Title
Efficient Learning Methods in Mixed Autonomy Traffic

Author
Kreidieh, Abdul Rahman

Publication Date
2022

Peer reviewed|Thesis/dissertation

Efficient Learning Methods for Mixed Autonomy Traffic

by

Abdul Rahman Kreidieh

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexandre M. Bayen, Chair
Emeritus Professor In-Residence Alexander Skabardonis
Professor Claire Tomlin

Fall 2022

Efficient Learning Methods for Mixed Autonomy Traffic

Abstract

Efficient Learning Methods for Mixed Autonomy Traffic

by

Abdul Rahman Kreidieh

Doctor of Philosophy in Engineering - Civil and Environmental Engineering

University of California, Berkeley

Professor Alexandre M. Bayen, Chair

Automated driving systems are expected to play a critical role in the future of transportation. With fast reaction times, vehicle-to-vehicle communication, and the potential for socially optimal driving behaviors, automated vehicles may serve a central role in improving driving conditions within existing road networks, reducing the prevalence of traffic congestion and enabling fast and energy-efficient driving. Generating behaviors that produce such effects in real world settings, however, is no trivial task. In particular, when coupled with human drivers in mixed-autonomy settings, coordination between human-driven and automated vehicles becomes increasingly delicate, and motivates the need for new and advanced tools for solving these tasks.

Through the research outlined in this document, we aim to identify efficient methods for learning congestion-mitigating control strategies that can be employed by automated vehicles in partially automated road networks. Recent advances in deep reinforcement learning have highlighted the potential of said techniques in producing control strategies that match or outperform classical approaches on a variety of decision making and control tasks. The applicability of similar approaches to mixed-autonomy traffic control, however, is hindered by a number of challenges. For one, exploration in these settings is difficult, as individual actions may not influence the flow of traffic until multiple timesteps in the future. In addition, the process of modeling and executing simulations of realistic traffic flow networks at the level of individual vehicles is a difficult and computationally costly endeavor. Through techniques such as hierarchical learning, imitation from experts, and robust learning in simplified tasks, we hope to design data-efficient methods for generating control strategies for automated vehicles that are transferable to the real world.

To my parents, Jamal and Rima.

To my sister, Maya.

Thank you all for your support.

Contents

Acknowledgments

First and foremost, I want to thank my advisor, Alexandre Bayen, for his mentorship and for guiding me through my PhD. His commitment and continuous support has made this PhD possible and shaped me into a better researcher. Thank you also to my committee members, Alexander Skabardonis and Claire Tomlin for their support and guidance. And finally, in terms of mentors, thank you Steven Glaser for affording me the opportunity of exploring the realm of research within Berkeley during my undergraduate years, and for his support and encouragement to pursue my Masters, and eventually PhD, here.

I am grateful to have had the opportunity to collaborate with an amazing group of students, faculty, and researchers during my PhD. For the work in this thesis, I want to thank Cathy Wu, Eugene Vinitsky, Zhe Fu, Glen Berseth, Kathy Jang, and Nathan Litchlé. I would also like to thank all of the undergraduate students I worked with during my PhD including Yibo Zhao, Samyak Parajuli, Gilbert Bahati, Brandon Trabucco, Kanaad Parvate, and Nishant Kheterpal. In addition, thank you to my other labmates as well for the fun Friday and weekend lunches, the lab retreats to Maui and Yosemite, and all the interesting and joyful conversations: Alexander Keimer, Yashar Zeiynali Farid, Alben Rome Bagabaldo, Joy Carpio, Théophiles Cabannes, Fangyu Wu, Fang-Chieh Chou, Yiling You, Marsalis Gibson, Ashkan Yousefpour, Jessica Lazarus, Arwa AlAnqary, Sulaiman Almatrudi, Saleh Albeik, Shuxia Tang, and others I may have have missed. And thank you to all the member of the CIRCLES consortium, who are far too numerous to list without me missing one, for playing a role in possibly the most ambitious and largest project I will ever being a part of, and for all the important lessons you taught me along the way. Despite a turbulent few years, we still managed to deploy 100 automated vehicles simultaneously in an attempt to reduce traffic congestion. After all those years of hard work, the largest traffic experiment of its kind to date, and a job well done!

