assume the detected vehicles' position and velocities have noise and therefore make our assumptions closer to real world scenarios. Let $\mathcal{N}_f^t(\mu, \sigma^2)$ denote a truncated Gaussian distribution with support $[\mu - f\sigma, \mu + f\sigma]$. Then, the distance and velocity measurement errors are modeled as truncated Normal distributions $\mathcal{N}_3^t(d, \sigma_\mathrm{d}^2)$ and $\mathcal{N}_3^t(v, \sigma_\mathrm{v}^2)$, respectively, i.e. with a 3σ range around the true value.

Since the agent does not receive all required information for safe and optimal decision-making, the problem is formulated as a POMDP.

Occluded Intersection Model Using Lanelet2 Maps

Using Lanelet2 HD maps [Pog+18], we extract all information about the upcoming conflict zones in the intersections in order to create the observations for the RL agent. Every intersection is mapped inside the Lanelet2 map with

its lanes as polygons where each lane has a unique ID. All lanes that have intersection with the ego lane will be identified as intersecting lanes l_i. Among perceived vehicles from sensor fusion, those that are matched to end in an intersection lane will be extracted as relevant vehicles for the decision-making policy. Then, position and velocity of detected vehicles and for the ego vehicle are mapped to their Lanelet center lines in order to extract vehicle distance and velocity along the Lanelet center line, resulting in one dimensional distance and velocity values for each vehicle.

In addition to the relevant vehicles, information about the occluded areas inside the intersecting lanes are needed for optimal decision-making handling occluded vehicles. For that, instead of using grid-based representation for RL agents (e.g. as proposed in [Ise+18]), we propose modeling the occluded areas by providing the distance between conflict zone and the point where an occlusion begins along the Lanelet center line. This can be done by matching detectable 2D grid-maps from sensor fusion module with the intersection lane l_i and calculate the distance of closest occluded grid to the conflict zone as d_{l_i}. In order to provide information about the lane speed limit (which can be assumed for as reasonable worst-case speed of an occluded vehicle), the lane speed limit from the Lanelet2 map for each lane is represented as v_{l_i}. Therefore, occluded information and speed limit for an intersection lane l_i is provided in our formulation with a ghost vehicle with distance d_{l_i} and velocity v_{l_i}.

3.2.2 Observation Space

We represent the whole situation at an occluded intersection using this observation matrix:

$$o_t = \begin{bmatrix} \overbrace{\quad}^{ego} & \overbrace{\quad\quad\quad}^{vehicles} & & & \overbrace{\quad\quad\quad}^{lanes} & & \\ d_{\text{ego,goal}} & d_1 & \dots & d_N & d_{l_1} & \dots & d_{l_M} \\ v_{\text{ego}} & v_1 & \dots & v_N & v_{l_1} & \dots & v_{l_M} \\ a_{\text{ego}} & d_{\text{ego},1} & \dots & d_{\text{ego},N} & d_{\text{ego},l_1} & \dots & d_{\text{ego},l_M} \end{bmatrix}^T \tag{3.2}$$

where $d_{\text{ego,goal}}$ is the ego vehicle distance to the other side of the intersection (goal), v_{ego} its velocity, and a_{ego} is its acceleration. We assume maximum N vehicles and M intersecting lanes are existing in the scenario. d_i, v_i are distance

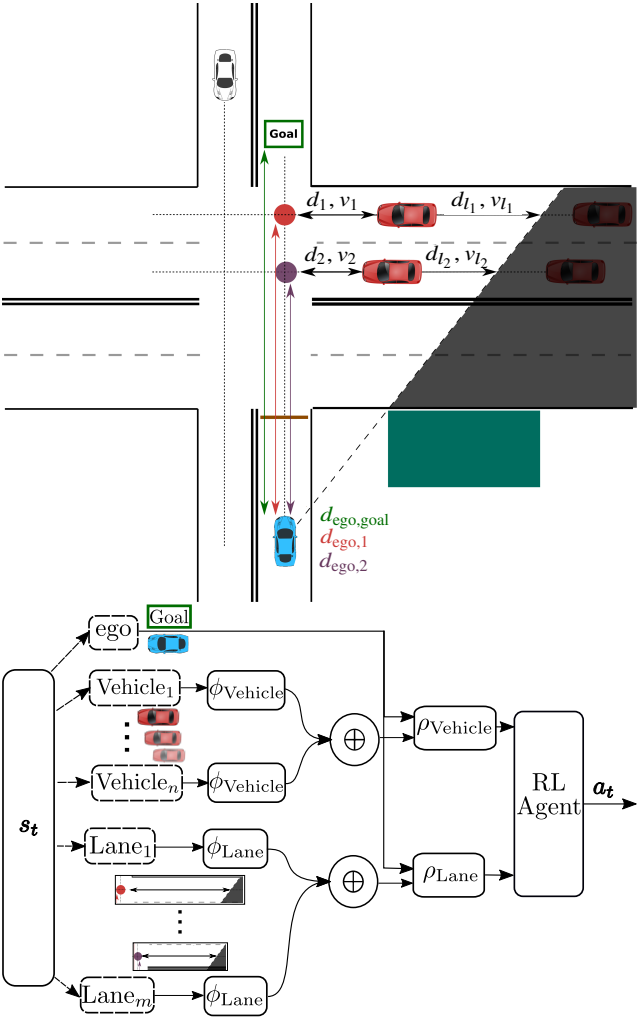


Figure 3.7: **Top:** Parameters in our observation model for intersections with multiple conflict zones and occluded areas (graphic from [Kam+20], ©2020 IEEE). **Bottom:** Proposed Deep Sets based structure for learning scalable policies which are not sensitive to the input (vehicles) order and size.

and velocity of every observable vehicle to their conflict zones and d_{l_i}, v_{l_i} ghost vehicles distance and velocity indicating maximum visible distance and speed limit on every intersecting lane. Finally, $d_{\text{ego},i}$ and d_{ego,l_i} represent the distance of the ego vehicle to the conflict zone with vehicle i and the ghost vehicle at lane l_i respectively. We extend this representation for merging scenarios where we also observe the distance and velocity of the closest front vehicle on the target lane. Fig. 3.7 depicts an overview about our observation model for occluded intersection scenario. Other traffic participants such as pedestrians or cyclists can also be added into this model in order to provide a more generic model. This representation is more compact compared to the grid-based representations used in [Ise+18; KZL19] to represent detected objects and sensor occlusions.

3.2.3 k-Markov State Approximation

In order to make RL agent able to solve this POMDP problem, we provide a k-Markov approximation [Bou+18] of the state as input to the RL neural network in order to enable Q-learning:

$$s_t = \begin{bmatrix} o_t & o_{t-1} & \dots & o_{t-(k-1)} \end{bmatrix} \tag{3.3}$$

By using history of recent observations as RL input state, the neural network and its training process become less complex compared to other implementations for POMDPs like [Tra+18] which utilize Long Short-Term Memory (LSTM) cells [HS97] to incorporate past information. Similarly, in [Mni+15] authors provided the last k=4 observations and actions as the current state of a DQN agent for classic Atari 2600 games. In our experiments we supply the last $k = 5$ observations to the network. Note that the direction of vehicles is specified by their velocity sign and their intention can be estimated from the observation history.

3.2.4 Deep Sets for Scalable Intersection State Representation

The dimension of the observation Equation 3.2 can rapidly grow for complex scenarios when the number of vehicles (N) or intersecting lanes (M) increase.

This is a general issue related to scalability of RL policies according to the policy state representation. In similar works like [Bou+19b] a fixed number of relevant vehicles are provided as the input to the RL, however, this model can result in suboptimal behaviors since such representation can not completely provide all the information about the whole scenario. The other challenge is that different orders of the vehicles in the observation for the exact scenario will have different representations for the RL neural network which will result in inefficiency and ambiguity during training [Hue+19].

This requires to reformulate the interface between the state representation and the RL policy. In order to address this problem, we use Deep Sets [Zah+17; Hue+19] architectures that decouple the network size of machine learning algorithms from the number of input elements.

Deep Sets approach has already been applied to learn a lane change policy with DQN for highway scenarios in [Hue+19]. In this thesis, we apply a Deep Sets architecture for automated navigation at occluded merging and crossing scenarios (Fig 3.7). A representation is calculated for each type of the input element (real vehicles or ghost vehicles) with the ϕ networks. After that, all features for each element type are combined with a permutation invariant operator, we used sum as the permutation operator. Finally, ρ networks extract fixed size features from the combination of each type of the input element with the ego vehicle state.

Action Space

Here we consider actions as jerk commands applied to the automated vehicle similar to [Wer+10; MB19]. This explicitly results in comfortable maneuvers, as we have restricted jerk commands. Therefore, we do not need to include comfort rewards (or discomfort penalties) in the reward function. Therefore, we set possible discrete actions as $\mathcal{A} = \{-1.5\frac{m}{s^3}, 0.0\frac{m}{s^3}, 1.5\frac{m}{s^3}\}$.

3.2.5 Proactive Safety Verification for Safe RL at Intersections

In this section, we explain our proposed approach for worst-case safety verification of RL agents. Similar to the shielding mechanism, we utilize a safety verification module which prevents unsafe decisions to be executed. In order to identify unsafe actions, it considers the worst-case scenario that can happen for the current state. The RL agent is therefore not responsible for the safety anymore and only needs to consider other criteria such as utility, cooperation with other participants and comfort as *soft constraints* in order to reason and learn the optimal policy.

The proactive safety verification module provides safe actions for every state $s \in \mathcal{S}$ as those that are guaranteed to be safe by making sure that a fail-safe action is feasible to be executed if one of the worst-case assumptions is going to happen. We call it a proactive safety verification approach since the worst-case scenario always considers preventing being in an unsafe state in the future by assuming the worst-case scenario to happen.

Worst-Case Scenarios for Proactive Safety Verification

We define a set of worst-case scenarios $\mathcal{H}$ that can happen during automated driving behind a conflict zone:

1. On the intersecting lane l_i, an occluded vehicle is driving with velocity of speed limit on that lane at the closest occluded distance to the conflict zone.
2. On the intersecting lane l_i, an observed vehicle has an estimation error of $3\sigma_\mathrm{d}$ and $3\sigma_\mathrm{v}$ for its distance and velocity and accelerates with $a^{\max}$ to reach speed limit on that lane.
3. On the ego vehicle lane, an observed vehicle in front of the ego vehicle has an estimation error of $3\sigma_\mathrm{d}$ and $3\sigma_\mathrm{v}$ for its distance and velocity respectively. Additionally, it decelerates with $a^{\min}$ to reach zero velocity.

By assumptions in $\mathcal{H}$ we over-approximate the state of an occluded vehicle at intersecting lanes similar to [OML18; TS18] and for detected vehicles as well.

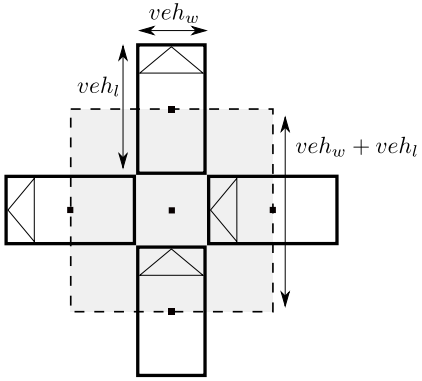


Figure 3.8: Our modelling for the conflict zone on two intersecting lanes. Note that we assume fixed vehicle dimension ($L_{\text{veh}}, W_{\text{veh}}$) for all vehicles in our experiments and therefore $w_{\text{CZ}} = (L_{\text{veh}} + W_{\text{veh}})/2$ is calculated as the width of the conflict zone (graphic from [Kam+20], ©2020 IEEE).

In addition to that, we also over-approximate vehicles' distance and velocity estimation errors in our worst-case assumptions.

Feasibility of Emergency Maneuvers for Safety Verification

Similar to [Nau+19], in order to verify safety of a vehicle state at intersecting conflict zones, we can verify if the vehicle can either safely stop before the conflict zone or leave it before any other vehicle with priority can enter there. Therefore, for every intersecting lane l_i and also the ego vehicle lane inside the observation o, feasibility of executing one of the following two emergency maneuvers is evaluated to verify the observation o as a Proactive Safe State (PSS) according to the worst-case assumptions $\mathcal{H}$:

- Emergency stop maneuver: Ego vehicle is able to stop at a distance bigger than d_{SG} before the conflict zone.
- Emergency leave maneuver: Ego vehicle is able to leave a conflict zone existing in the current observation for duration t_{SG} earlier than the shortest time other vehicles can enter the conflict zone.

In order to verify if the emergency stop maneuver is feasible or not, we calculate the remaining distance of the ego vehicle to the closest conflict zone which is at line l_1 after a full stop emergency maneuver as $d_{\text{ego}}^{\text{FS}}$. Therefore, $d_{\text{ego}}^{\text{FS}} - w_{\text{CZ}}/2 > d_{\text{SG}}$ means that the ego vehicle is able to execute a safe full stop maneuver before the closest conflict zone with safety distance gap of d_{SG} where w_{CZ} is the width of the conflict zone (Fig. 3.8).

For verification of the safe emergency leave maneuver, we assume maximum acceleration for the ego vehicle ($a_{\text{ego}}^{\max}$) and for other vehicles ($a^{\max}$), and calculate the difference between the time that ego vehicle leaves the conflict zone and the time when other vehicle reaches the conflict zone. It should be noted that depending on the vehicles' current distance, velocity and maximum velocity, different equations should be used to find the time to reach or leave the conflict zone based on accelerated or fixed velocity assumptions.

For simplicity, here we only explain the case with zero initial velocity and assume that both ego and other vehicles will not reach their maximum velocity before the time they leave or reach the conflict zone. For this case, the time headway between the time that ego vehicle can leave the conflict zone at intersecting lane l_i and the time that the front vehicle on this lane can reach this conflict zone is calculated as below:

$$t_i^{\text{HW}} = \sqrt{\frac{2(d_{\text{ego},i} + w_{\text{CZ}}/2)}{a_{\text{ego}}^{\max}}} - \sqrt{\frac{2(d_i^{\text{front}} - w_{\text{CZ}}/2)}{a^{\max}}} \tag{3.4}$$

where d_i^{front} is the distance of front vehicle on intersecting lane l_i, $a_{\text{ego}}^{\max}$ maximum ego acceleration, and $a^{\max}$ is maximum front vehicle acceleration.

Therefore, the condition $t_i^{\text{HW}} > t_{\text{SG}}$ verifies that the ego vehicle is feasible to safely leave the conflict zone at this lane.

PSS verification for the given observation o including intersecting lanes l_i can be formulated as:

$$\text{PSS}(o) = \left(d_e^{\text{FS}} - w_{\text{CZ}}/2 > d_{\text{SG}}\right) \bigvee \left(\bigwedge_{l_i \in o} t_i^{\text{HW}} > t_{\text{SG}}\right), \tag{3.5}$$

Note that here we validate safety according to all worst-case assumptions in $\mathcal{H}$ simultaneously, therefore, if any of them or all happen, the situation should remain safe.

Proactive Safety Verification of Actions

We assume the RL agent selects action $a \in \mathcal{A}$ for the history of observations $o \in \mathcal{O}$. In order to verify safety of a, we need to make sure that it results in a proactive safe state even if the worst-case scenario happens. Therefore, we simulate the future state of the ego vehicle by simulating the execution of a and the worst-case state of other vehicles based on o with the worst-case assumptions in $\mathcal{H}$ until the next decision time. We name this simulated worst-case state $\hat{s}^a_{\mathcal{H},o}$.

If $\hat{s}^a_{\mathcal{H},o}$ is a Proactive Safe State (PSS), then action $a \in \mathcal{A}$ is verified as a Proactive Safe Action (PSA):

$$\text{PSA}(o, a) = \begin{cases} \text{True}, & \text{if PSS}(\hat{s}^a_{\mathcal{H},o}), \\ \text{False}, & \text{otherwise.} \end{cases} \tag{3.6}$$

The main advantage of the worst-case proactive safety verification compared to reachability-based [AD14] and prediction-based approaches [INF18] for safety is that it only needs to predict the whole situation for one time step and only for the worst-case scenarios defined in $\mathcal{H}$ instead of all possible maneuvers for all participants in the whole future horizon. Since for each intersecting lane, the proactive safety verification approach only considers the vehicle closest to the conflict zone, it has computation complexity of $O(m)$ where m is the number of intersecting lanes in the scenario. This complexity is much smaller than the safety verification strategy proposed in [INF18] which has complexity of $O(n \times t_{\text{HR}})$ where $n >> m$ is the number of agents and t_{HR} is the time horizon considered for safety verification in that approach.

3.2.6 Safe and Risk-Aware Automated Driving with Distributional Reinforcement Learning

In this section, we explain our final safe and adaptive RL framework for automated driving at occluded intersections with sensor uncertainty. Using the proactive safety verification concept for a candidate action (Equation 3.6), we can have the set of all safe actions for the current observation (o) as A_{PSA}:

$$A_{\text{PSA}}(o) = \{a \in \mathcal{A} | \text{PSA}(o, a) = \text{True}\}, \tag{3.7}$$

Therefore, we make sure that every action from the RL agent is a member of safe action set before executing it. In case the RL action is not safe, one of the emergency maneuvers are executed in order to prevent a collision. However, we aim to minimize the amount of safety interventions in order to maximize comfort during automated driving. For that, we penalize the RL during training for every selection of an action which is not PSA using this reward function:

$$R(o_t, a) = \begin{cases} 1, & \text{if goal reached} \\ -\lambda, & \text{if } a \notin A_{\text{PSA}}(o_t) \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

The proposed framework has two main differences compared to other safe RL frameworks like [Als+17; Mir+18]. Firstly, the emergency action is not necessarily a member of $\mathcal{A}$, therefore the training episode is terminated when a safety interference is required. For the same reason, the replay buffer which includes policy interactions during training keeps the unsafe action generated from the RL instead of the emergency action applied in the environment. Secondly, the agent receives a large penalty of $\lambda > 0$ whenever its action is not a member of A_{PSA} which is also not suggested in [Als+17] and not used in [Mir+18] for reducing the amount of safety interference.

One challenge here is choosing the safety interference penalty λ. Lazarus et al. in [LLK20] applied a similar rewarding scheme for an autopilot system of the aircraft which penalized the RL agent whenever the emergency controller had to be deployed. They showed choosing higher values for λ will encourage more

conservative behavior, whereas faster policies with more interruptions can be expected for lower values of λ.

One may try to find the best value for λ through different experiments, however, the learned policy generates proper behavior only for the environments with state transition probabilities T similar to the training. In other words, when the probability on the occurrence of the worst-case scenarios changes in the environment (due to changes in sensor noise or occlusion severity or drivers' behavior), the RL policy will generate improper actions, since it tries to maximize the expected future return which is approximated according to its training transitions. Moreover, due to the safety verification penalty which considers worst-case assumptions, RL either learns a superconservative policy or a fast policy with too many safety interventions. In our experiments, we learned a balanced policy by a DQN with $\lambda = 1$ as a normal RL baseline.