Finally, I'd like to thank my parents, Jamal Kreidieh and Rima Mehio, and my sister, Maya, for all their years of love and support. I am truly blessed to have a support structure in my family that ensured I would never go wanting neither financially nor emotionally. Thank you also to all the people unnamed here whose interactions and incident encounters made my time here fuller and taught me much about the richness of human experiences. And thank Anna for all the trips we took across the United States, for the fun times we had together, and for exposing me to a part of this world that I was otherwise blind to.

CHAPTER 1

Introduction

1.1 Early developments in vehicle autonomy

Automated driving systems have long been a topic of fascination and debate. Interest in vehicle autonomy dates back to the 1920s and 30s, when radio-controlled "phantom autos" took to the streets, captivating audiences and drawing thousands of spectators in cities across the United States [85, 1]. Excitement towards autonomy would then grow and dissipate over the next few decades, spurred in part by visions of expressways connecting the country and allowing drivers to travel safely and easily without loss of speed [14, 108]. This vision would culminate in various experiments conducted in 1950s and 60s, whereby carmakers used inductive wires and coils placed on the road and coupled with simple feedback loops to guide vehicles in a driverless manner. These demonstrations, while garnering attention, where infeasible at scale, and as such, automated vehicles were relegated to popular culture once again.

Vehicle autonomy would not truly see a renaissance until the 1980s, whereby advancements in sensing modalities and information processing allowed for researchers to encode more intricate behaviors into their cars. Onboard devices including cameras and radar systems were coupled with, at the time, advanced machine learning tools for feature extraction, edge detection, and motion capture to allow vehicles to perform simple yet astonishing feats such as road following and obstacle avoidance [41, 171]. These events marked a pivot in automated vehicle technology, suggesting that machine learning, as opposed to standard control mechanisms alone, would play an important role in the advancement of the field. This trend would then continue throughout the 1990s and early 2000s, with seminal competitions such as the DARPA Grand [23] and Urban [24] challenges pushing the state-of-the-art in machine learning for terrain analysis, and demonstrating that such systems can enable vehicles to navigate long distances in unstructured landscapes with no human feedback.

1.2 Machine learning in automated driving

Today, machine learning systems have permeated nearly every challenge surrounding automated driving. Deep learning, in particular, has become an increasingly popular tool. In comparison to similar machine learning methods reliant on handcrafted features for data analysis, deep neural networks allow for the automation of the feature generation procedure, with representations of input observations learned and stored internally within a model's parameters [90]. This approach, when coupled with the abundance of data and compute resources in recent years, began to outperform its counterparts in a variety of vision [84] and language-processing [67] tasks, and quickly found its way to other problems as well.

In the context of automated driving, deep learning methods have become the standard through which vehicles perceive and understand their surroundings. High-dimensional inputs such as those provided by HD cameras or LiDAR are segmented via a variety of tools (e.g [53, 134, 37, 133, 210]) into canonical "objects" whose categorizations, be they other vehicles, pedestrians, or terrain, may readily be identified. This provides automated vehicles a scenic understanding of their environment and is then used to either predict the movements of agents surrounding a given vehicle [4, 206, 91, 140], or coupled with feedback control strategies to dictate safe and effective ways for said vehicle to maneuver towards its destination [125]. For a more thorough review of each such technique, we refer the readers to [56].

More pressing to the present document, however, deep learning and artificial intelligence methods have similarly begun to *directly* define how vehicles act and navigate within their surroundings, thereby negating the need for extensive rule-based controllers or additional perception and prediction models. This paradigm, whereby sensory information is directly mapped to control outputs, is often termed *end-to-end learning*, and appears in a variety of different forms [211]. Most prominently, researchers and practitioners have attempted to exploit supervised learning techniques, particularly in the context of imitation learning, to map sensory inputs to steering and throttling commands performed by human drivers in similar situations. This approach has a long history of use dating back to the 1980s with the autonomous car ALVINN [132], and has, as with other machine learning applications, benefited greatly in recent years from an abundance of human-labeled data [34, 11, 62].