Risk-aware Policies with Distributional Reinforcement Learning

The main problem with applying normal RL in environments under uncertainty is its risk-neutral characteristic, which cannot distinguish the variance in an action's return and only considers the expected values. Therefore, one risky action with negative tail in its return but higher total expected value is always preferred to a safer (lower variance) action. For that, Tang et al. modeled the return for each action as a normal distribution and optimized an RL policy by learning the return distribution parameters (μ,σ) [TZS20]. However, we believe that the return is a multimodal distribution, and therefore we applied Implicit Quantile Network (IQN) [Dab+18a] which approximates the quantile function that implicitly defines the return distribution. Moreover, utilizing the return distribution also allows expanding risk-neutral policies to risk-sensitive policies by applying distortion risk measures like the Conditional Value-at-Risk (CVaR) [RU+00]:

$$\text{CVaR}_\alpha = \mathbb{E}\left[\mathcal{Z}^\pi | \mathcal{Z}^\pi \leq F^{-1}_{\mathcal{Z}^\pi}(\alpha)\right], \tag{3.9}$$

where $\mathcal{Z}^\pi$ is distribution for the sum of discounted rewards under policy π and $F^{-1}_{\mathcal{Z}^\pi}(\alpha)$ is its quantile function at $\alpha \in [0, 1]$.

Therefore, we train the IQN agent with the same reward function as DQN (Equation (3.8) with $\lambda = 1$) and tune the risk-sensitivity of the policy using α

at execution time without requiring to train multiple policies for different risk levels. In order to leverage the whole solution space over different risk-sensitive policies, α is uniformly sampled with $\alpha \sim U(0, 1)$ at the beginning of each episode during training and applied to the IQN results (Fig. 3.9).

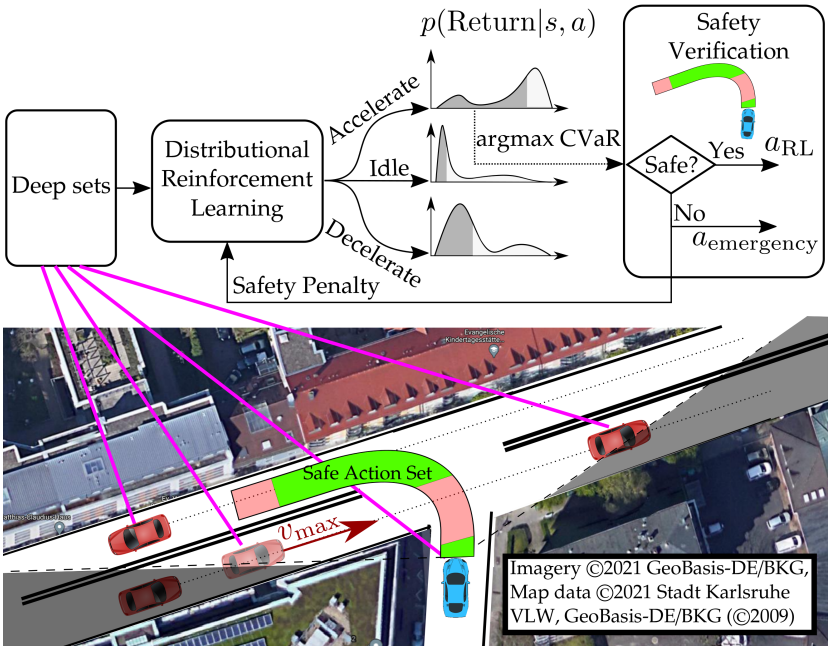


Figure 3.9: The proposed distributional RL framework for safe and comfortable navigation at occluded intersections. Deep sets architecture helps for embedding surrounding vehicles, distributional RL learns risk-aware policies and the safety verification layer filters out unsafe actions (graphic from [Kam+21], ©2021 IEEE).

3.3 Evaluations

In this section, we evaluate the efficiency of the proposed safe and adaptive RL framework for automated driving at intersections under uncertainties. We first explain our simulation environment, baselines, and evaluation metrics. Finally,

we provide the results of experiments and compare the performance of each baseline.

3.3.1 Simulation Scenarios

We simulate automated driving at occluded intersections using our high level simulator which can simulate different randomized scenarios. Fig. 3.10 shows an example scenario generated by our simulator. In order to create a random scenario, the location and size of obstacles (orange boxes) for simulating sensor occlusions are generated randomly causing some vehicles at intersections to be invisible for the ego vehicle. Moreover, every vehicle at the beginning of each scenario will have a random desired velocity sampled from a normal distribution $\mathcal{N}(\mu_{\text{vdis}}, \sigma^2_{\text{vdis}})$ with uniformly sampled parameters $\mu_{\text{vdis}} \in \{6, 9, 12\}(\frac{\text{m}}{\text{s}})$ and $\sigma_{\text{vdis}} \in \{2, 4, 6\}(\frac{\text{m}}{\text{s}})$. Each vehicle drives with the Intelligent Driver Model (IDM) [THH00] controller with maximum acceleration $1\ \frac{\text{m}}{\text{s}^2}$, minimum deceleration $10\ \frac{\text{m}}{\text{s}^2}$ and comfortable deceleration $1.6\ \frac{\text{m}}{\text{s}^2}$. The IDM model has safety distance of 2 m and time headway 1.6 s in order to prevent collision with front vehicles. However, the IDM does not keep distance to the ego vehicle (when it drives into the intersection), thus a collision between a vehicle and the ego vehicle is possible.

At every simulation step a new vehicle is generated in the simulator with probability $p_{\text{new}} \in \{0.1, 0.4, 0.7\}$. Vehicles are cooperative with probability $p_{\text{c}} \in \{0.1, 0.3, 0.7\}$ meaning that they yield to the ego vehicle by setting their desired velocity to zero when the ego vehicle is close to the intersection.

In order to simulate perception noise during simulation, distance and velocity of vehicles are perturbed according to the truncated Normal distributions $\mathcal{N}^t_3(0, \sigma^2_{\text{d}})$ and $\mathcal{N}^t_3(0, \sigma^2_{\text{v}})$ respectively, where $\sigma_{\text{d}} = 1$ m and $\sigma_{\text{v}} = 2\ \frac{\text{m}}{\text{s}}$ are fixed during RL training. For the ego vehicle, the RL policy generates a discrete jerk action from $\mathcal{A} = \{-1.5\frac{\text{m}}{\text{s}^3}, 0\frac{\text{m}}{\text{s}^3}, 1.5\frac{\text{m}}{\text{s}^3}\}$ every 0.3 second. RL policies are trained with over 1000 randomized intersection scenarios providing more than 3×10^5 training steps in total.

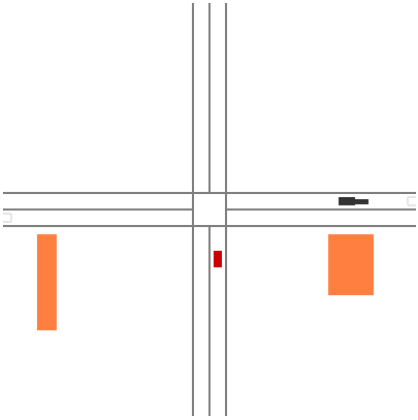


Figure 3.10: Example of an occluded intersection scenario generated by our simulator. Orange obstacles have random size and position causing sensor occlusions. Ego vehicle (red) should pass the intersection without having collision with vehicles driving from left or right side.

3.3.2 Baseline Policies

We evaluate the performance of three baselines in our experiments:

- DQN Agent: A RL agent based on DQN.
- IQN Agent: A distributional RL agent based on IQN.
- Rule-based Agent: A rule-based policy which selects the fastest safe action from $\mathcal{A}$ and in case of emergency selects an emergency action with the lowest jerk.

For the proactive safety verification module which we used for all baseline policies, we set $d_{\text{SG}} = t_{\text{SG}} = 0.5$ in our simulation experiments which resulted in collision free maneuvers during training and evaluation of all baselines.

3.3.3 Evaluation Metrics

After training the RL agents, we evaluate their efficiency using 30 benchmark scenarios with random uncertainty configurations $\sigma_\mathrm{d} \in 0\mathrm{m}, 1\mathrm{m}, 2\mathrm{m}$, $\sigma_\mathrm{v} = 2\sigma_\mathrm{d}$, and $p_\mathrm{c} \in 0.1, 0.4, 0.7$. Note that, no collision with other vehicles happens during evaluations thanks to the proactive safety verification layer. In case of an emergency situation, i.e. $a_\mathrm{RL} \notin \mathcal{A}_\mathrm{PSA}$, an emergency maneuver from the safety layer with maximum allowed emergency jerk limit of 5 $\frac{\mathrm{m}}{\mathrm{s}^3}$ is sent to the controller instead of the unsafe RL action. In order to compare the effect of interference applied for each policy, we define a metric for measuring the amount of applied emergency interference:

$$J_\mathrm{Interference} = \frac{\Sigma a_\mathrm{emg}^2}{N_\mathrm{episodes}}, \tag{3.10}$$

where a_emg is the emergency jerk command applied to replace the unsafe action and N_episodes is the total number of evaluation episodes. Note that this metric measures the average value for the sum of all squared jerk values applied to the vehicle during one episode identifying the amount of discomfort caused by each policy.

In addition to the amount of emergency interference for each policy, we also measure the average crossing time as the metric for utility. An optimal policy which maximizes the total reward defined in Equation 3.8 should minimize the emergency interference penalties and also minimize the duration of finishing the episode in order to receive the final finishing reward. Therefore, a policy that only has good performance in one of the two metrics is either too much conservative or too much uncomfortable.

3.3.4 Results

Fig. 3.11 compares the speed and comfort of each baseline based on the recorded measurements when they control the automated vehicle in the benchmark simulation episodes. The IQN agent shows an adaptive behavior based on the α percentile. The policy becomes faster but less comfortable on average by increasing α. For lower α percentile, it has less absolute jerk and therefore results in more comfortable driving.

Environment Configuration (σ_d, p_c)		(0, 0.3)		(2, 0.3)		(2, 0.7)	
Policy	Metric for optimal α	Time	$J_{\text{Int.}}$	Time	$J_{\text{Int.}}$	Time	$J_{\text{Int.}}$
IQN	Speed	9.8	25.0	8.8	43.0	9.9	33.3
	Comfort	11.8	**10.0**	11.1	25.5	12.2	**18.8**
DQN	—	20.0	10.8	21.4	**14.2**	21.3	20.1
Rule-based	—	**7.7**	83.6	**7.8**	92.3	**7.7**	103.1

Table 3.1: Evaluation of the IQN with best α for different metrics and comparison with other baselines for different environment configurations.

Distribution of recorded vehicle jerk values for IQN with different α values and other baselines is depicted in Fig. 3.12. The IQN with lower α has less jerk values outside the RL jerk limit which indicates less emergency interference and more comfortable maneuvers. DQN agent shows the most conservative behavior, but it also has the highest crossing time proving the fact that it fails to optimize both comfort and utility at the same time.

On the other hand, the rule-based policy has the highest amount of emergency interference as a risk neutral policy resulting in a wider jerk distribution and uncomfortable maneuvers.

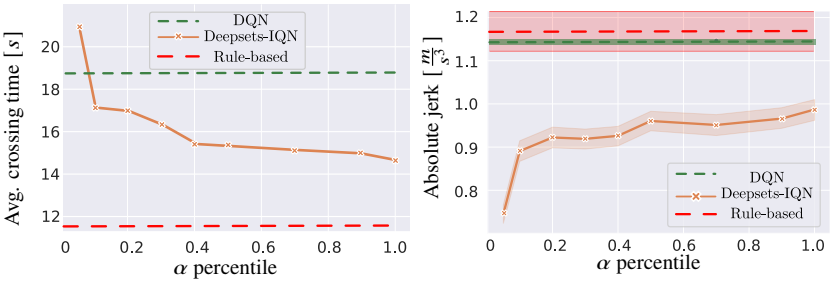


Figure 3.11: Comparing performance of each baseline on benchmark scenarios. **Left:** Average crossing time for each baseline. The IQN policy becomes faster and less conservative with higher α. The rule-based policy is the fastest and DQN is the slowest policy. Note that all policies are completely safe without any collision due to the safety verification layer. **Right:** Average of vehicle absolute jerk and its confidence interval for each baseline. By increasing α, IQN policy becomes less conservative (higher jerk) (graphics from [Kam+21], ©2021 IEEE).

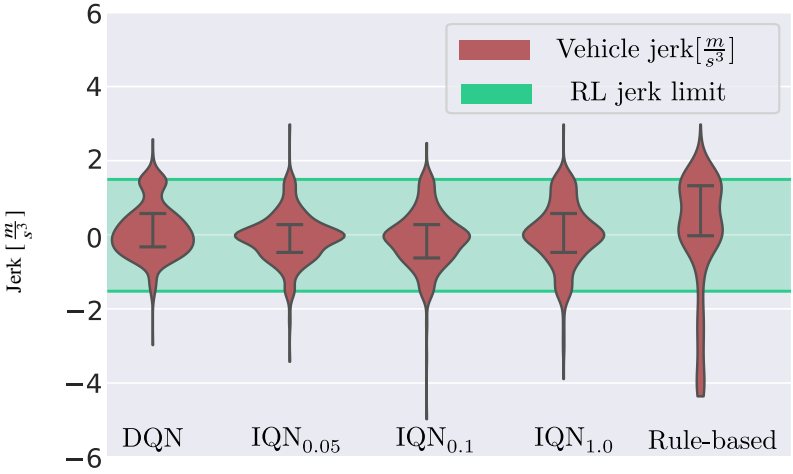


Figure 3.12: Distribution of vehicle jerk for each baseline on benchmark scenarios. The green interval shows the RL jerk limits. Jerk values out of this interval are due to emergency interference and indicate uncomfortable driving (graphic from [Kam+21], ©2021 IEEE).

We can conclude that the IQN agent can learn a family of adaptive policies which are not as highly conservative as DQN and also not as risk-neutral and uncomfortable as the rule-based policy. The same conclusions can be drawn from Table 3.1 where we recorded the average crossing time and $J_{\text{Interference}}$ for all policies at different environment configurations. Here we only consider two IQN sub-policies, the ones with the fastest behavior and the ones with the lowest interference cost (most comfortable ones). We evaluated the policy in 10 uniform samples of α and selected the best policy according to each metric. Note that finding the best α value could be done in a more automated approach but here we only want to compare the best behavior of the IQN agent with other baselines and examine how much other metrics are sacrificed when the policy performs as the best in only one of metrics. In all three configurations considered in Table 3.1, the rule-based policy is the fastest one, however, it is also the worst policy in terms of comfort. The DQN policy, on the other hand, always has low interference cost, but drives more than two times slower

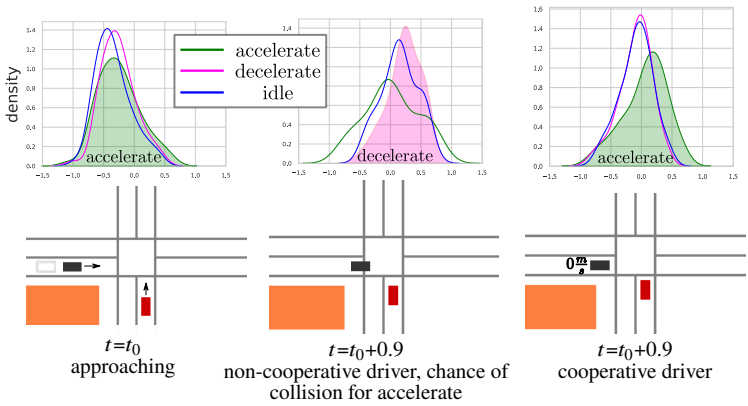


Figure 3.13: **Left:** Learned return distributions for two scenarios with similar initial situation. Ego vehicle (red) follows the action with the highest CVaR that is shown with filled distribution. Gray ghost vehicle indicates maximum visible distance limited due to orange obstacles. **Middle:** Black vehicle is non-cooperative causing more negative return chance for the *accelerate* action. **Right:** Black vehicle is cooperative causing higher CVaR for *accelerate* action (graphics from [Kam+21], ©2021 IEEE).

than the others. The IQN policy trades-off between average crossing time and interference cost by showing relatively fast behavior which requires 3.6 to 8 times lower interference than the rule-based policy depending on the environment configuration.

IQN Policy Reactions to Drivers' Intentions

Since the RL policies receive history of the last 5 recent observations as their input, they can predict the intention of other drivers and generate optimal actions based on that. Fig. 3.13 shows examples of the return distributions learned by the IQN agent for two similar scenarios with the only difference of having cooperative (yielding to the ego vehicle) and non-cooperative drivers. As can be seen, when the other vehicle is cooperative and reduces its velocity, the IQN policy generates higher return for the $a = 1.5 \frac{\mathrm{m}}{\mathrm{s}^3}$ action allowing the ego vehicle to enter into the intersection.

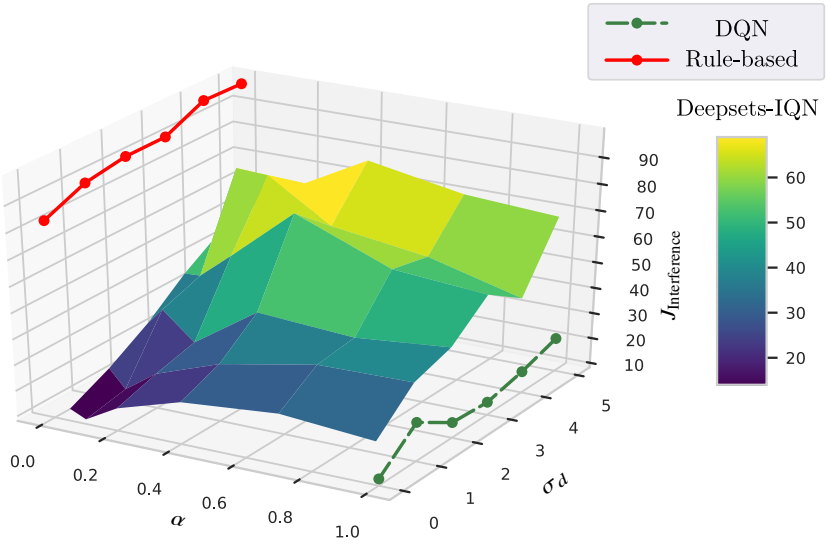


Figure 3.14: Safety interference cost applied to the IQN policy for different environment noise levels and its comparison with baselines (graphic from [Kam+21], ©2021 IEEE).

Evaluation under Higher Perception Uncertainties

In addition to the experiments where we applied training noise levels during testing, we studied the sensitivity of each policy to higher amounts of noise in the environment. The RL and rule-based agents were evaluated for 5 different noise levels $\sigma_d \in \{0, 1, 2, 3, 4, 5\}$ (in meters) and $\sigma_v = 2\sigma_d$ (in $\frac{\text{m}}{\text{s}}$). The results are depicted in Fig. 3.14. As the diagram reveals, the IQN policy has lower interference cost for lower α and also lower noise levels. For higher noise levels, it requires more interference which can be reduced by decreasing α. The rule-based policy always has high cost and DQN has low cost independent of the amount of noise existing in the environment. Here DQN seems to be a comfortable policy, however it is the worst policy in case of utility as discussed before.

3.4 Conclusions

In this section we addressed safety as one of the most important challenges of decision-making problems in real-world. Focusing on automated driving, we acknowledge that perception modules cannot provide all required information due to sensor noise, occlusions, and unknown drivers' intentions, to name a few. Moreover, RL solutions usually tolerate collisions without any guarantee to perform completely safe. Therefore, we utilized a proactive safety verification approach to validate the safety of actions with the goal of preventing unsafe situations in critical applications. In order to minimize uncomfortable safety interference, we used RL to learn policies that are penalized for resulting in states where an emergency maneuver is necessary. We showed how the IQN agent can mitigate the conservative behavior existing in DQN policies by learning return distributions which provide policies that can generate adaptive behaviors after training in order to become more comfortable or less conservative. The proactive safety layer however prevents hazardous outcomes in case any worst-case scenario for a less conservative policy happens. Moreover, the learned distributional IQN policy requires less safety interference in noisy environments compared to a rule-based safe policy even when the noise level is higher than the training configurations.

4 Interactive and Transferable RL

One of the important uncertainties for automated driving in real-world is unknown intention of other drivers. Even for human drivers at specific scenarios, it can be challenging to correctly understand the intention of other drivers and make the best decision. For automated vehicles, even with the best perception modules which can estimate position and velocity of surrounding vehicles with good accuracy, intention of the drivers cannot be easily predicted. This is inherently a problem when humans are part of the system, as they generally act as non-stationary and partially observable elements in an environment [Xie+20].

On the other hand, for making optimal decisions, the intention of surrounding vehicles is a crucial factor for the behavior generation policy. Therefore, a history of observations and interactions with the other vehicles is needed in order to correctly estimate their intentions. For that, a human driver or the behavior generation module in the automated driving vehicle may need to first gather enough information in order to observe the behavior of other drivers and then make the best decision based on that. More specifically, an efficient interactive decision-making policy should consider these factors:

- Other drivers can have different desires and intentions such as being cooperative, non-cooperative, or inattention, which are not directly observable by the perception system.
- Since the drivers' intention is not observable, in the worst-case, the policy needs to perform data-gathering behaviors in order to receive more information required to estimate these unknown parameters.
- The policy can influence cooperative drivers by interacting with them in a way that they understand ego vehicle's intention and cooperate with it (if they are cooperative).

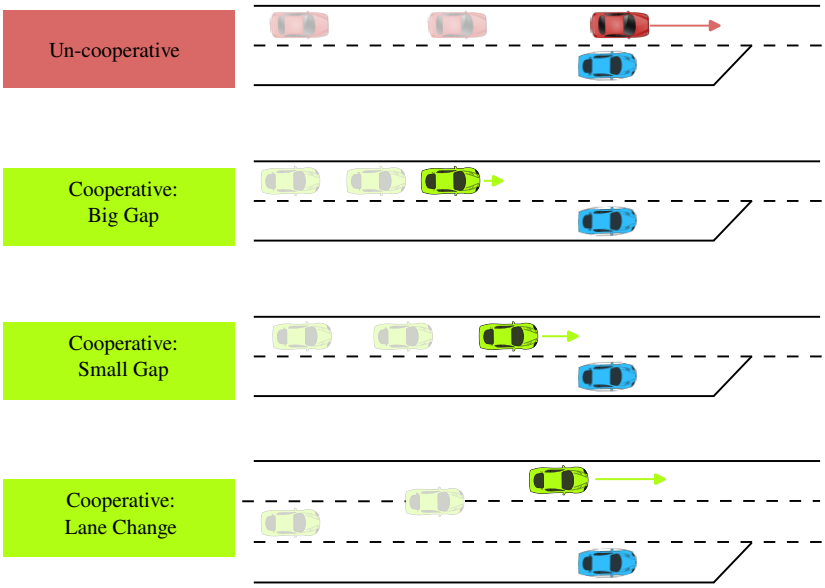


Figure 4.1: Multiple interactive merging scenarios where cooperative drivers with same intention have different history of observations.

- In case other drivers' intention is wrongly estimated (either by the policy or an external intention estimator), a safety verification module should prevent undesired outcomes and maybe execute emergency actions.

Another important factor regarding RL under partial observation is poor generalization to new environments with different transition dynamics [EL21]. In critical applications like automated driving, factors such as local infrastructures, rules or different driving styles and cultures may lead to varying behaviors of human-controlled agents in new environments which may lead to unsuccessful policy transfer [Cen+19].

Fig. 4.1 shows three different behaviors from cooperative drivers in an interactive merging scenario. Let assume that in our training environment, all cooperative drivers yield to our automated vehicle by reducing their velocity and opening large merging gaps for us. Now assume that we want to transfer the trained policy into a new environment with lower speed limit where cooperate vehicles

open a smaller merging gap for our vehicle since they have relatively smaller velocities compared with the cooperative drivers in the training environment. It is also possible that a cooperative vehicle does not decelerate at all and instead changes its lane in a different transfer environment. In such cases, the automated vehicle observes *novel cooperative behaviors* which was not demonstrated in the training environment, and therefore it can result into suboptimal decision-making.

All of these factors can affect low-level transition dynamics of the underlying MDP, making it complex to be solved by a *generalist* or a family of *specialist* policies trained for all possible environments and scenarios at the same time or separately [PSK20]. The generalist RL requires to be trained on all possible cooperative maneuvers. In such problem formulations based on rich observations, similar states can be mapped to different observations which can increase the complexity of underlying transition dynamics. Therefore, generalist RL requires complex learning structures and huge amount of training data. On the other hand, the specialist RL can be trained in a more straightforward way, while it cannot be transferred into new environments with new cooperative maneuvers.

We believe that such problem can be modeled as Hidden Parameter Block Markov Decision Process (HiP-BMDP) where unknown intention of drivers are modeled by hidden parameters. This helps to model drivers' intention prediction and decision-making as two separate tasks where supervised learning can be used for the first one and RL can be used for the second task. The main advantages of this modular framework are:

- More abstract state representation, smaller neural network and therefore more sample-efficient RL.
- When transferring to new environments, only the intention prediction module needs to be updated or replaced. Depending on the extent of the differences between the transfer and training environments, the RL policy can still be used, or it may also need updating but with less training complexity.

In the remaining parts of this chapter, we first provide a literature review about recent papers regarding interactive automated driving and transferable RL policies. Then we explain our solution for interactive RL under intention uncertainty based on HiP-BMDP. We introduce a new method called Hidden

Parameter Adversarial Perturbation (HiP-AP) RL, which is able to handle uncertain intention predictions which may happen in new environments with different dynamics.

This idea of uncertainty-aware transfer was first introduced in a method that also modeled the problem as a HiP-MDP and assumed that the true value of the hidden parameter would be revealed in hindsight [Pon+22]. We apply a similar approach and model the situation where the hidden parameter estimator is not confident (especially at the beginning of interaction in a new environment), requiring more exploration to be done by the policy. This strategy is helpful in both offline RL and sim-to-real transfer learning contexts where the agent could have access to the true hidden parameters either by post-processing of the interactions (similar to [Pon+22]) or receiving that from the simulator.

Finally, we evaluate our solution and compare its performance with a classical history-based RL which addresses the partial observation issue by providing a long history of observations as the input to the RL agent. We evaluate our approach in the interactive merging environment and provide different results comparing the efficiency of our approach with the history-based RL agent. We will explain the main drawback of history-based RL methods which is their inability to transfer to new environments with different traffic dynamics, and show the ability of our solution (HiP-AP RL) to efficiently transfer to those new environments.

4.1 Related Work

4.1.1 Interactive Merging as Partially Observable MDP

Considering the unknown route of each driver at an un-signalized intersection as not observable parameters in a POMDP setting, authors in [Hub+17] proposed a POMDP solver which utilizes belief estimations for predicting each driver's route based on the generated acceleration from the predefined transition model of each possible route. Although route uncertainty is well formulated in this work, authors made specific assumptions about constant reference velocity and constant interaction based acceleration which makes their approach not generic to be used in different configurations. Moreover, for automated driving in

merging scenarios, the unknown intention is more related to drivers' longitudinal velocity rather than their global route which makes it more challenging compared with intersection scenarios.

In a newer approach, authors in [Hub+18] addressed unknown intention of other drivers about being cooperative or non-cooperative in a merging scenario. Authors deployed a logistic regression classifier called *yield classifier* which is trained on real-world data and estimates the probability for cooperative behavior of a human driver. This probability will be turned into a binary value via a threshold which can be tuned for more or less aggressive merging behavior.

In [Taş22] authors applied a POMDP solver which is encouraged through information rewards to execute dedicated actions for revealing driving intentions of other traffic participants. By identifying their driving intentions the planner is able to plan execute more optimal trajectories achieving higher rewards.

Some researchers proposed to solve similar POMDP problems using RL agents with specific belief estimators for the hidden parameters (in our scenario the unknown intention of other drivers). Authors in [Bou+19b] tackled unknown intention of other drivers in the cooperative merging scenario by maintaining a belief over the level of cooperation of other drivers and used it as extra input for the RL policy. Although this approach enables reasoning about the intention of other drivers, its main drawback is that it simplifies the transition dynamics of the vehicles with different intentions which makes the approach not generalized and valid in new environments with different transition models.

4.1.2 K-Markov Approximation for Interactive Reinforcement Learning

As explained in Section 3.2.3, k-Markov Approximation is one way to tackle the unknown intention challenge by providing the recent history of last interactions with each driver included in the recent k observation history:

$$(o_{t-k-1}, a_{t-k-1}), ..., (o_{t-1}, a_{t-1}), (o_t, a_t), \tag{4.1}$$

where k is the history length ([Bou+19a; Kam+21]).

One question here is how long should be the history length of the observation matrix. Providing longer histories can include more information and tackle the problem of partial observation from the environment making the policy to have better reasoning about the intention of each driver and also the whole scenario. On the other hand, longer history can have adverse effect on the training procedure since it necessitates bigger or more complex learning architectures [KRL21; LM19; Hue+19]. We will later answer this question in the evaluations by comparing different RL agents trained with different history lengths for an interactive automated merging scenario.

Moreover, high-dimensional observations can result in complex underlying transition functions, where different states are mapped to very similar observations, requiring large amounts of training data to cover all possible behaviors and observations [Sod+22; Du+19]. Finally, a history-based RL policy may overfit to the training environment and have poor performance when transferred in a new environment with novel traffic dynamics (such as drivers' new strategies of cooperation with the ego vehicle).

4.1.3 Unknown Intentions as Latent Variables

In [Xie+20] authors proposed to consider unknown strategies of another opponent as latent variables and proposed to learn an RL policy which is conditioned on these latent variables and simultaneously learns these latent variables from the previous interactions with the other participants (opponents). They utilized an encoder-decoder learning architecture for predicting future interactions of the participants based on these latent variables and therefore formulated the total loss function in order to minimize the prediction error for future interactions of the opponent agent. Therefore, the encoder is aimed to learn latent variables representing opponent's strategy which will affect its future interactions.

4.1.4 Generalized Reinforcement Learning

Both k-Markov approximation and latent variable encoding of unknown human intentions can be good solutions to tackle interactive RL policies for a fixed environment, but not able to be transferred in new environments with new

transition dynamics. In this section we briefly explain some previous works which tackle generalization and ability to transfer to new environments for RL policies.

Generalized RL for different environments can be framed as the problem of transferring a policy from one or more source MDPs to one or more unseen target MDPs. Prior works have explored various approaches to this problem, which we briefly review in the following. Generally, they can be divided into two categories: those that meta-learn few-shot transferable, i.e., easily adaptable policies [FAL17; Rot+18; Rak+19] and those that directly learn zero-shot transferable policies [PBS15; Gel+19; Teh+17; Yu+20]. Since our goal is to develop ready-to-use policies for automated driving that are generalized for different environments, our discussion of related works focuses on zero-shot policy transfer.

Zero-Shot Policy Transfer with Representation Learning

Prior work has explored approaches that learn state or task representations to ease learning in subsequent transfer tasks. In [Hig+17], families of MDPs with varying observations, but structurally similar transitions and rewards are considered. At the first stage, they learn a Variational Auto-Encoder (VAE) that maps observations to disentangled features by recording random trajectories in a sufficient number of different environments. The subsequent RL step uses these disentangled features as inputs to the policy. They demonstrate that the method improves zero-shot policy transfer performance, like in the case of parameter changes or sim-to-real transfer. A similar approach is presented in [Yan+19]. Other methods have avoided the use of a reconstruction loss by leveraging bi-simulation metrics [Zha+20a], or adversarial losses [Li+21].

Policy Transfer with Latent-Variable Models

A large body of related-work has focused on modelling the system dynamics using latent-variable models. In [PSK18], variational inference is used to determine the latent variables of the transition dynamics. The inferred model is subsequently used for model-predictive control.

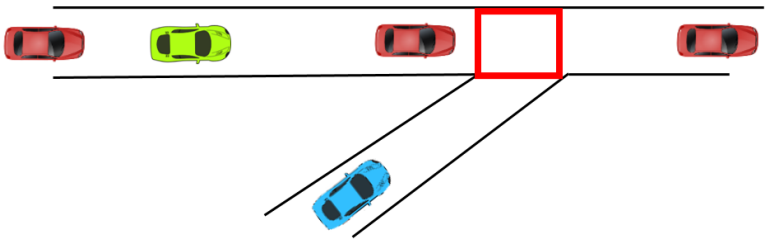


Figure 4.2: Interactive automated merging scenario under intention uncertainty for other drivers. The blue vehicle represents the ego vehicle which attempts to merge between the other vehicles. The red rectangle depicts the conflict zone used as the reference to calculate vehicles distance on their route. Cooperative car is colored in green and the other cars are colored in red.

In [Yao+18], the tasks are modeled as instances of a Hidden Parameter Markov Decision Process (HiP-MDP) [DK16]. It is assumed that an optimal policy is available for every task. The authors propose to infer the hidden parameters using expectation-propagation [Her+16] and to distill the task-specific policies into a global policy that is directly conditioned on the hidden parameters of the MDP. This approach assumes that the policy will be transferred to new environments, where some unknown parameters affect the transition function. The policy uses a Bayesian Neural Network (BNN) to estimate the hidden parameters. Based on this estimation, the policy will be adapted to the new task by comparing the transitions from the current environment with observed transitions from previous instances. The policy also contains global parameters which are trained before transfer. A drawback of this work, which is addressed via uncertainty-aware transfer, is the lack of a mechanism for task identification by the policy when it faces a new environment.

4.2 Problem Formulation

In this section we explain our problem formulation to address unknown intention of other drivers in the automated merging scenario where the ego vehicle needs

to interact with other drivers in order to safely and efficiently merge into a merging lane (Fig 4.2).

4.2.1 Drivers' Intention and Behavior Model

Similar to [Hub+18], we model the intention of a driver with index i in our observation as a Boolean parameter which can have one of the two possible values:

$$\theta_i \in \{\text{cooperative, non-cooperative}\} \tag{4.2}$$

A cooperative driver will only reduce its velocity and open a merging gap for the ego vehicle if it observes clear merging intention from the ego vehicle (e.g. driving with non-negative acceleration towards the merging area). On the other hand, a non-cooperative driver never reacts to the ego vehicle merging intention and continues with its original strategy (e.g. drives with constant velocity) even if the ego vehicle is clearly driving to merge in front of this vehicle.

We assume that drivers' intention is fixed during one episode, meaning cooperative drivers never stop their cooperative yielding maneuver when they observe a merging attempt from our vehicle. Also, a non-cooperative driver never starts a yielding maneuver even if it observes a clear merging attempt from our vehicle. In addition to these characteristics in our drivers' intention model which are similar to the model used in [Hub+18], we assume that multiple drivers with same intention can follow different maneuvers and with different dynamics. In other words, we assume that, based on their intention, drivers follow a specific policy from a family of policies regarding that intention:

$$\pi_i \in \Pi_{\theta_i}, \tag{4.3}$$

Cooperative drivers which all have same intention parameter (θ_i = cooperative) may follow different cooperative policies such as small gap, big gap or lane change strategies (see Fig. 4.1). In the same way, non-cooperative drivers may have different behaviors such as constant velocity driving, acceleration, or even change their lane to the ego vehicle merging lane. Therefore, in this dissertation

we can tackle a wider range of scenarios and not assume a fixed policy and dynamics for drivers with specific intentions.

4.2.2 Novel Drivers Behavior in the Transfer Environment

In order to tackle a more realistic interactive merging scenario, we assume that not all the policies for a specific intention are available during training and a subset of them are demonstrated to the RL agent during training. Instead of training generalist policies that are required to train for all possible scenarios and behaviors, we propose generalized RL agents that do not necessarily need to observe all possible behaviors during training. Therefore, the policies are able to transfer to a new environment where drivers follow new policies for their intentions.

4.2.3 State Space

The state space summarizes all required information for optimal decision-making in the merging scenario. Fig. 4.3 depicts an overview of this scenario and the parameters used for the state representation. For more general representation, position and velocity of the ego vehicle and all N vehicles are transformed into their distances and velocities along their routes. The beginning edge of conflict zone is the origin for vehicles distance on their route meaning the vehicles behind the merging point have positive distances. Finally, the whole state is represented as below:

$$s = \begin{bmatrix} \overbrace{ego}^{} & \overbrace{vehicles}^{} \\ d_{\text{ego}} & d_1 \ \ldots \ d_N \\ v_{\text{ego}} & v_1 \ \ldots \ v_N \\ a_{\text{ego}} & \theta_1 \ \ldots \ \theta_N \end{bmatrix} \tag{4.4}$$

Where d_{ego}, v_{ego}, and a_{ego} are the ego vehicle distance, velocity and acceleration along their routes. Similarly, d_i, v_i are the distance and velocity of other vehicles, and θ_i is their intention parameter.

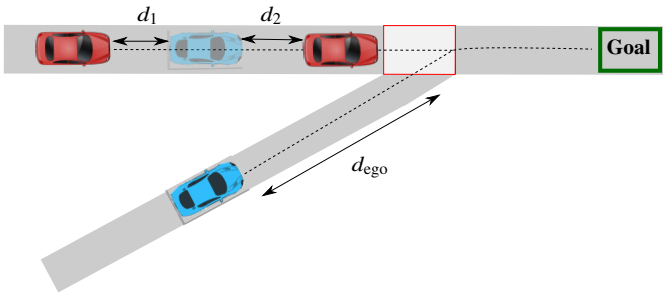


Figure 4.3: State representation for the interactive merging scenario. Ego vehicle (Blue) should merge between other vehicles (red) and finally reach the goal. The intention of other vehicles θ_i are not directly observable (graphic from [KRL21], ©2021 IEEE).

4.2.4 Observation Model

We assume drivers' intentions (θ_i) are hidden parts of the state which cannot be observed by the policy, however, they can be revealed after some proper information-gathering interactions between the ego vehicle and the drivers. For simplicity, we assume other parts of the state (vehicles' distance and velocity) can directly be observed from the environment without any measurement noise:

$$o = (o_{\text{ego}}, o_1, ..., o_N), \tag{4.5}$$

where $o_{\text{ego}} = (d_{\text{ego}}, v_{\text{ego}}, a_{\text{ego}})$ and $o_i = (d_i, v_i)$.

4.3 Approach

In order to address intention uncertainty for interactive automated driving, we model the problem as a Hidden Parameter Block Markov Decision Process (HiP-BMDP) [Zha+20a]. We model unknown intentions of drivers by hidden parameters in our framework which can be estimated via intention estimators as separate modules outside RL. Such hidden parameter estimators can be trained in parallel with the policy by incremental supervised learning. Drivers' intentions can also be estimated by available trajectory predictors based on

machine learning [Sal+20], probabilistic graphical models [DDL17], or any other form.

In case the hidden parameter predictor has a high uncertainty (e.g., due to lack of observations from the new environment for updating the model), the proposed policy is able to perform optimal information gathering actions to obtain more observations and reduce the prediction uncertainty due to the adversarial perturbations applied on it during RL training.

HiP-BMDP is defined as a combination of Hidden Parameter Markov Decision Process (HiP-MDP) [DK16] and Block Markov Decision Process (BMDP). We first explain HiP-MDP part of the model and later provide a complete model definition based on HiP-BMDP.

4.3.1 Modelling Unknown Driver Intentions with Hidden Parameters

A POMDP problem can generally be formulated as a HiP-MDP considering these two assumptions:

- There exists a bounded number of latent variables such that, if they are known, the entire dynamics of the system would be specified.
- These latent variables remain fixed during one episode (until task is finished), but can change for a new episode (task).