In addition, reinforcement learning techniques have also begun to see a growth in popularity. Reinforcement learning (RL), described in Chapter 2, is a class of machine learning algorithms that, through interactions with an environment, attempts to learn the parameters of a policy that optimize its performance on a given sequential decision making task. These methods, coupled with the expressive capabilities of deep neural networks, have produced remarkable results on a variety of control and strategy-driven tasks [114, 152, 94], thereby motivating similar research in the context of automated driving. Today, research into RL for automated driving is conducted primarily in simulation settings (e.g. [43]) with few exceptions [76]. Studies in this regard look to applications such as car-following [40, 208], lane changing [193, 112, 69], and merging [192, 61], with an emphasis on learning behaviors that are both comfortable [70] and safe [149, 190] for passengers. The vision here: the deployment of fleets

of vehicles that would facilitate a more productive use of the transit time and reduce the ill effects of driving inattentiveness or stress.

A far less appreciated aspect of driving in machine learning systems, however, has been the implications of such systems on the mobility or efficiency of existing road networks. Automated vehicles, as with a multitude of robotics applications, do not exist and react to their environment in isolation, but rather their environment reacts back, often in strange and unexpected ways. For automated vehicles, these effects may readily be seen in factors such as traveling time [165] and fuel-economy [187], particularly as the market penetration rate for said devices progresses over time. As such, design choices and algorithms used to define how automated vehicles act must be approached with apprehension. For instance, RL methods that solve simply for comfort or safety may be insufficient when looking to the long-term ramifications of such maneuvers on the emergence and propagation of congested states of traffic. This dilemma is the primary focus of this document, and is discussed in further detail below.

1.3 Thesis overview and contributions

In this document, we aim to determine whether RL methods are suitable for deriving control strategies capable of improving the state of traffic in typical road networks. We begin in **Chapter 2** by introducing the prominent machine learning tools and algorithms used to solve different mixed-autonomy traffic control problems. Then, in **Chapter 3**, we discuss some of the physical and dynamical phenomena that result in the emergence of congested states of traffic, and present tools developed by myself and others designed to efficiently and flexibly integrate simulations of such phenomena with state-of-the-art RL algorithms. From the straightforward adaptation of RL, we identify several factors that hinder the efficacy of such algorithms in defining truly high-performing traffic control policies. Within this document, we narrow our focus to two main challenges, listed below:

Challenge I: Sparse/delayed feedback. In mixed-autonomy traffic control settings, meaningful events occur over prolonged periods of time as oscillations in traffic states propagate slowly through a network. As a result, actions by AVs often have a delayed effect on metrics of improvement or success and, at times, require hundreds of steps to meaningfully affect the reward or feedback signals assigned to learning policies. The delayed nature of this feedback exacerbates the credit assignment problem within mixed-autonomy tasks, making reasoning on whether a specific action contributed to or mitigated the propagation of congestion difficult. This, in practice, prevents standard RL techniques from generating meaningful policies without relying on reward engineering. Reward engineering, however, can be challenging to finetune and often results in undesirable learned responses.

Challenge II: Cost of modeling and simulating human drivers. In addition to the difficulties of exploration and credit assignment, the process of both modeling and simulating

large-scale transportation networks presents a unique set of challenges from a reinforcement learning perspective. These challenges take on two forms. First, we find that readily available tools for simulating traffic are generally inefficient at producing numerous instantiations of a given road network. This is problematic from a machine learning perspective, as algorithms often require millions of interactions with a simulation environment to produce potentially meaningful results. As such, simulating traffic alone becomes one of the largest bottlenecks when exploring new and interesting problems. Next, as the size of explored networks increases, creating and finetuning models that accurately reproduce the dynamics of such a network becomes increasingly more intricate. This, once again, is potentially problematic, as inaccurate reconstructions of traffic exacerbate the sim-to-real gap within machine learning problems and, as a result, may invalidate any policies learned in such a simulation.