These assumptions fit very well to our interactive automated merging problem:

- Considering drivers' intention as unknown latent variable, the whole dynamics of the system would be specified if the drivers' intention becomes revealed and therefore the problem can be modeled as an MDP.
- Drivers' intention remains fixed during one episode, meaning that a cooperative vehicle reacts to our vehicle merging attempt and never changes its behavior.

Therefore, the problem of partial observation due to unknown drivers' intention in automated driving can be reformulated as a family of HiP-MDP problems

which are conditioned on the fixed hidden parameters that are in our case the cooperative intentions of all surrounding vehicles:

$$\theta = (\theta_1, \theta_2, ..., \theta_N) \tag{4.6}$$

It should be noted that the size of hidden parameter vector (N) is bounded since we only provide a fixed number of surrounding vehicles as the input state representation for the policy.

This model helps us to break the problem of planning under intention uncertainty into two smaller tasks: 1) Hidden parameter estimation and 2) Decision-making based on predicted parameters.

4.3.2 Final Model based on Hidden Parameter Block MDPs

Now we explain our final modelling of the interactive merging problem using HiP-BMDP. As explained in Section 2.3, HiP-BMDP has an extra attribute in addition to HiP-MDP which is the state block assumption for different observations that refer to a same state. In our problem, we assume different observation dynamics from the drivers with the same intentions can be modeled with same states. In other words, we assume drivers with same intentions (e.g. being cooperative) have same states but since they may have different driving styles (e.g. being cooperative by fast deceleration or keeping constant distance to the ego vehicle), they can have different observations (Fig. 4.4).

In summary, the main HiP-BMDP modelling for the problem of interactive automated driving under intention uncertainty with different behavior dynamics can be described as below:

- State Set $\mathcal{S}$: The state that fully describes the current situation for an optimal policy, consisting of the location, velocity and intention of every participant, as well as the ego vehicle (as explained in Sec. 4.4).
- Action Set $\mathcal{A}$: Set of possible actions the ego vehicle can select. In our case it will be jerk commands that are selected from the policy to control the acceleration of the automated vehicle.
- Task Parameter set Θ: Different combinations of intentions that all drivers can have. This is assumed to be hidden and not observable.

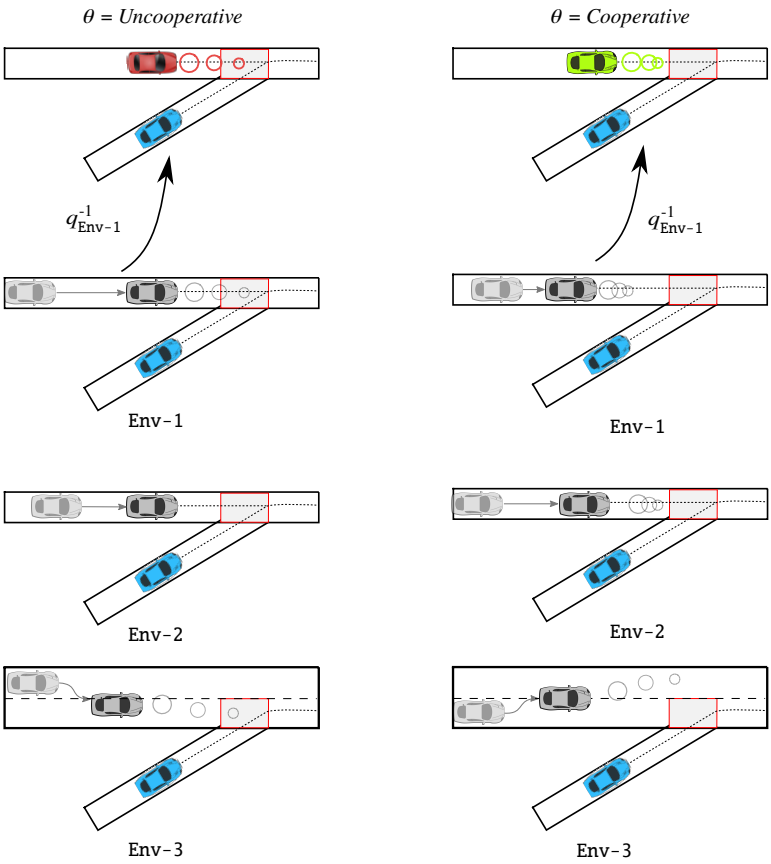


Figure 4.4: In real-world, drivers with same intentions (cooperative or non-cooperative) may follow different policies which can result in different observations, while they all have same hidden parameters. Shaded cars depict previous positions of the drivers and circles depict their intended future positions.

- Observable Space $\mathcal{X}$: The part of the state that can directly be observed. This includes the location and velocity of every participant, as well as the ego vehicle.

- Emission Mapping function q: The stochastic mapping that emits the sequence of observations $q(x|s)$, in our case, the sequence of vehicles' positions and velocities given their intention.

4.3.3 Decoupling Intention Prediction and Decision-Making

So far we modeled the problem of interactive merging under intention uncertainty based on HiP-BMDP. Different RL models can be used in order to solve HiP-BMDP problems. In [Xie+20] hidden parameters are implicitly learned as part of the learning framework where observations are provided for part of the RL network which predicts the current task parameters as latent variables by reconstruction. Similar to *k-Markov Approximation* [Bou+19a], it is assumed that history of k recent observations is enough in order to identify the current task parameters (drivers' intentions in our scenario). However, this methodology has two main drawbacks:

- Since the policy is conditioned on the recent observation history, we require a more complex and less sample efficient training architecture.
- Once the policy is trained, it cannot be generalized into new environments where drivers have new maneuvers with novel dynamics.

Therefore, instead of providing a history of recent k observations, we propose to utilize an external task estimator which in our case predicts the intention of drivers based on their histories and only the predicted state from this module is provided as the input to the RL agent:

$$\hat{s}_t = (o_t, q^{-1}(o_t, o_{t-1}, ..., o_{t-k+1})) \tag{4.7}$$

Where q^{-1} is the function which estimates the value of hidden parameters in the state from the recent history of observations from the environment. This function can be implemented as a classifier which predicts the probability of different possible values for the existing hidden parameters. In our application, this function provides estimates of the drivers' intention:

$$q^{-1}(o_t, o_{t-1}, ..., o_{t-k+1}) = (\hat{\theta_1}, \hat{\theta_2}, ..., \hat{\theta_N}) \tag{4.8}$$

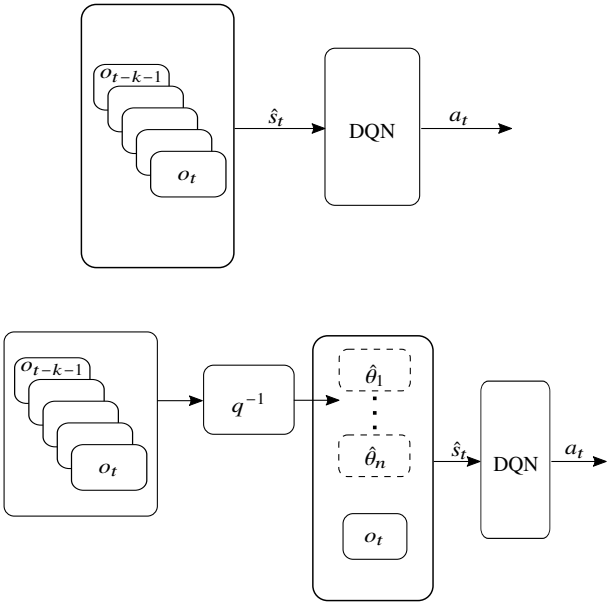


Figure 4.5: **Top:** Implicit intention estimation in k-Markov RL models. **Bottom:** External hidden parameter estimation in HiP-BMDP RL.

Fig. 4.5 depicts the difference between the implicit intention estimation and our proposal which uses external intention estimators. As it is visible, the main difference is in the input of each RL agent while they both have a same output (action) interface.

Lernszenarien

More specifically, we utilize an external intention predictor module (q^{-1}) which explicitly predicts the cooperative intention based on previous interactions with each vehicle. Using an external drivers' intention prediction has these major benefits:

- **Generalization:** The RL does not require to be trained in all environments in order to observe all possible behaviors for different intentions. Instead, the policy can only be trained in one of the environments, by having access to the true states in that environment, and then be transferred to new environments by only updating its intention predictor module (q^{-1}).

- **Sample Efficiency** There is no need to provide history of k recent observations as the RL agent input and therefore the RL neural network structure becomes less complex and more efficient to train.
- **Modular Structure** Due to the decoupling between RL and predictions, we can benefit from several classical and efficient prediction models available in the literature to use as behavior predictor. Moreover, maintenance of both modules can be done separately.

The main drawback of this structure is the potential loss of benefits that come with an end-to-end solution. In simpler tasks, this approach may actually increase the complexity of learning since it requires training two separate models independently. Additionally, the policy must account for variations and uncertainties in the predicted intentions in new environments. We will address this issue in the next section.

4.3.4 Policy Transfer with Hidden Parameter Adversarial Perturbation (HiP-AP) RL

We assume that drivers have one of cooperate and non-cooperative intentions (Eq. 4.2.1) but follow different policies at different environments (Eq. 4.2.1). By modelling the problem with HiP-BMDP, we assume that high level states of drivers with same intentions across different environments are similar and only their emission functions change depending on the environment.

Therefore, we propose to train the RL policy with direct access to the true hidden variables in the source environment which makes it transferable between related HiP-MDPs, that only differ in the emission functions (q'). The true hidden parameters can come from the assigned parameters in the simulator or by post-processing and hindsight extraction [Pon+22] from recorded datasets and identifying cooperative and non-cooperative drivers based on their final behavior. However, when deploying the trained policy in a test environment, an external hidden parameter estimator is required to predict the existing latent variables for the policy. It can happen that by transferring the policy to a new environment with completely new emission functions (q'), the hidden parameter estimator have high uncertainty and therefore our assumption about having access to the true hidden parameter becomes invalid.

We propose to make the policy robust to uncertain hidden parameter estimations at test time by applying adversarial perturbations ([Zha+20b]) on the true hidden parameters during training, in order to make the policy more robust at transfer time. In the context of automated driving, we apply a rule-based adversarial function ($f_{\text{AP}}(.)$) to the true driver intentions based on their time to collision (TTC):

$$f_{\text{AP}}(\theta_i, t_0) = \begin{cases} \theta_i, & \text{if } |TX_i - TX_{\text{ego}}| > t_{\text{AP}} \\ -1, & \text{otherwise} \end{cases} \tag{4.9}$$

Here TX_i is the time for the vehicle i and TX_{ego} for the ego vehicle to reach the conflict zone assuming driving with constant velocity. We calculate the difference between these two time values which indicates the risk of collision between the two vehicles on intersecting lanes [JNG13] and also the complexity of estimating vehicle's intention. When the time difference is close to zero, vehicle i and ego vehicle have a high risk of collision when reaching the conflict zone, and therefore it is complex to estimate if the other vehicle is cooperative or non-cooperative. On the other hand, when the other vehicle reaches the conflict zone earlier or later than the ego vehicle by constant velocity assumption for both vehicles, it can be less complex to estimate its intention using an advanced behavior/intention prediction module. This is simple model of drivers' intention prediction and could be replaced by a non-linear perturbation in future works.

4.4 Evaluations on Interactive Automated Merging in Simulation

4.4.1 Simulation Environment and Main Challenges

We consider the automated merging problem where the RL agent controls an automated vehicle (ego vehicle) on the merging lane, which needs to interact with the other vehicles driving on the target lane. For that, we use the open-source simulator [Leu18] where other vehicles are controlled by the Intelligent Driver Model (IDM) [THH00] to keep safety distance with their front vehicle, but do not react to the ego vehicle. Therefore, in the original version of the simulator, all drivers have uncooperative intention. We applied some modifications to implement the cooperative behaviors for randomly selected cooperative vehicles. The cooperative drivers consider the projected position of the ego-vehicle on their lane as their front vehicle position and keep a safe distance to that.

In order to evaluate and compare the ability of a RL agent to generalize well enough when transferred to a new environment with new dynamics, we implemented three different automated merging environments:

- `Early-Brake`: Cooperative drivers reduce their velocity as early as possible to open a big merging gap for the ego vehicle.
- `Late-Brake`: Cooperative drivers keep their velocity until just before the conflict zone where they perform a harsh deceleration.
- `Mixed-Brake`: Cooperative drivers are randomly assigned to have one of the two (early-brake or late-brake) behaviors.

The main difference between these environments is the behavior of cooperative drivers regarding the merging attempt of the ego vehicle by having different comfortable deceleration limit in their IDM controller ($a_{\text{comf-max}}$=1.0 m/s^2 for early-brake and $a_{\text{comf-max}}$=5.0 m/s^2 for late-brake behaviors).

In summary, the main challenges in our simulation are: 1) Unknown drivers intentions and 2) Novel behaviors for cooperative drivers in the test environment.

Reward Function

The desired behavior to learn by RL is to safely merge into the target lane as fast as possible without collision or harsh deceleration, which is implied by this reward function:

$$R = \begin{cases} 1, & \text{if goal reached} \\ \text{-0.25}, & \text{if collision happened} \\ \text{-0.009}\ , & \text{if } v_{\text{ego}}^{(t)} < v_{\text{ego}}^{(t-1)} \\ \text{-0.003}\ , & \text{otherwise} \end{cases} \tag{4.10}$$

In the above, $v_{\text{ego}}^{(t)}$ is the velocity of the ego vehicle at current time (t), which we penalize its decrease to reduce harsh deceleration and encourage comfortable driving.

Action Space

We assume that the ego vehicle utilizes a low level path follower controller like [Ack+95] in order to follow a fixed route into the merging point. Therefore, the RL only controls the longitudinal velocity of the ego vehicle according to the current observation by selecting one of possible jerk commands from our action space:

$$\mathcal{A} = \{-1.5\frac{m}{s^3}, 0.0\frac{m}{s^3}, 1.5\frac{m}{s^3}\} \tag{4.11}$$

4.4.2 Baselines

We compare the performance of four different RL baselines during training in the source environment and also at test time when they are transferred to the test environments:

K-Markov Generalist

As proposed in [PSK20], one way to learn a generalized policy for HiP-MDP problems is to train a *generalist* policy using several training batches from all

possible tasks. We call this baseline *k-Markov Generalist* agent since it receives a k recent observations and is trained across all environments at the same time.

K-Markov Specialists

We implement *k-Markov Specialists* RL agents that are conditioned on observation history but only trained in one specific environment (`Early-Brake` or `Late-Brake`) and therefore are called k-Markov `Early-Brake` or k-Markov `Late-Brake`, respectively.

HiP-AP Specialists

Finally, we implement our proposed HiP-AP RL agent which is trained in one of the environments using the proposed hidden parameter adversarial perturbation approach (equation 4.3.4) and is subsequently transferred to the new environment using an online intention estimator network that is trained incrementally on the novel cooperative behaviors in that environment. Depending on which environment this agent is trained, it is named HiP-AP `Early-Brake` or HiP-AP `Late-Brake` accordingly.

HiP-Oracle Specialists

We also implement HiP-Oracle specialists as HiP-based baselines (Hidden Parameter (HiP)-Oracle `Early-Brake` or HiP-Oracle `Late-Brake`) which have access to the true driver intentions during training (provided by the simulator). Note that this is not a realistic assumption, but we are interested to see how fast these policies can converge during training.

4.4.3 Impact of HiP-MDP Modelling on Sample Efficiency During Training

In order to show the impact of modelling driver intentions as hidden parameters in a HiP-MDP formulation, we first trained HiP-Oracle and k-Markov specialist agents in each of the environments.

Fig. 4.6 (top) depicts the average episode reward of each policy during training, indicating that having direct access to the hidden parameter helps the HiP-Oracle policy to reach higher reward with fewer interactions and training updates. On the other hand, the k-Markov agents require more training and still do not reach the maximum reward achieved by the HiP-Oracle agent. We find that providing longer observation histories can help to improve the policy to some extent, but it is not a sample-efficient solution.

Moreover, Fig. 4.6 (bottom) shows the average episode reward of HiP-AP agent, which has direct access to perturbed drivers' intentions. Obviously HiP-AP requires more time to converge compared with the HiP-Oracle, due to the adversarial perturbations on the states. But this policy has the advantage to be transferred in new environments and receive uncertain estimations about drivers' intentions, which is not the case for the HiP-Oracle agent that only assumes to receive true values of the hidden parameters.

4.4.4 Policy Transfer in Target Environments with Novel Cooperative Behaviors

In this section, we compare the efficiency of each baseline when trained in one environment and transferred to new environments. For comparing performance of each baseline at transfer time, we formulate our criteria by asking the following questions.

Which source environment leads to the best performance in all target environments?

This can be an interesting question specially when a RL policy is trained with one accessible environment at training but needs to be robust to novel behaviors in other environments.

Table 4.1 provides average reward, collision rate and average episode length for two different HiP-Oracle agents trained in `Early-Brake` or `Late-Brake` environments and evaluated in all three environments. According to this table, the HiP-Oracle agent that is trained in the `Late-Brake` environment has higher return in all evaluation environments compared with the HiP-Oracle

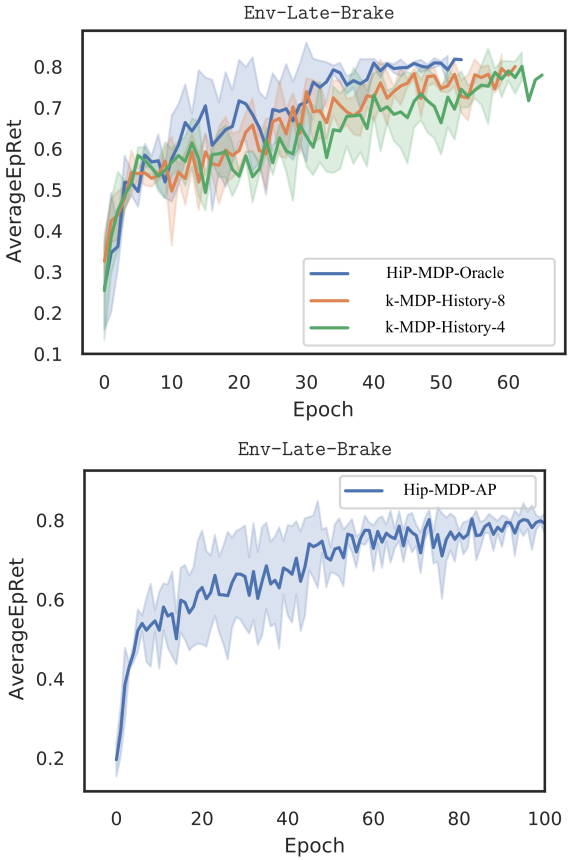


Figure 4.6: **Top:** Average episode return for the HiP-Oracle and k-Markov RL with different history lengths in the `Late-Brake` environment. **Bottom:** Average episode return for training the proposed HiP-AP agent in the `Late-Brake` environment. Note that due to the adversarial perturbation of the true hidden parameters, the training takes longer than other baseline agents.

agent trained in the `Early-Brake` environment. Note that the agent trained in the `Late-Brake` environment is about 4 seconds in average faster than the agent trained in the `Early-Brake` environment. This is mainly because in `Late-Brake` environment, the agent learns how to merge in tight merging

Transfer Env.	Metric	Training Env.	
		Early-Brake	Late-Brake
Early-Brake	Avg. Return	0.933 ± 3e-3	**0.942 ± 3e-3**
	Collision rate (%)	**1**	2
	Avg. Length (s)	52.5 ± 1.2	**48.2 ± 0.4**
Late-Brake	Avg. Return	0.93 ± 3e-3	**0.94 ± 4e-3**
	Collision rate (%)	2	2
	Avg. Length (s)	53.3 ± 1.2	**48.4 ± 0.6**
Mixed-Brake	Avg. Return	0.93 ± 6e-3	**0.94 ± 3e-3**
	Collision rate (%)	2	2
	Avg. Length (s)	53.5 ± 0.4	**49.3 ± 1.0**

Table 4.1: Comparing Oracle HiP-AP agents transferred in the training environment and in the other environments.

gaps and is therefore able to merge in easier scenarios with big merging gaps. While the agent trained in `Early-Brake` only learns to merge in large gaps and therefore the agent needs to yield to the most of the drivers even if they are late-brake cooperative.

How well the HiP-AP agent transfers to new environments with online intention predictor?

We compare the baselines with realistic assumptions that there is no direct access to the driver intentions when transferred to new environments. For that, we transfer all the baselines to different environments and measure their average performance over 100 episodes for 10 epochs (1000 episodes in total). The results are reported in Fig. 4.7. For the proposed HiP-AP agent with online intention estimator, we utilize an Multi-layer Perceptron (MLP) classifier that is trained incrementally to reduce the average prediction loss (Fig. 4.8) for predicting the true driver intentions using the recorded trajectories from previous epochs.

Due to similar reason explained in previous section, k-Markov specialist agent that is trained in `Late-Brake` outperforms the same agent trained in `Early-Brake` with higher return in all three test environments. The k-Markov generalist outperforms the `Late-Brake` specialist in the `Early-Brake` test environment and has similar performance in other environments.

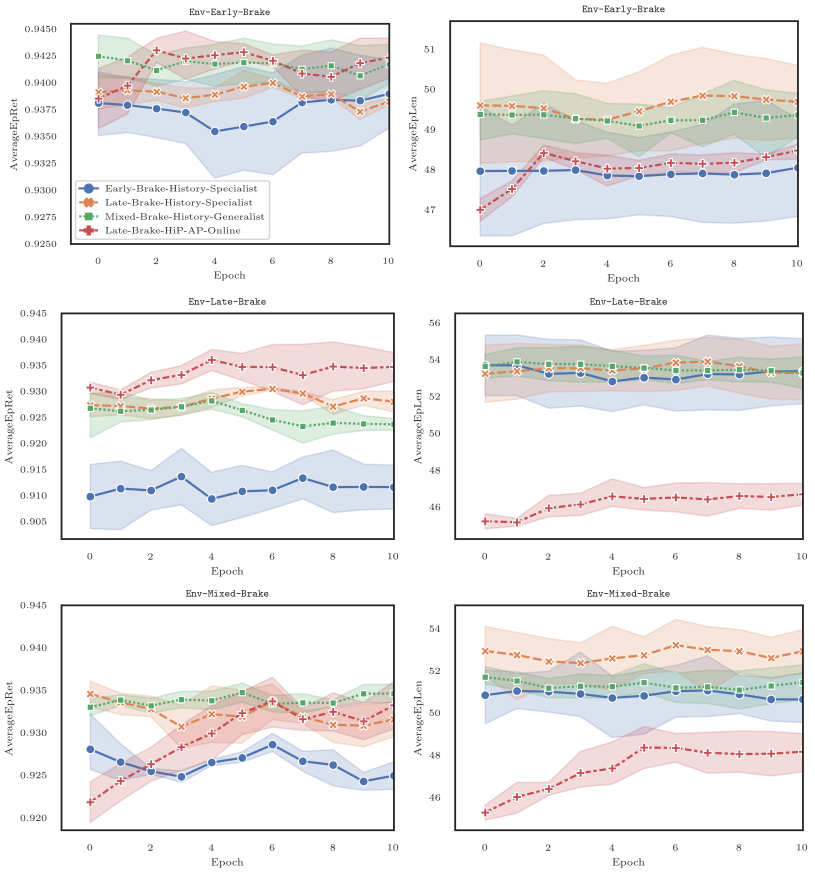


Figure 4.7: Average episode return (first column) and average episode length (second column) for different baselines (zero-shot) transferred to `Env-Early-Brake` (first row), `Env-Late-Brake` (second row) and `Env-Mixed-Brake` (last row).

The proposed HiP-AP agent is the only policy which is able to update its intention predictor module at transfer time. The prediction loss for the intention predictor module at transfer time is shown in Fig. 4.8 which shows the improvements on the accuracy of intention predictions during transfer time. It is clearly visible that the HiP-AP performance has a huge improvement from the first epoch to the

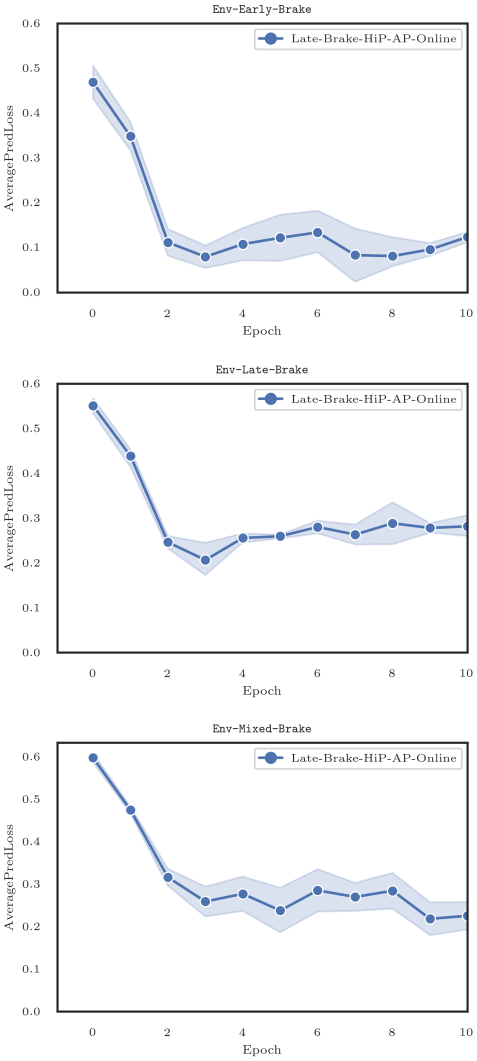


Figure 4.8: Average loss for the intention predictor loss of Late-Brake-HiP-AP agent while incrementally trained at different environments.

last epoch in all three environments. This is mainly the result of improvements on the intention predictor module as the RL policy is fixed. Therefore, after some epochs it outperforms both specialist history-based agents with higher reward.

4.5 Evaluations on Interactive Automated Merging with Real-World Datasets

In this section, we present the evaluation results of the RL merging policy which is trained based on real-world observational datasets. For that, first we explain the pipeline implemented in order to process the datasets and generate training data for the RL algorithm. The dataset is also used for providing a simulation environment and evaluating the trained RL policy with the real-world data.

We process the Interaction dataset ([Zha+19]) as an open-source automated driving dataset which includes recorded trajectories of vehicles in different scenarios including merging scenario. We use several recorded trajectories in Germany (DEU) and China (CHN) in this dataset. Each dataset contains several hours of recorded trajectories which will be processed in order to extract several merging scenarios for training the RL agent. In the following, we explain the main procedure to process the datasets and extract all merging scenarios from them.

4.5.1 Dataset Preprocessing

Vehicle State Transformation

Fig. 4.9 depicts an example frame of vehicles driving at a merging scenario in highway from German and China datasets. Using available open-source tools from common-road benchmarks ([AKM17]), we extract merging and normal highway lanes and project each vehicle position on its route in order to compute its position and velocity along its route (4.10). The vehicles' longitudinal distances are relative to the starting edge of the conflict zone, resulting in positive (negative) distances for vehicles behind (after) the conflict zone edge.

Merging Scenario Identification

We are interested to observe the trajectory of vehicles that are merging into the highway and let the RL observe their interactions with other drivers while merging. Therefore, for all time frames in a dataset, the list of vehicles on the

merging lane are extracted and the vehicle which has the minimum positive distance is identified as the current merging vehicle for that time frame.

Every time frame that has a new merging vehicle is identified as a starting time for a new merging scenario. Note that a new scenario may start only in one of these cases: 1) When a merging vehicle enters into the merging lane for the first time and there is no other vehicle on its lane before that vehicle. 2) When a merging vehicle merges into the highway lane and therefore the vehicle behind this vehicle has no other vehicle in front and will be considered as the new merging vehicle.

After identification of the starting time for a new merging scenario, the time frame which the merging vehicle reaches 20 meters after the conflict zone is identified as the ending time of that scenario. Therefore, each extracted merging scenario has three characteristics: vehicle ID for the merging vehicle, starting time and end time of the merging scenario.

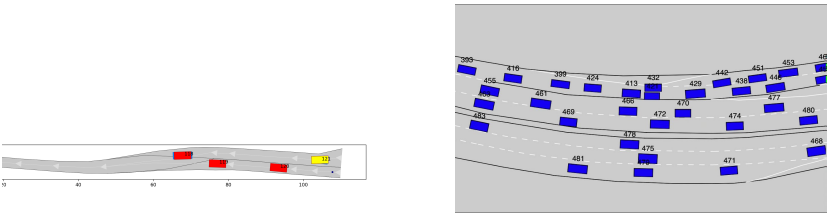


Figure 4.9: One frame from the merging scenario in the Germany (left) and China (right) Interaction dataset.

4.5.2 MDP Dataset Generation

After extracting all merging scenarios in a dataset, for each scenario the array of all MDP transitions from start to the end of merging trajectory is generated. Each MDP transition includes the current MDP state (s), next MDP state (s'), merging vehicle action (a) and MDP reward ($r = R(s, a, s')$) for that transition.

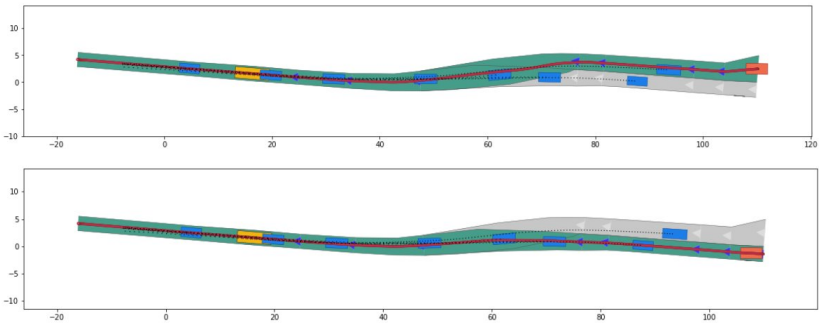


Figure 4.10: Merging and normal highway routes provided in the German Interaction dataset. We use these routes in order to compute vehicles' position and velocity along their lanes' center lines.

MDP State

Similar to our simulation experiments, the MDP state extracted for these observational trajectories provides all available information about the existing situation for a decision-making policy in order to generate the optimal action. We provide the distance and speed of the merging vehicle along its route, and closest vehicles in front and behind this vehicle on the highway lane. The true drivers' intention is calculated based on their final location relative to the ego vehicle at the end of the scenario. If their final position is behind the ego vehicle (they yield to our vehicle) their intention will be labeled as cooperative. If they are located after the ego vehicle (they merged earlier than our vehicle) their intentions are labeled as uncooperative.

The next MDP state is generated in the same way from the next time frame in the dataset. Fig. 4.11 and Fig. 4.12 show histogram of different state parameters generated from all identified merging scenarios from the DEU and CHN datasets respectively.

MDP Action

Since we aim to train a RL as an acceleration controller for merging in highway, the MDP action is calculated as the acceleration of the merging vehicle as

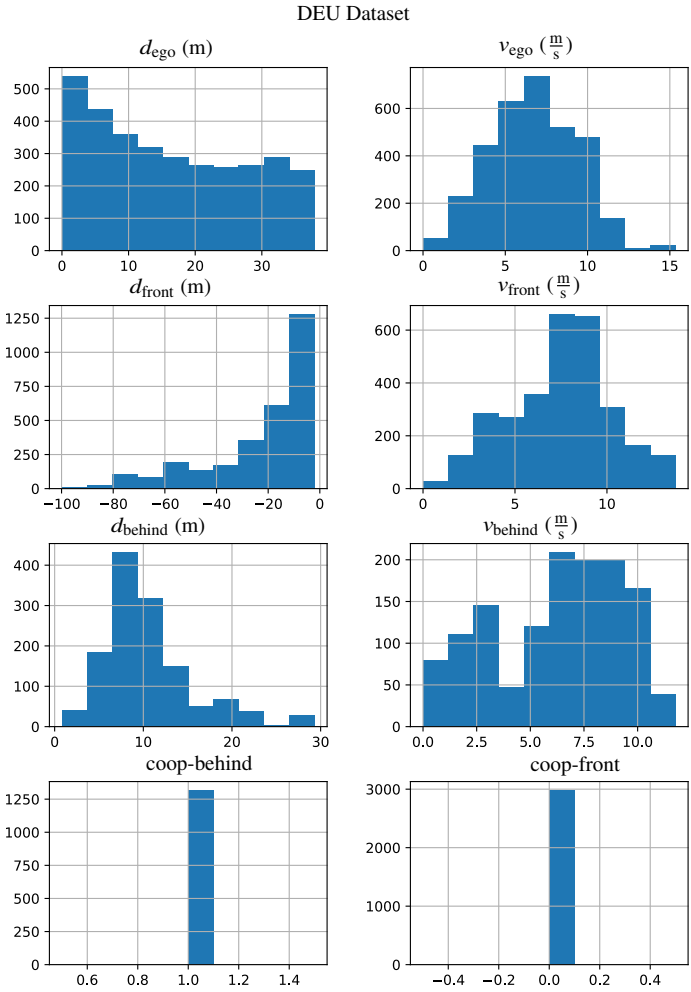


Figure 4.11: Histogram of different observation elements in the generated MDP dataset from the DEU dataset.

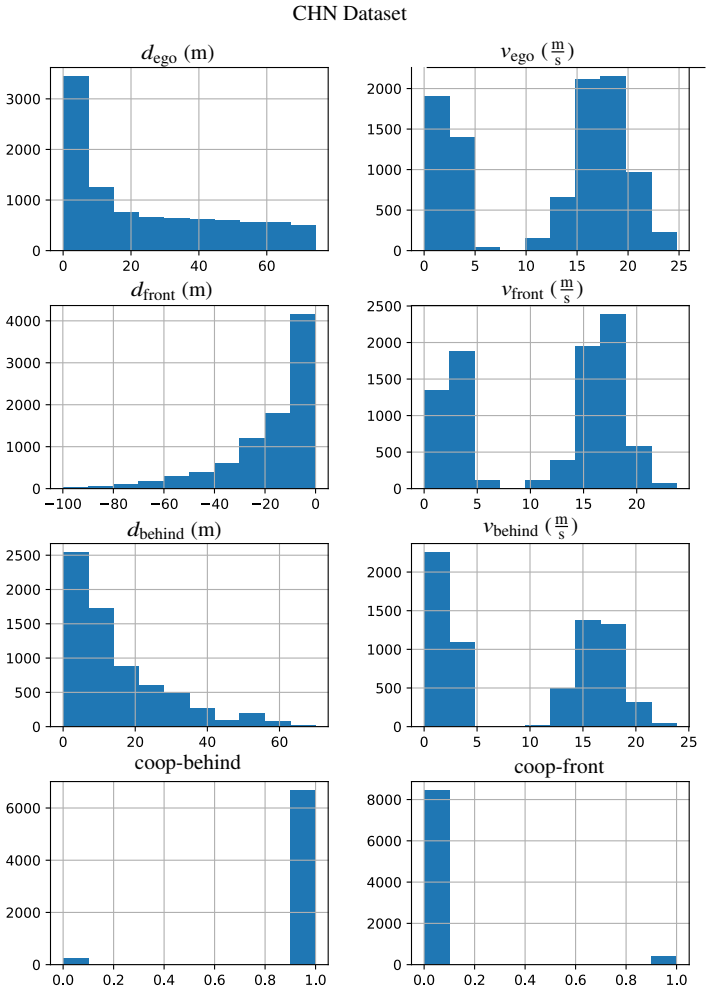


Figure 4.12: Histogram of different observation elements in the generated MDP dataset from the CHN dataset.

the difference of its longitudinal velocity along merging route between the current and the next time frame divided by the time difference. Fig. 4.13 show histogram of generated actions in the MDP datasets for DEU and CHN datasets.

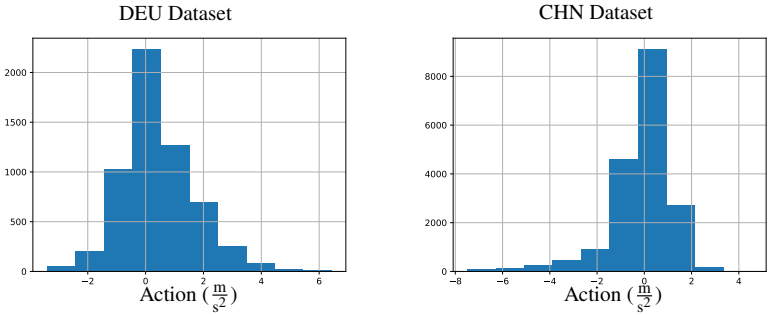


Figure 4.13: Histogram of MDP actions generated from the DEU dataset (left) and CHN dataset (right).

MDP Reward

We calculate the reward for the MDP transitions based on the Eq. 4.5.2. This reward function consists of a negative time penalty for every time frame except the last time frame where the vehicle reaches the end of merging scenario and the MDP reward will be a positive value. Fig. 4.14 shows the histogram of all calculated reward values for the DEU and CHN datasets.

$$R = \begin{cases} -0.001, & \text{if } d_{\text{ego}} > d_{\text{end}} \\ 0.1, & \text{otherwise.} \end{cases} \tag{4.12}$$

After generating all MDP transitions for all merging scenarios, they will be combined and stored in a database as training data for the offline RL policy. Note that the order of MDP transitions does not need to be stored in this database since in the end they will be randomly selected in small batches for iterative policy training.

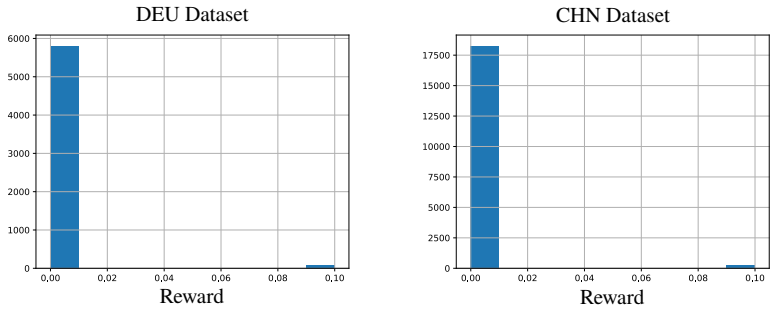


Figure 4.14: Histogram of MDP rewards generated from the DEU (left) and CHN (right) datasets.

4.5.3 Offline Reinforcement Learning Baselines

After generating the MDP datasets from our merging scenarios, we train RL agents using those datasets for an interactive automated merging policy. Since we are using recorded datasets and the RL agent does not have the possibility for online interactions with the environment in order to generate new transitions, receive feedback and update itself, we need to apply offline RL methodology.

In offline RL settings ([FMP19; ASN20; Lev+20]), it is assumed that the agent observes several trajectories from interactions of another policy which is called behavior policy. This policy can be a random action selector, an expert agent or different suboptimal agents. Selecting the appropriate offline RL algorithm depends on different factors especially the size of training data, type of RL action space, and importance of policy exploration [Lev+20].