After introducing the above challenges, we present solutions capable of addressing each in practice. These solutions are the focus of the remainder of this document and consist of three parts:

Part I: Learning across long time horizons. In this first part, we address the challenge of learning through sparse or delayed feedback. To do so, we explore the use of hierarchies, prominent models for enabling structured exploration in complex long-term planning problems. For the class of problems explored here, however, we find that standard algorithms for the concurrent training of policy layers within a hierarchy fail to effectively assign higher-level latent goals, and instead quickly regress to greedier, locally optimal solutions. Drawing connections between these and similar challenges in mixed cooperative-competitive multi-agent RL problems, in **Chapter 4** we introduce a simple yet effective technique for loss-sharing among policy layers within a hierarchy. Finally, we demonstrate that such an approach enables hierarchies to solve various challenging long-term control tasks more efficiently than prior state-of-the-art approaches and without the need for explicit reward engineering.

Part II: Exploiting expert demonstrations. Next, we explore methods for enhancing the data efficiency of learning procedures within large-scale traffic simulations. This solution, similar to that above, draws from the challenges of credit assignment in standard RL approaches. In particular, in large-scale networks consisting of multiple interacting agents, exploration becomes increasingly delicate, as the objective an RL agent attempts to maximize is heavily influenced by the behaviors of other learning agents within the environment. This interaction has a destabilizing effect on learning and, in practice, often prevents or slows down forward progression. To this, supervised learning procedures, and specifically imitation learning, may serve a beneficial role. Unlike within RL, imitation learning procedures provide more stable and direct targets for agents to achieve and are unaffected by the exploration-exploitation dilemma. However, the choice of target behavior to imitate plays an important role. In particular, imitating human driving behaviors, which has become the standard in multiple driving applications, is likely insufficient to produce traffic regulating policies,

as humans themselves act as contributors to the proliferation of congestion. Instead, in **Chapters 5 and 6**, we construct a set of expert policies that, in simulation, exploit knowledge of global states of traffic to homogenize the movement of vehicles towards their desirable uniform-flow equilibrium. We then demonstrate that imitation learning techniques can successfully capture such behaviors solely using local observations and in a more data-efficient manner.

Part III: Generalizing to complex-to-model dynamics. Finally, in the third part of this document, we explore solutions to the remaining challenge described above, i.e. the sensitivity of learned behaviors to variations in dynamical responses within simulated and target networks. In **Chapter 7**, we consider two networks in which stop-and-go waves are induced through different mechanisms. Within this problem setup, we demonstrate that generalizable behaviors may be learned in simplified (ring road) representations of traffic by introducing somewhat similar, albeit stochastic, perturbations to vehicle states within the simpler problem. The findings here suggest that much research conducted in the absence of intricate traffic models may be made transferable to complex road networks through simple considerations into robustness and the dynamical mismatch between both problems.

Finally, in **Chapter 8**, we conclude by recapping some of the findings within this document and present future research directions through which future studies into mixed-autonomy systems may improve upon what is presented here.

CHAPTER 2

Review of Machine Learning Frameworks

In this chapter, we introduce the terminology and some of the operating principles behind the machine learning algorithms explored within this document. In particular, we narrow our focus to two general classes of algorithms: *reinforcement learning* and *imitation learning*. These two fields of study have been at the center of a plethora of profound and substantive leaps within the field of artificial intelligence in recent years. When coupled with large quantities of data and highly expressive neural models of varying architectures [68, 84, 204, 182], these techniques have succeeded in producing behaviorally "intelligent" agents capable of achieving a number of impressive feats. Prominent examples include achieving human-level performance in games with delayed rewards [114], outplaying grand champions in intellectually-demanding two-player games, [151, 152, 145], as well as solving a variety of interesting control and navigation problems[1]. These achievements and more motivate much of the work tackled within this study.