Due to limited amount of scenarios in our dataset and considering generating vehicle accelerations as continuous actions, we selected BCQ in order to train our offline RL agent. For implementing and training the BCQ agent, we use the open-source d3rlpy library [SI22] which has an efficient interface to generate MDP datasets from our offline data and train several state-of-the art offline RL algorithms.

Oracle-BCQ

As the first baseline, we train a BCQ agent which has full observation on all state parameters including their true intention labels and call it Oracle-BCQ. We expect to achieve the best performance from this baseline, but it should also be considered that this is not a realistic solution, since the trained policy is conditioned on drivers' intention which is usually hard to estimate in real-world settings. However, comparing the performance of Oracle BCQ and other baselines with partial observation can provide useful insights about the complexity of the task and importance of considering drivers' intention for optimal policy training in our experiments.

Partially Observable BCQ (PO-BCQ)

The PO-BCQ does not observe the drivers' intention but has access to the other parameters in the state.

Hidden Parameter Adversarial Perturbation BCQ (HiP-AP-BCQ)

Finally, we implement our HiP-AP solution for addressing generalized and sample efficient RL under intention uncertainty in the offline RL setting and call it HiP-AP-BCQ.

Similar to the strategy explained in simulation experiments, here we apply the adversarial perturbation function in the Equation 4.3.4 on the true intention labels. In order to set a reasonable value for t_{AP} parameter which identifies the power of perturbation, we first have a look into the time difference of cooperative vehicles in the dataset as those that the ego vehicle could merge into their lanes earlier than they enter into the conflict zone. As it is visible in Fig. 4.15 the most frequent time difference ($TX_i - TX_{\mathrm{ego}}$) in both datasets is between 0.0 and 2.5 seconds. Interestingly, in the CHN dataset, we observe some cooperative drivers with negative time difference, meaning that those drivers could reach the conflict zone faster than the ego vehicle, but in the end they reduced their velocity and yielded to the ego vehicle. Considering the range of time differences in both datasets, we apply a uniform random selection of perturbation threshold for each specific training episode: $t_{\mathrm{AP}} \sim U(0, 2.5)$.

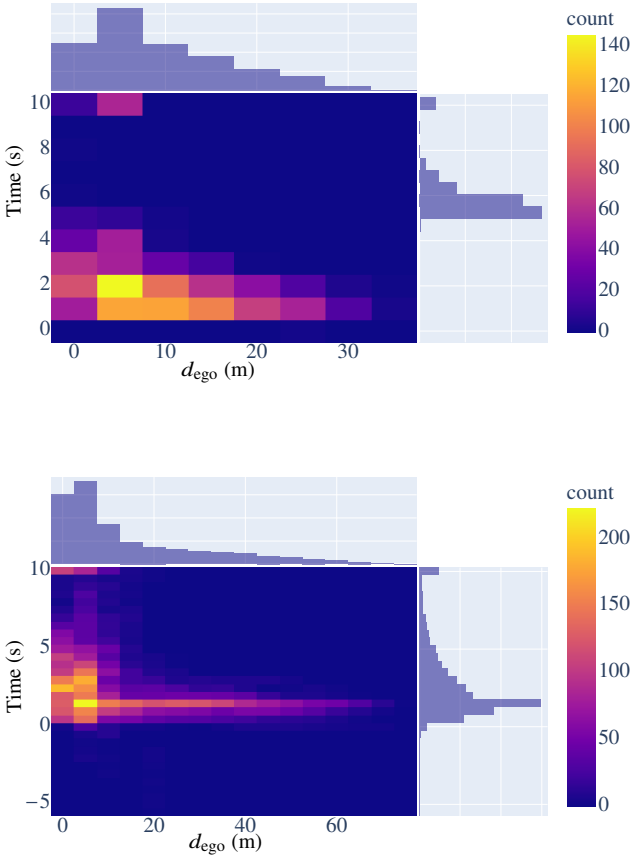


Figure 4.15: Histograms of cooperative vehicles' time difference and its relationship to the ego vehicle distance to the merging point for DEU dataset (top) and CHN dataset (bottom).

4.5.4 Policy Simulation with Recorded Datasets

In order to evaluate the trained RL policy with the recorded merging scenarios from the Interaction dataset, we deploy a test (artificial) merging vehicle which

starts exactly with the same position and velocity of the real merging vehicle for each scenario. However, the future states of the test vehicle are updated based on the control decisions (accelerations) generated by the RL policy, rather than following the states in the recorded trajectories. Since there are no collisions in the real data, a policy that strictly follows the recorded trajectories would result in a zero collision rate. Although this might not be the most effective way to evaluate the trained policies, it is our only option for testing their performance with our offline dataset. Therefore, we also assess the policies visually by examining their generated trajectories in various scenarios to provide an additional layer of evaluation.

Similar to our simulation experiments, the RL only generates accelerations without any lateral control and the vehicles follow their fixed merging path. Based on that, we implement a simulator which can randomly select a merging scenario and then provide the initial MDP state of the environment for the RL policy and follow the RL actions in order to generate the next simulated MDP states. Note that only the state of the ego vehicle will be simulated and for other vehicles in the scenario, their recorded trajectories in the real-world will be followed for future simulation steps.

4.5.5 Evaluation Results

In this section, we provide evaluation results of different BCQ baselines trained with our offline MDP dataset and evaluated in the recorded scenarios from real-world merging scenarios in DEU and CHN datasets.

As the most important evaluation metric, we record the number of collisions that happen during testing each baseline and based on that, we report success rate as the average number of successful episodes. It may happen that the other vehicles have collision with the simulated test vehicle since they just follow their recorded trajectory without any interaction with this vehicle. We only neglect those types of collisions only if they happen when the test vehicle is behind the merging conflict zone, since they are not indicating bad merging policy for the test vehicle. However, the collisions that happen when the test vehicle is after or inside the merging conflict zone are considered as consequences of bad decisions from the RL policy and will be reported in our evaluations. The amount of success rates for different baselines in different observation settings

are provided in Table 4.2. Note that HiP-AP-BCQ is the only baseline that can be evaluated in both full observation (with access to intentions) and partial observation settings (without access to intentions). In other words, here we do not learn or embed an external intention predictor for HiP-AP-BCQ and instead evaluate its performance in partial or full observation settings.

We also provide time-distance diagrams for the ego, front, and behind vehicles during some example episodes. Additionally, we provide velocity and acceleration profiles of the ego vehicle during those episodes.

Observation	Oracle-BCQ	PO-BCQ	HiP-AP-BCQ (ours)
Full observation	90	–	85
Partial observation	–	75	80

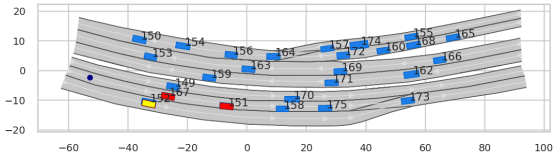
Table 4.2: Success rate of different baselines evaluated in the CHN dataset.

Comparing Oracle-BCQ and PO-BCQ

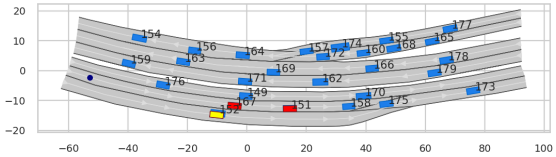
In table 4.2, we can observe that Oracle-BCQ has higher success rate than PO-BCQ agent. This justifies the importance of knowing about the intention of other vehicles for interactive merging and also the fact that the problem needs to be formulated as a POMDP rather than MDP due to decision-making under intention uncertainty.

We have a closer look into one of the episodes where PO-BCQ fails while Oracle-BCQ can finish the episode successfully. Fig. 4.16 shows some frames of the Oracle-BCQ trajectory in scenario CHN-19-8. The velocity and acceleration profiles of this agent is also visible in Fig. 4.17. Since Oracle-BCQ knows about the uncooperative intention of the front vehicle, it starts with negative acceleration in order to safely merge after this vehicle and successfully finishes the episode.

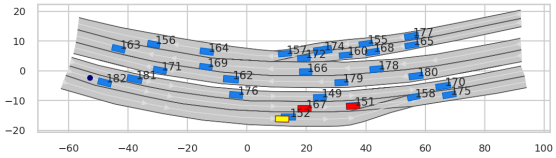
For the same episode, we run the PO-BCQ to control the merging vehicle. Since PO-BCQ has no access to the uncooperative driver's intention, it fails and has a collision with this vehicle. By having a closer look into this episode (Fig. 4.18 and Fig. 4.19), we observe that PO-BCQ accelerates at t=0 with acceleration 1.4 ($\frac{\text{m}}{\text{s}}$) and first tries to merge before the uncooperative driver in front. However, at t=2.0 (s) it changes the decision to decelerate but can not be successful and has a collision with this vehicle at t=5.1 (s).



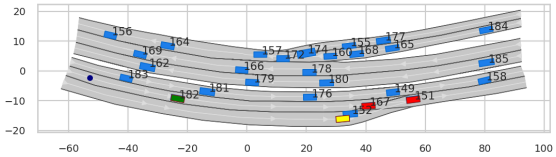
(a) t = 0 s



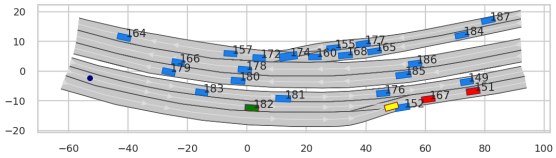
(b) t = 1.5 s



(c) t = 3 s



(d) t = 4.5 s



(e) t = 6.0 s

Figure 4.16: Different frames of the scenario CHN-19-8 where Oracle-BCQ agent (yellow) with access to the driver's intention is controlling the merging vehicle (number 152) and knows about the intention of the front uncooperative vehicle (number 167). Therefore, Oracle-BCQ can safely merge after that vehicle. Note that the original trajectory of the merging vehicle (number 152) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

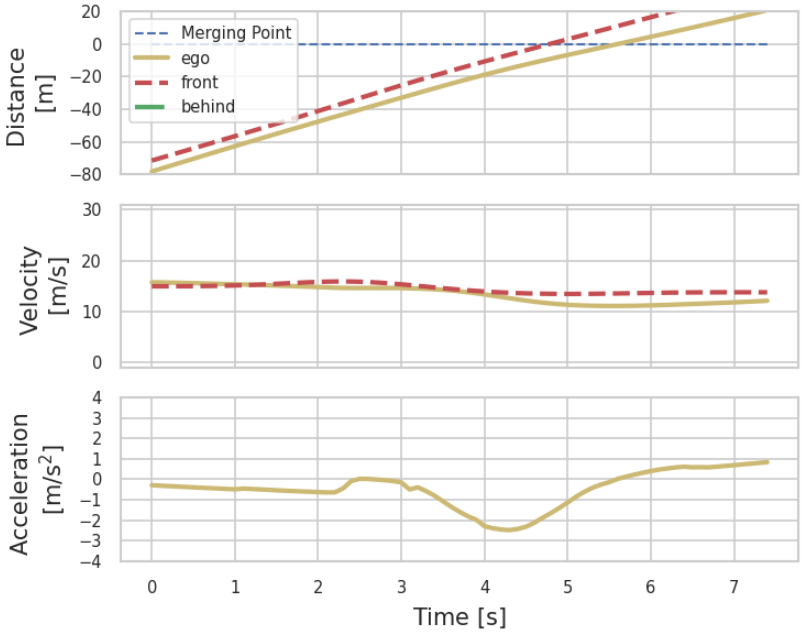
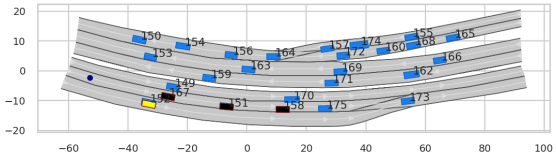
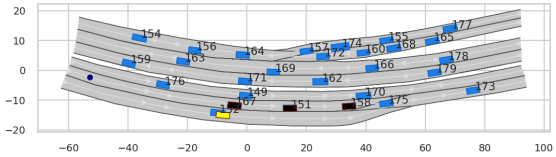


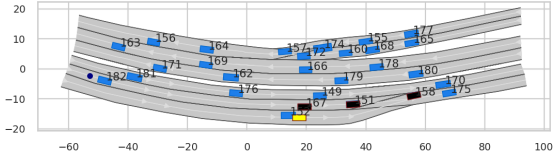
Figure 4.17: Distances, velocity and acceleration profile for the Oracle-BCQ agent at scenario CHN-19-8. The Oracle agent knows about the intention of front uncooperative vehicle, therefore it can safely merge after that vehicle.



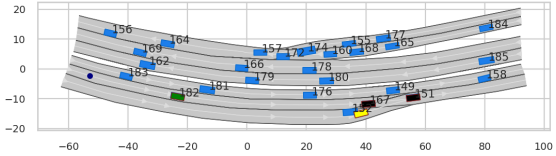
(a) t = 0 s



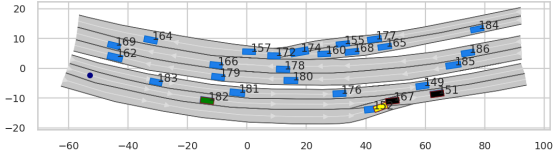
(b) t = 1.5 s



(c) t = 3 s



(d) t = 4.5 s



(e) t = 5.1 s

Figure 4.18: Different frames of the scenario CHN-19-8 where PO-BCQ agent (yellow) without access to the driver's intention is controlling the merging vehicle (number 152) and has a collision with the uncooperative front vehicle (number 167) at t=5.1 (s). Note that the original trajectory of the merging vehicle (number 152) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

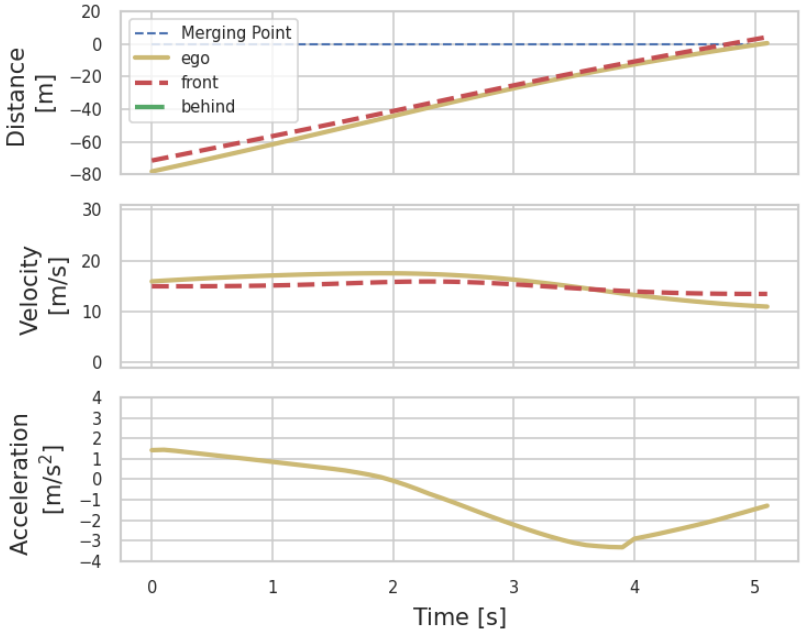
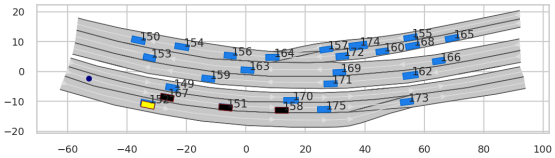


Figure 4.19: Distances, velocity and acceleration profile for the PO-BCQ agent at scenario CHN-19-8. The PO-agent does not know about the intention of front uncooperative vehicle and has a collision with it at t=5.1 (s).

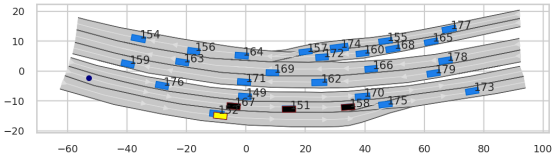
Comparing PO-BCQ and HiP-AP-BCQ

According to the Table 4.2, there are some scenarios where PO-BCQ agent fails but HiP-AP-BCQ with same partial observations (unknown intention) does not fail. This shows the advantage of applying adversarial perturbation on drivers' intention during policy training which improves the performance of policy at runtime. In other words, since the agent knows about the true drivers' intention for *some* training scenarios, it can learn a better solution about *how to merge between vehicles* by observing examples about *how cooperative and non-cooperative drivers behave*. At the same time, the agent also experiences some drivers with unknown intention (due to perturbation) and therefor learns how to deal with these situations.

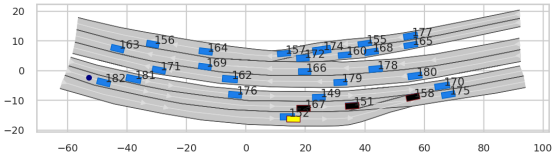
We have a closer look again in episode "CHN-19-8" where PO-BCQ failed but Oracle-BCQ could be successful. This time we run HiP-AP-BCQ with partial observation settings (similar to PO-BCQ) in this episode (Fig. 4.20). As it is visible in the velocity and acceleration profiles of this agent in Fig. 4.21, HiP-AP-BCQ starts to accelerate first and merge before the front vehicle. But in contrast to the PO-BCQ, it decelerates earlier (t=1.7 (s)) and can successfully merge after the uncooperative car in front.



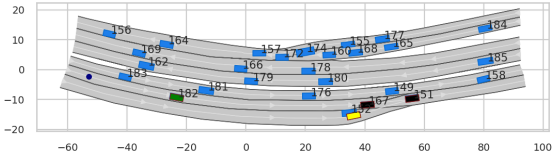
(a) t = 0 s



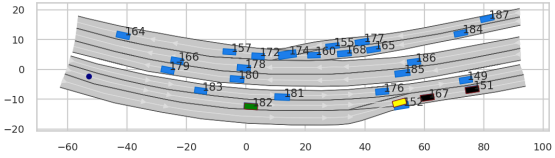
(b) t = 1.5 s



(c) t = 3 s



(d) t = 4.5 s



(e) t = 6.0 s

Figure 4.20: Different frames of the scenario CHN-19-8 where HiP-AP-BCQ (our approach) agent (yellow) without access to the driver's intention is controlling the merging vehicle (number 152). Although HiP-AP-BCQ does not know about the intention of uncooperative front vehicle (number 167), it safely merges after this vehicle. Note that the original trajectory of the merging vehicle (number 152) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

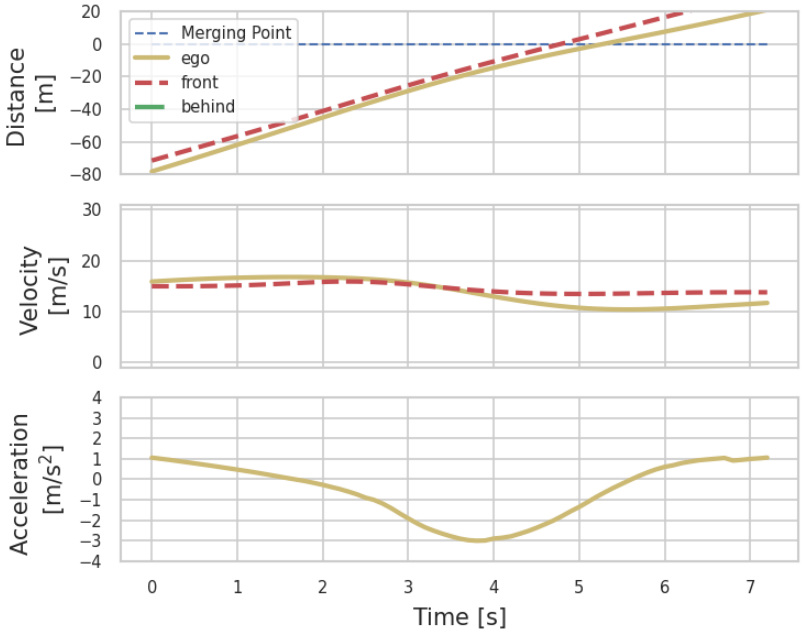
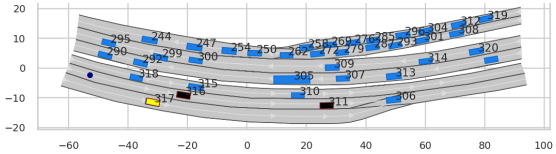


Figure 4.21: Distances, velocity and acceleration profile for the HiP-AP-BCQ agent at scenario CHN-19-8. Although HiP-AP-BCQ does not know about the intention of uncooperative front vehicle, it safely merges after this vehicle

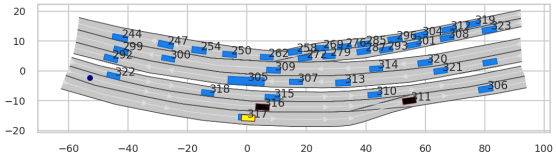
Applying Safety Verification

As we discussed the results in Table 4.2, all three baselines, even the Oracle, are not collision free. In Chapter 3 we discussed multiple reasons that may result in unsafe behaviors from a trained agent and the need for the safety verification module in order to prevent unsafe outcomes in real-world applications. After applying our proactive safety verification on top of the decisions from each agent, we observe that the collisions will not happen anymore, although there will be uncomfortable emergency maneuvers from the safety module in order to prevent them.

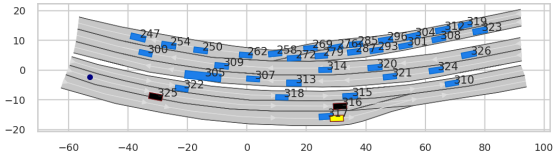
Fig. 4.22 and Fig. 4.23 show performance of HiP-AP-BCQ in episode CHN-15-686 which is one of the failure episodes for this agent. Fig. 4.24 and Fig. 4.25 show the performance of this agent combined with proactive safety module. We can see how the proactive safety module forces the agent to reduce its acceleration much earlier and prevent a collision with the front vehicle.



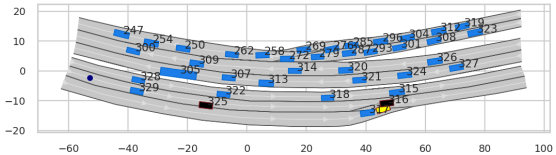
(a) t = 0 s



(b) t = 1.5 s



(c) t = 3 s



(d) t = 3.9 s

Figure 4.22: Different frames of the scenario CHN-15-686 where HiP-AP-BCQ agent (yellow) without access to the driver's intention is controlling the merging vehicle (number 317) and has a collision with the uncooperative front vehicle (number 316) at t=3.9 (s). Note that the original trajectory of the merging vehicle (number 317) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

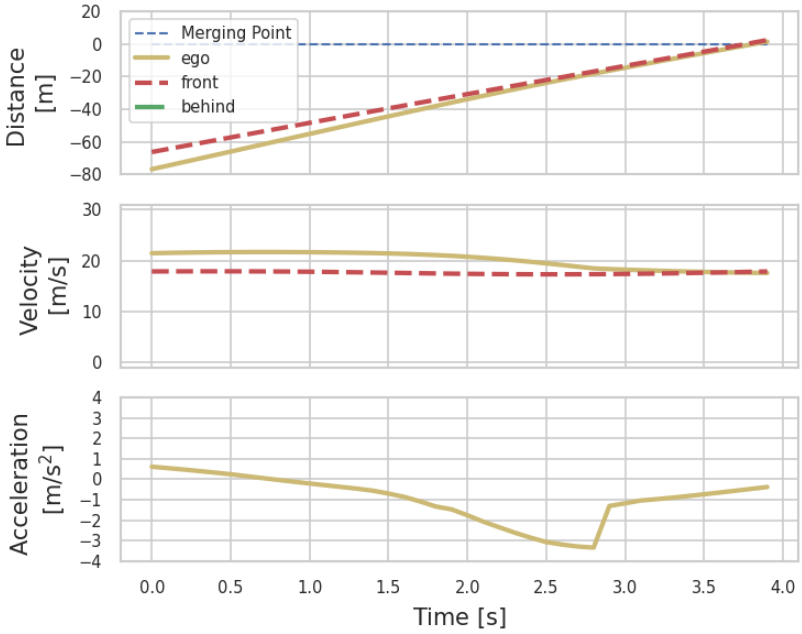
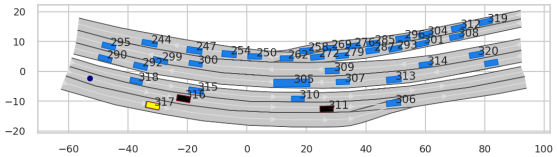
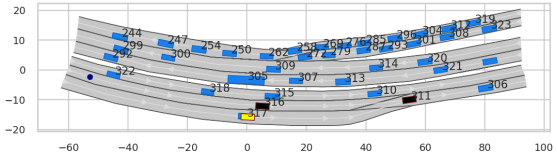


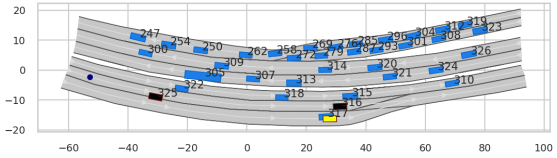
Figure 4.23: Distances, velocity and acceleration profile for the HiP-AP-BCQ agent at scenario CHN-15-686. The HiP-AP-BCQ agent does not know about the intention of front uncooperative vehicle and has a collision with it at t=3.9 (s).



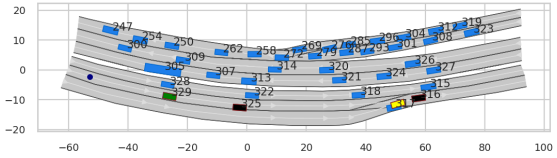
(a) t = 0 s



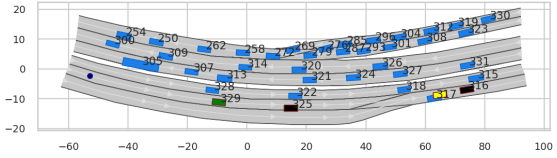
(b) t = 1.5 s



(c) t = 3 s



(d) t = 4.5 s



(e) t = 5.4 s

Figure 4.24: Different frames of the scenario CHN-15-686 where HiP-AP-BCQ agent with safety verification (yellow) is controlling the merging vehicle (number 317). The agent does not know about the uncooperative intention of the front vehicle (number 316) and tries to overtake this vehicle and merge first. However, the safety verification layer interrupts and decelerates to prevent a collision. Note that the original trajectory of the merging vehicle (number 317) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

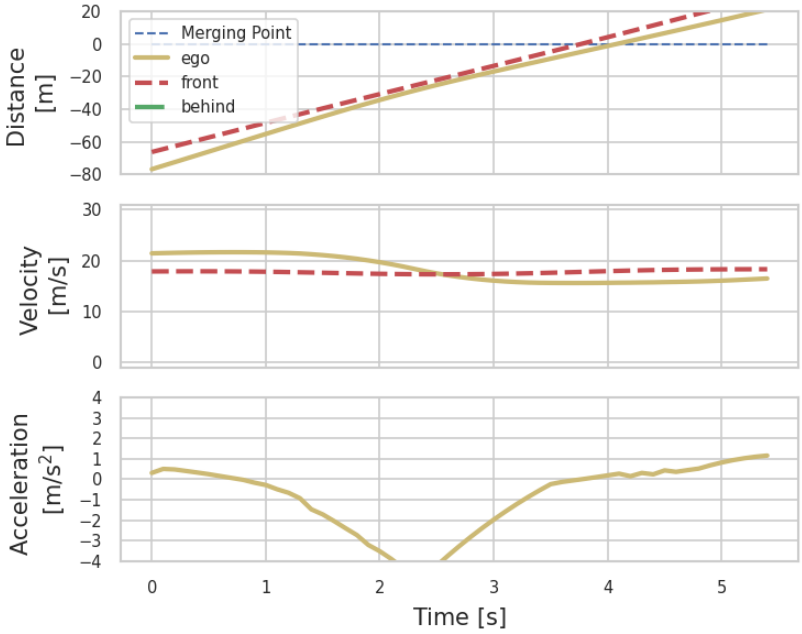


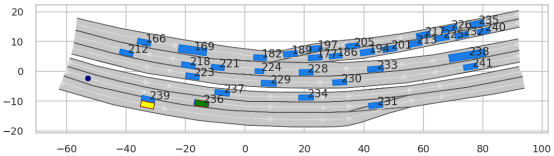
Figure 4.25: Distances, velocity and acceleration profile for the HiP-AP-BCQ agent with safety verification at scenario CHN-15-686. The HiP-AP-BCQ agent does not know about the intention of front uncooperative vehicle and starts with an acceleration. However, the safety verification layer interrupts and decelerates to prevent a collision.

Safety Verification with Invalid Intention Predictions

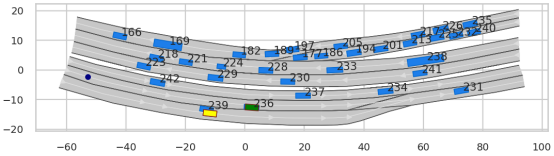
We also evaluated the performance of HiP-AP-BCQ with and without safety module in situations where the intention predictor provides invalid predictions due to inaccurate detections, novel behavior observations, or any other reason. For that, we perform an ablation study by changing the intention of uncooperative drivers to cooperative inside the provided observations for the HiP-AP-BCQ agent.

Fig. 4.26 and Fig. 4.27 show some frames and velocity/acceleration profiles of the HiP-AP-BCQ agent without safety module which receives invalid predictions during the episode CHN-13-323. As it is visible, the HiP-AP-BCQ agent tries to overtake the front vehicle (assuming it is a cooperative driver) with positive accelerations at the beginning. From t=3.0 (s) HiP-AP-BCQ decelerates to stop before the conflict zone but finally can not prevent a collision with this vehicle.

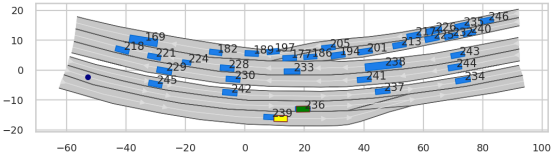
Now we apply the safety module together with the HiP-AP-BCQ agent for the exact episode with invalid intention predictions. In Fig. 4.28 and Fig. 4.29, the results of running safety module on HiP-AP-BCQ decisions are provided. As it is depicted, this time the proactive safety layer forces the vehicle to decelerate harshly and prevent a collision. This is mainly because the safety module always considers the worst-case assumption for the front vehicle (to be uncooperative).



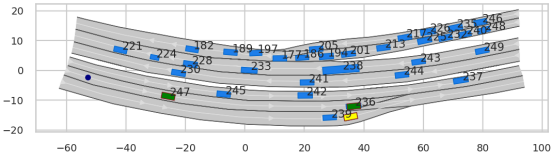
(a) t = 0 s



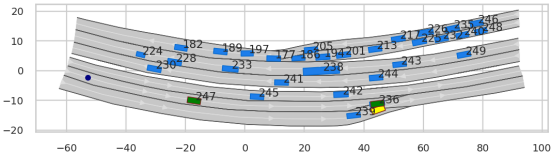
(b) t = 1.5 s



(c) t = 3 s



(d) t = 4.5 s



(e) t = 5.2 s

Figure 4.26: Different frames the scenario CHN-13-323 where HiP-AP-BCQ agent (yellow) is controlling the merging vehicle (number 236) while receiving wrong prediction for the intention of uncooperative front vehicle (green). Therefore, the HiP-AP-BCQ agent tries to overtake this vehicle and finally has a collision with it at t=5.2 (s). Note that the original trajectory of the merging vehicle (number 236) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

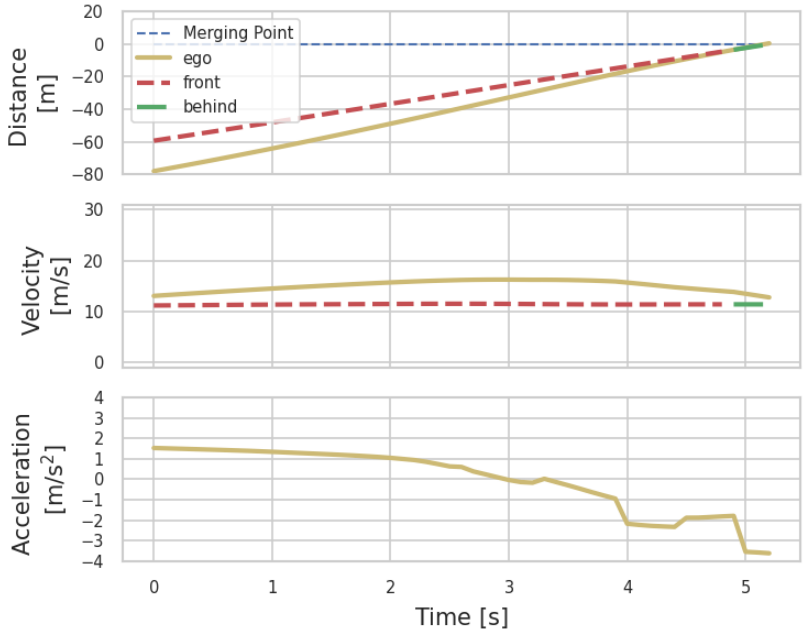
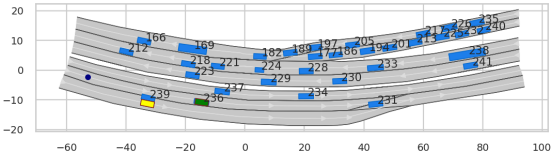
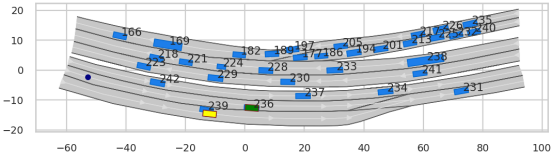


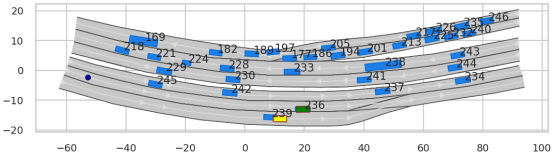
Figure 4.27: Velocity/acceleration profile in the scenario CHN-13-323 for the HiP-AP-BCQ agent while receiving wrong prediction for the intention of uncooperative front vehicle. Therefore, the HiP-AP-BCQ agent tries to overtake this vehicle and finally has a collision with it at t=5.2 (s).



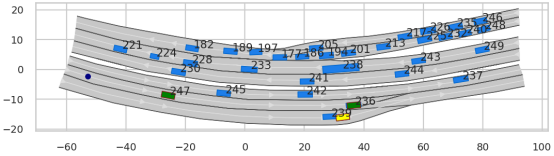
(a) $t = 0$ s



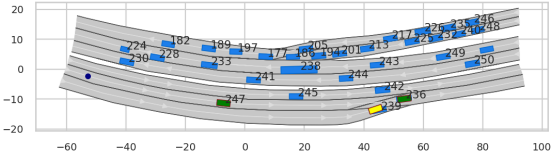
(b) $t = 1.5$ s



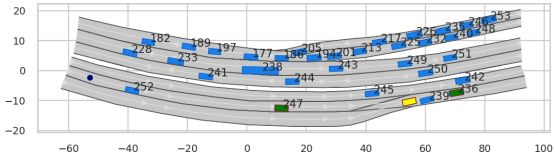
(c) $t = 3$ s



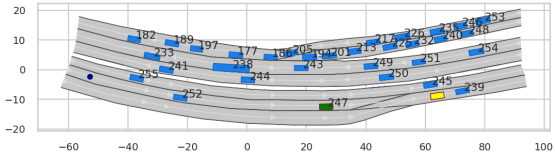
(d) $t = 4.5$ s



(e) $t = 6.0$ s



(f) t = 7.5 s



(g) t = 8.7 s

Figure 4.28: Different frames of the scenario CHN-13-323 where AP BCQ agent with safety module (yellow) is controlling the merging vehicle (number 317). The agent does not know about the uncooperative intention of the front vehicle (number 316) and tries to overtake this vehicle and merge first. However, the safety verification layer interrupts and decelerates to prevent a collision. Note that the original trajectory of the merging vehicle (number 317) is not part of the simulation and is only shown here (blue vehicle below yellow) for comparison.

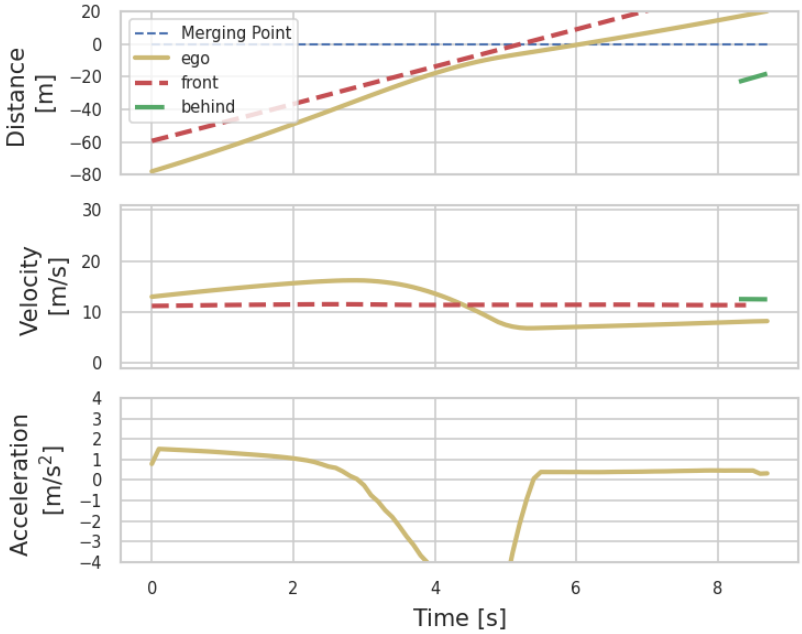


Figure 4.29: Velocity/acceleration profile in the scenario CHN-13-323 where HiP-AP-BCQ agent with safety layer is controlling the merging vehicle while receiving wrong prediction for the intention of uncooperative front vehicle. Due to wrong prediction, the HiP-AP-BCQ agent tries to overtake this vehicle while the safety layer forces the agent to decelerate from 2.9 (s). Finally, the agent can safely merge after the uncooperative vehicle. Note that a vehicle behind the ego vehicle appears in the list of observations during the last second of the scenario with its trajectory is shown in green.

4.6 Conclusions

In this chapter we addressed generalized RL under intention uncertainty for automated driving. We explained the conventional k-Markov approach for addressing partial observation in RL and its main drawbacks regarding modularity, sample efficiency, and inability to be transferred into new environments with novel behaviors for cooperative/non-cooperative drivers.