The remainder of this chapter is composed of three sections. In Section 2.1, we introduce Markov decision processes, the mathematical framework through which sequential decision-making processes are understood in the context of reinforcement learning. Next, in Section 2.2 we introduce reinforcement learning (RL), and present the mathematical concepts surrounding the some of the most commonly used RL algorithms. Finally, in Section 2.3 we present imitation learning, a method for the supervised learning or optimal behaviors, and introduce a popular algorithm for addressing training instabilities arising from domain mismatches in demonstration data.

2.1 Markov decision processes

Markov decision processes (MDPs), at a high level, provide a mathematically intuitive framing for interactive sequential decision-making problems. They define the manner through which states and actions from the perspective of a reward-maximizing agent are linked to one

[1]Examples of some of the control and navigation tasks that have been solved by machine learning systems are provided throughout this document when relevant.

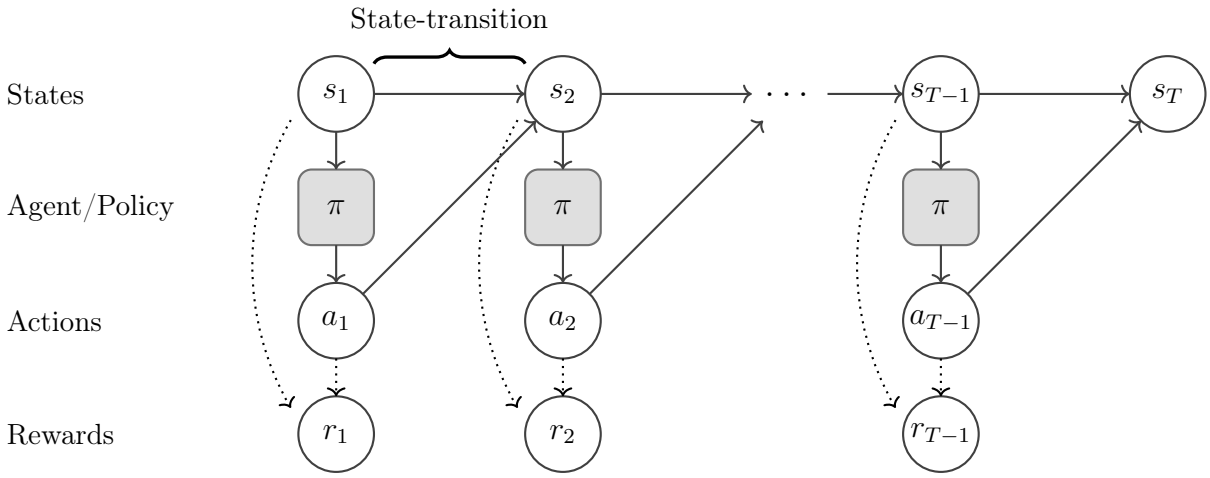


Figure 2.1: A graphical representation of the relationships between states, actions, and rewards within a Markov decision process.

another and evolve as time progresses. The feedback assigned to these agents in the form of rewards then serves to distinguish preferable and poor-performing (states, actions) pairs, a factor which is then exploited by learning agents as they search for optimal goal-reaching behaviors.

Formally put, a discrete-time MDP [17] is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$, where $\mathcal{S}$ is a (discrete or continuous) state space, $\mathcal{A}$ a (discrete or continuous) action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ a state-transition probability satisfying the Markov property, and $r : \mathcal{S} \to \mathbb{R}$ a reward function. Within an MDP, an *agent* receives sensory inputs in the form of *states* $s_t \in \mathcal{S}$ from the environment and then interacts with its environment by performing *actions* $a_t \in \mathcal{A}$ within it. These actions are defined by a policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ parameterized by θ, and influence the distribution of next states via the state-transition probability $p(s_{t+1}|s_t, a_t)$. This interaction then produces a new state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ with which the agent once again interacts with and repeats the process for a predefined *time horizon* of length T. The resulting sequence of (state, action) pairs form a *trajectory* τ, defined as:

$$\tau = (s_1, a_1, s_2, a_2, \ldots, a_{T-1}, s_T) \tag{2.1}$$

and within which individual nodes are graphically linked to one another as in Figure 2.1. The Markovian relationship between timesteps allows us to define simple representations of the probability $p_\theta(\tau)$ of witnessing a given trajectory conditioned of the actions specified by π_θ. In particular, they allow us to factor the effects of individual (state, action) pairs on the distribution of entire trajectories, producing a Markov chain of the form:

$$p_\theta(\tau) = \rho(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{2.2}$$

where $\rho(s_1)$ is the distribution of initial states. This factorization is used in a number of RL algorithms presented later on.

From here, distributions of preferable behaviors may be identified indirectly from the distributions of trajectories they produce. To do so, trajectories are assigned preference dependent on the discounted sum of the sequence of rewards $\{r_t = r(s_t, a_t) \mid t = 1, 2, \ldots, T\}$ assigned to (state, action) pairs within them. The objective function $J(\theta)$ for such a problem is then:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \right] \tag{2.3}$$

where actions $a_t \sim \pi_\theta(a_t|s_t)$ are sampled from π_θ, and $0 \leq \lambda \leq 1$ is a *discount factor*.

Finally, we define the optimal behavior for a policy π_θ within an MDP as the set of parameters θ^* that satisfy the condition:

$$\theta^* = \arg\max_\theta J(\theta) \tag{2.4}$$

2.1.1 Example: Mixed-autonomy ring road

Markov decision processes appear in multiple different forms in the context of mixed-autonomy traffic and traffic flow regulation as a whole. We present here one such formulation for a standard mixed-autonomy traffic problem. This example is adapted in part from the work of Wu et al. [202]. We describe the solution to this MDP once employed in an RL loop in Chapter 3.2.2.

Consider a platoon of N vehicles placed sequentially in a single lane circular track of circumference L (Figure 2.2). This is conceptually a reconstruction of the seminal Sugyiama ring experiment [156]. In this setting, drivers, when instructed to drive at a fixed speed, fail to consistently maintain such speeds for long periods of time. Instead, small fluctuations in speeds by individual drivers compound and expand as those behind them break harder to maintain safe driving conditions. This event, repeated ad infinitum, results in the formation of stop-and-go oscillations typical to many highway networks (Figure 2.3). The reasoning behind why drivers may break harder that their predecessors, however, arises largely from human factors in driving[2], and as such may be addressed by AVs in the near future.

We wish to replicate the above behavior in simulation so that agents may interact with the problem and attempt to answer the question: What can an AV do to dissipate stop-and-go congestion? With this question arises a number of design decisions that must be made around the dynamics of individual vehicles what learning agents sense and how they interact with their surroundings. These are all choices that must, in part at least, be defined within our MDP. We present one possible formulation of the MDP below.

[2]We describe this phenomenon further in Chapter 3.1.1.