In order to address such issues, we proposed our approach HiP-AP RL which models drivers intentions as hidden parameters which have different emission functions in different environments. Therefore, the RL policy assumes to receive estimates about the value of hidden parameters from an external module which can be trained online at the transfer time or be provided as off-the-shelf. In order to make the policy robust to uncertain predictions from this module, we apply adversarial perturbation on the true values of hidden parameters during training.

In our simulation experiments we showed the ability of the HiP-AP agent to outperform k-Markov baselines when transferred in new environments with novel behaviors. Moreover, in our experiments with real-world observational datasets, we demonstrated an offline RL agent based on HiP-AP which is able to operate in both situations: with or without access to accurate drivers' intention predictions. The policy also demonstrated completely safe evaluations when it is combined with the proactive safety verification module proposed in Chapter 3 even when receiving invalid predictions.

5 Conclusions and Future Directions

5.1 Conclusions

In this dissertation, we addressed the safety and generalization of Reinforcement Learning (RL) as applied to automated driving. In Chapter 3, we proposed a safe and risk-aware RL framework for automated driving at occluded intersections. Our framework considers a set of worst-case assumptions for unknown parameters in the scenario, such as intention of other drivers, occluded vehicles, and perception noise.

In contrast to most previous works, which aim to minimize the number of crashes or unsafe states without completely preventing them, our framework guarantees that the RL policy is collision-free based on the worst-case assumptions. This is done in the safety module in our framework which makes sure that the action generated by the RL framework will result in a proactive safe state, meaning that if any of the worst-case assumptions (or all together) happen, an emergency maneuver is feasible to execute in order to prevent a collision. We combined this safety layer with distributional RL to learn comfortable and adaptive policies which can tune their risk sensitivity based on a single parameter provided by the user at run-time.

In addition to safety, in Chapter 4, we addressed generalization and sample-efficiency of RL policies in interactive automated driving scenarios. We showed that previous approaches addressing this problem utilize a history of recent observations as a *rich state* representation in order to implicitly predict the intention of other drivers resulting in complex learning models. Such models are not sample efficient and also not transferable to new environments with new dynamics. For that, we proposed to model the problem with Hidden-Parameter Block MDP (HiP-BMDP) formulation which allows us to condition the RL policy on *compact state* representations resulting in modular and sample-efficient RL policies. The policy therefore receives a prediction of drivers'

intentions from a predictive model which can either be trained in parallel with the RL procedure or be provided as a plug in. In order to address generalization, we proposed hidden parameter adversarial perturbation (HiP-AP) RL which is trained with perturbed values of true states and therefore is able to transfer to new environments with new dynamics and uncertain estimations about the intention of other drivers.

Finally, in Sec. 4.5, we combined the proactive safety verification, hidden parameter adversarial perturbation and offline RL to learn safe, interactive and sample efficient policies from real-world datasets in automated driving. We showed that using the proposed HiP-AP methodology, we could learn sample efficient and robust policies for interactive policies with limited real-world datasets. Moreover, we empirically showed how the proactive safety verification layer results in safe and collision free RL policies in these real-world evaluations.

5.2 Future Directions

Several future works regarding safe, generalized and interactive RL under uncertainty for real-world decision-making problems are promising.

In the context of automated driving, our framework for addressing safe and interactive learning-based decision-making approaches can be applied on other scenarios such as safe interactions with pedestrians. Moreover, for interactions with drivers, the drivers' intention model (cooperative vs non-cooperative) can be extended to cover other types of behaviors (e.g. rule compliant vs non-compliant) for different scenarios. The worst-case assumptions can also be increased to cover more complex scenarios such as a sudden lane change of another vehicle in our target lane.

Our contribution to interactive policies with hidden parameter adversarial perturbation can be applied to other similar problems, especially those where unknown hidden parameters play an important role in optimal decision-making. In such problems, the identification of hidden parameters is usually either too expensive or time-consuming.

Our approach can effectively address the problem of unknown genetic information in patients, which is crucial for selecting the best treatment by a physician, or the unknown preferences of a customer in a product recommender system.

In both cases, the decision-making policy can be trained in an Oracle training environment with full access to compact states and then transferred to the final environment, where it receives estimations of the hidden parameters from specialized predictors tailored to that environment.

Bibliography

[Ack+95] J. Ackermann, J. Guldner, W. Sienel, R. Steinhauser, and V. I. Utkin. “Linear and nonlinear controller design for robust automatic steering”. In: *IEEE Transactions on control systems technology* 3.1 (1995), pp. 132–143.

[AD14] M. Althoff and J. M. Dolan. “Online verification of automated road vehicles using reachability analysis”. In: *IEEE Transactions on Robotics* 30.4 (2014), pp. 903–918.

[AKM17] M. Althoff, M. Koschi, and S. Manzinger. “CommonRoad: Composable benchmarks for motion planning on roads”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 719–726.

[Als+17] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. “Safe reinforcement learning via shielding”. In: *arXiv preprint arXiv:1708.08611* (2017).

[ASN20] R. Agarwal, D. Schuurmans, and M. Norouzi. “An optimistic perspective on offline reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 104–114.

[BDM17] M. G. Bellemare, W. Dabney, and R. Munos. “A distributional perspective on reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 449–458.

[Bel66] R. Bellman. “Dynamic programming”. In: *Science* 153.3731 (1966), pp. 34–37.

[Bou+18] M. Bouton, K. Julian, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. “Utility decomposition with deep corrections for scalable planning under uncertainty”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 462–469.

[Bou+19a] M. Bouton, K. D. Julian, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. “Decomposition methods with deep corrections for reinforcement learning”. In: *Autonomous Agents and Multi-Agent Systems* 33.3 (2019), pp. 330–352.

[Bou+19b] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. “Cooperation-aware reinforcement learning for merging in dense traffic”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 3441–3447.

[Bus20] M. Busch. “Dynamic Input for Deep Reinforcement Learning at Unsignalized Intersections with Deep Sets”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020.

[Cen+19] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli. “Liability, ethics, and culture-aware behavior specification using rulebooks”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8536–8542.

[Dab+18a] W. Dabney, G. Ostrovski, D. Silver, and R. Munos. “Implicit Quantile Networks for Distributional Reinforcement Learning”. In: *International Conference on Machine Learning*. 2018, pp. 1096–1105.

[Dab+18b] W. Dabney, M. Rowland, M. Bellemare, and R. Munos. “Distributional reinforcement learning with quantile regression”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[DDL17] C. Dong, J. M. Dolan, and B. Litkouhi. “Intention estimation for ramp merging control in autonomous driving”. In: *2017 IEEE intelligent vehicles symposium (IV)*. IEEE. 2017, pp. 1584–1589.

[DK16] F. Doshi-Velez and G. Konidaris. “Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations”. In: *IJCAI: Proceedings of the Conference*. Vol. 2016. 2016, p. 1432.

[Du+19] S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudik, and J. Langford. “Provably efficient RL with rich observations via latent state decoding”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1665–1674.

[EL21] B. Eysenbach and S. Levine. “Maximum entropy rl (provably) solves some robust rl problems”. In: *arXiv preprint arXiv:2103.06257* (2021).

[Eng21] T. Engelgeh. “Risk-Sensitive Behavior Generation for Vehicles at Partially Occluded Intersections Using Attention-Based Distributional Reinforcement Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2021.

[FAL17] C. Finn, P. Abbeel, and S. Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *arXiv:1703.03400 [cs]* (July 2017). arXiv: 1703.03400.

[Fat+21] M. Fatemi, T. W. Killian, J. Subramanian, and M. Ghassemi. “Medical dead-ends and learning to identify high-risk states and treatments”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4856–4870.

[FMP19] S. Fujimoto, D. Meger, and D. Precup. “Off-policy deep reinforcement learning without exploration”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2052–2062.

[Gel+19] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. “Deepmdp: Learning continuous latent space models for representation learning”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2170–2179.

[Goo+13] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. “An empirical investigation of catastrophic forgetting in gradient-based neural networks”. In: *arXiv preprint arXiv:1312.6211* (2013).

[Her+16] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner. “Black-box alpha divergence minimization”. In: *International Conference on Machine Learning*. PMLR. 2016, pp. 1511–1520.

[Hig+17] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. “Darla: Improving zero-shot transfer in reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1480–1490.

[How60] R. A. Howard. *Dynamic programming and markov processes*. The MIT Press, 1960.

[HS97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[Hub+17] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller. "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1671–1678.

[Hub+18] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller. "A belief state planner for interactive merge maneuvers in congested traffic". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1617–1624.

[Hue+19] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker. "Dynamic Input for Deep Reinforcement Learning in Autonomous Driving". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 7566–7573.

[INF18] D. Isele, A. Nakhaei, and K. Fujimura. "Safe Reinforcement Learning on Autonomous Vehicles". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 1–6.

[Ise+18] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. "Navigating Occluded Intersections with Autonomous Vehicles Using Deep Reinforcement Learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 2034–2039.

[JNG13] F. Jiménez, J. E. Naranjo, and F. García. "An improved method to calculate the time-to-collision of two vehicles". In: *International Journal of Intelligent Transportation Systems Research* 11 (2013), pp. 34–42.

[Kam+20] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller. "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 1205–1212.

[Kam+21] D. Kamran, T. Engelgeh, M. Busch, J. Fischer, and C. Stiller. “Minimizing safety interference for safe and comfortable automated driving with distributional reinforcement learning”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1236–1243.

[Kam+22] D. Kamran, T. D. Simão, Q. Yang, C. T. Ponnambalam, J. Fischer, M. T. Spaan, and M. Lauer. “A modern perspective on safe automated driving for different traffic dynamics using constrained reinforcement learning”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, pp. 4017–4023.

[KL51] S. Kullback and R. A. Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[KRL21] D. Kamran, Y. Ren, and M. Lauer. “High-level decisions from a safe maneuver catalog with reinforcement learning for safe and cooperative automated merging”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 804–811.

[KW13] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[KWA20] H. Krasowski, X. Wang, and M. Althoff. “Safe reinforcement learning for autonomous lane changing using set-based prediction”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–7.

[KZL19] D. Kamran, J. Zhu, and M. Lauer. “Learning Path Tracking for Real Car-like Mobile Robots From Simulation”. In: *2019 European Conference on Mobile Robots (ECMR)*. IEEE. 2019, pp. 1–6.

[Leu18] E. Leurent. *An Environment for Autonomous Driving Decision-Making*. `https://github.com/eleurent/highway-env`. 2018. (Visited on 05/15/2021).

[Lev+20] S. Levine, A. Kumar, G. Tucker, and J. Fu. “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[Li+21] B. Li, V. François-Lavet, T. Doan, and J. Pineau. “Domain adversarial reinforcement learning”. In: *arXiv preprint arXiv:2102.07097* (2021).

[Li22] M. Li. “Decision-Making with Reinforcement Learning based on Interactive Behavior Predictors”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2022.

[Lil+15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[Liu+19] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer. “Algorithms for verifying deep neural networks”. In: *arXiv preprint arXiv:1903.06758* (2019).

[LLK20] C. Lazarus, J. G. Lopez, and M. J. Kochenderfer. “Runtime Safety Assurance Using Reinforcement Learning”. In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE. 2020, pp. 1–9.

[LM19] E. Leurent and J. Mercat. *Social Attention for Autonomous Decision-Making in Dense Traffic*. 2019. arXiv: 1911.12250.

[MB19] J. Müller and M. Buchholz. “A risk and comfort optimizing motion planning scheme for merging scenarios”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 3155–3161.

[MC89] M. McCloskey and N. J. Cohen. “Catastrophic interference in connectionist networks: The sequential learning problem”. In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.

[Men22] J. Meng. “Improving Generalization in Reinforcement Learning with Meta-Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2022.

[Mir+18] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker. “High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1–8.

[Mni+15] 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”. In: *nature* 518.7540 (2015), pp. 529–533.

[MS13] V. Mnih and D. Silver. “Playing Atari with Deep Reinforcement Learning”. In: (2013). arXiv: 1312.5602.

[Nau+19] M. Naumann, H. Konigshof, M. Lauer, and C. Stiller. “Safe but not overcautious motion planning under occlusions and limited sensor range”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 140–145.

[OML18] P. F. Orzechowski, A. Meyer, and M. Lauer. “Tackling occlusions & limited sensor range with set-based safety verification”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1729–1736.

[PBS15] E. Parisotto, J. L. Ba, and R. Salakhutdinov. “Actor-mimic: Deep multitask and transfer reinforcement learning”. In: *arXiv preprint arXiv:1511.06342* (2015).

[Pog+18] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr. “Lanelet2: A high-definition map framework for the future of automated driving”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1672–1679.

[Pon+22] C. T. Ponnambalam, D. Kamran, T. D. Simão, F. A. Oliehoek, and M. T. J. Spaan. “Back to the Future: Solving Hidden Parameter MDPs with Hindsight”. In: *Adaptive Learning Agents Workshop at the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. May 2022.

[PSK18] C. F. Perez, F. P. Such, and T. Karaletsos. “Efficient transfer learning and online adaptation with latent variable models for continuous control”. In: *arXiv:1812.03399 [cs, stat]* (Dec. 2018). arXiv: 1812.03399. (Visited on 02/24/2022).

[PSK20] C. Perez, F. P. Such, and T. Karaletsos. “Generalized hidden parameter mdps: Transferable model-based rl in a handful of trials”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5403–5411.

[Rak+19] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. “Efficient off-policy meta-reinforcement learning via probabilistic context variables”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5331–5340.

[Ren21] Y. Ren. “Risk-Aware Decision Making for Automated Driving Under Perception Uncertainty with Model Predictive Control”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2021.

[RMW14] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic back-propagation and approximate inference in deep generative models”. In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1278–1286.

[Rot+18] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. “Promp: Proximal meta-policy search”. In: *arXiv preprint arXiv:1810.06784* (2018).

[RU+00] R. T. Rockafellar, S. Uryasev, et al. “Optimization of conditional value-at-risk”. In: *Journal of risk* 2 (2000), pp. 21–42.

[Sal+20] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer. 2020, pp. 683–700.

[SB18] R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[Sch+17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[Sha+01] L. Sha et al. “Using simplicity to control complexity”. In: *IEEE Software* 18.4 (2001), pp. 20–28.

[SI22] T. Seno and M. Imai. “d3rlpy: An offline deep reinforcement learning library”. In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 14205–14224.

[Sil+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[Sod+22] S. Sodhani, F. Meier, J. Pineau, and A. Zhang. “Block contextual mdps for continual learning”. In: *Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 608–623.

[SSS16] S. Shalev-Shwartz, S. Shammah, and A. Shashua. “Safe, multi-agent, reinforcement learning for autonomous driving”. In: *arXiv preprint arXiv:1610.03295* (2016).

[Taş22] Ö. Ş. Taş. “Motion Planning for Autonomous Vehicles in Partially Observable Environments”. PhD thesis. Karlsruher Institut für Technologie (KIT), 2022. 191 pp.

[Teh+17] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. “Distral: Robust multitask reinforcement learning”. In: *Advances in neural information processing systems* 30 (2017).

[THH00] M. Treiber, A. Hennecke, and D. Helbing. “Congested traffic states in empirical observations and microscopic simulations”. In: *Physical review E* 62.2 (2000), p. 1805.

[Tra+18] T. Tram, A. Jansson, R. Grönberg, M. Ali, and J. Sjöberg. “Learning Negotiating Behavior Between Cars in Intersections using Deep Q-Learning”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), pp. 3169–3174.

[TS18] Ö. Ş. Taş and C. Stiller. “Limited visibility and uncertainty aware motion planning for automated driving”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1171–1178.

[TZS20] Y. C. Tang, J. Zhang, and R. Salakhutdinov. “Worst Cases Policy Gradients”. In: *Conference on Robot Learning*. 2020, pp. 1078–1093.

[VGS16] H. Van Hasselt, A. Guez, and D. Silver. “Deep reinforcement learning with double q-learning”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[WD92] C. J. C. H. Watkins and P. Dayan. “Q-learning”. In: *Machine Learning* 8.3 (1992), pp. 279–292.

[Wer+10] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. “Optimal trajectory generation for dynamic street scenarios in a frenet frame”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 987–993.

[WG08] M. Werling and L. Groll. “Low-level controllers realizing high-level decisions in an autonomous vehicle”. In: *2008 IEEE Intelligent Vehicles Symposium*. IEEE. 2008, pp. 1113–1118.

[Xie+20] A. Xie, D. P. Losey, R. Tolsma, C. Finn, and D. Sadigh. “Learning latent representations to influence multi-agent interaction”. In: *arXiv preprint arXiv:2011.06619* (2020).

[Xu19] P. Xu. “Behaviour Generation for Overtaking with Deep Reinforcement Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.

[Yan+19] J. Yang, B. Petersen, H. Zha, and D. Faissol. “Single episode policy transfer in reinforcement learning”. In: *arXiv preprint arXiv:1910.07719* (2019).

[Yao+18] J. Yao, T. Killian, G. Konidaris, and F. Doshi-Velez. “Direct policy transfer via hidden parameter markov decision processes”. In: *LLARLA Workshop, FAIM*. Vol. 2018. 2018.

[Yu+20] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. “Gradient surgery for multi-task learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5824–5836.

[Zah+17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. “Deep sets”. In: *Advances in neural information processing systems*. 2017, pp. 3391–3401.

[Zha+19] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps". In: *arXiv:1910.03088 [cs, eess]* (2019).

[Zha+20a] A. Zhang, S. Sodhani, K. Khetarpal, and J. Pineau. "Learning robust state abstractions for hidden-parameter block MDPs". In: *arXiv preprint arXiv:2007.07206* (2020).

[Zha+20b] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh. "Robust deep reinforcement learning against adversarial perturbations on state observations". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21024–21037.

Publications by the Author

[Kam+20] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller. "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 1205–1212.

[Kam+21] D. Kamran, T. Engelgeh, M. Busch, J. Fischer, and C. Stiller. "Minimizing safety interference for safe and comfortable automated driving with distributional reinforcement learning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1236–1243.

[Kam+22] D. Kamran, T. D. Simão, Q. Yang, C. T. Ponnambalam, J. Fischer, M. T. Spaan, and M. Lauer. "A modern perspective on safe automated driving for different traffic dynamics using constrained reinforcement learning". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, pp. 4017–4023.

[KRL21] D. Kamran, Y. Ren, and M. Lauer. "High-level decisions from a safe maneuver catalog with reinforcement learning for safe and cooperative automated merging". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 804–811.

[KZL19] D. Kamran, J. Zhu, and M. Lauer. "Learning Path Tracking for Real Car-like Mobile Robots From Simulation". In: *2019 European Conference on Mobile Robots (ECMR)*. IEEE. 2019, pp. 1–6.

[Pon+22] C. T. Ponnambalam, D. Kamran, T. D. Simão, F. A. Oliehoek, and M. T. J. Spaan. "Back to the Future: Solving Hidden Parameter MDPs with Hindsight". In: *Adaptive Learning Agents Workshop at the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. May 2022.

Supervised Theses

[Bus20] M. Busch. “Dynamic Input for Deep Reinforcement Learning at Unsignalized Intersections with Deep Sets”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020.

[Eng21] T. Engelgeh. “Risk-Sensitive Behavior Generation for Vehicles at Partially Occluded Intersections Using Attention-Based Distributional Reinforcement Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2021.

[Li22] M. Li. “Decision-Making with Reinforcement Learning based on Interactive Behavior Predictors”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2022.

[Men22] J. Meng. “Improving Generalization in Reinforcement Learning with Meta-Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2022.

[Ren21] Y. Ren. “Risk-Aware Decision Making for Automated Driving Under Perception Uncertainty with Model Predictive Control”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2021.

[Xu19] P. Xu. “Behaviour Generation for Overtaking with Deep Reinforcement Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.