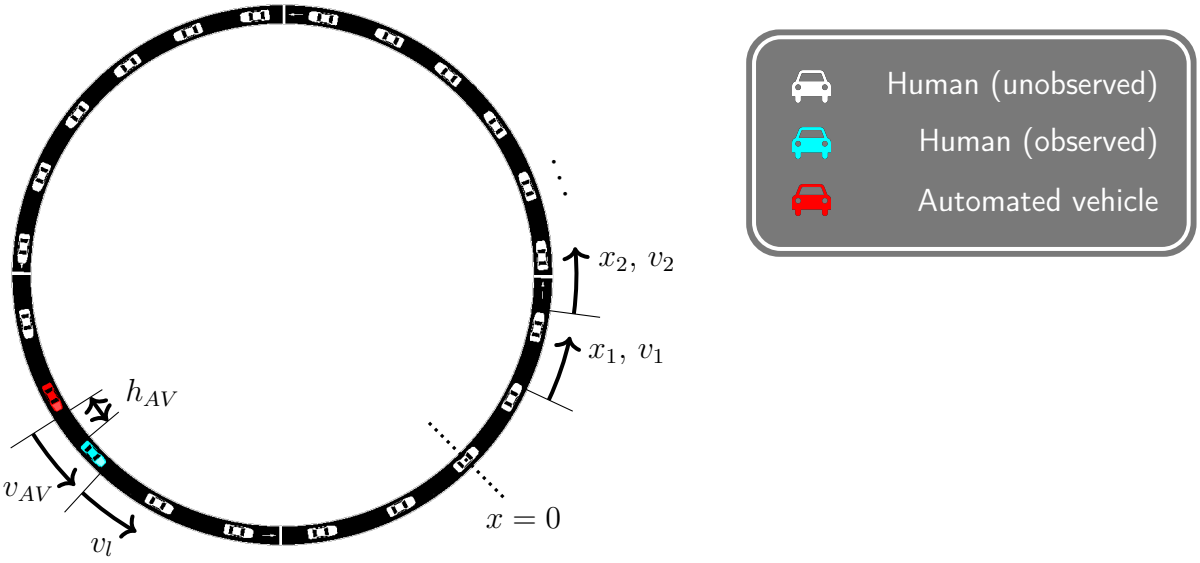


Figure 2.2: An illustration of the mixed-autonomy ring road environment.

States In the absence of additional lanes or perturbation induced by on-ramps or off-ramps, the state of the network at time t may in its simplest form be defined in its entirety by the positions x_i^t and speeds v_i^t of every vehicle i at timestep t. The full state is then:

$$s_t = \left[x_1^t, v_1^t, x_2^t, v_2^t, \ldots, x_N^t, v_N^t\right]^T \tag{2.5}$$

In many settings, however, inputs of this type are impractical to assign to learning agents, as vehicles are incapable of realistically perceiving the state of all other vehicles in their vicinity. Instead, agents in real-world settings must condition their actions on local, delayed, and/or noisy *observations* o_t as opposed to the full state s_t. This is the premise behind *partially observable* Markov decision processes, or POMDPs[3], and yields some of the most critical challenges when attempting to learn practical mixed-autonomy traffic control strategies. We delve further into these challenges in the following chapter.

For the purposes of this problem, we choose to denote the observation o_t as the state of the ego vehicle as that of the vehicle immediately preceding it:

$$o_t = \left[v_{AV}^t, v_l^t, h_{AV}^t\right]^T \tag{2.6}$$

where v_{AV}^t is the speed of the automated vehicle, v_l^t is the speed of its lead vehicle, and h_{AV}^t is the space gap between the two. For simplicity, we will generically refrain from distinguishing between states and observations throughout this document.

[3]In addition to the typical components that define an MDP, POMDPs also consist of two additional attributes, namely Ω, a set of observations, and $\mathcal{O} : \mathcal{S} \times \Omega \to \mathbb{R}_+$, an observation probability distribution.

Actions We aim to identify the longitudinal (car-following) behaviors that automated vehicles can adopt to maintain idealized driving conditions within ring roads. Accordingly, we define the actions agents perform within this environment as a bounded vector $a_t \in \mathbb{R}^{N_{AV}}_{[a_{\min}, a_{\max}]}$ denoting the desired acceleration by each AV, where $N_{AV} = 1$ is the number of vehicles controlled by the agent and $a_{\min}$ and $a_{\max}$ are bounds on the assigned accelerations.

State-transition The dynamics, or state-transition, of ring road problems have been widely studied previously and as such may be readily defined from prior work. Typically, the dynamics of ring road problems are represented via systems of ODEs, which when discretizing in time, result in a state transition function defined as:

$$\begin{cases} x_i^{t+1} = x_i^t + v_i^t \Delta t + \frac{1}{2} u_i^t \Delta t^2 \\ v_i^{t+1} = v_i^t + u_i^t \Delta t \end{cases}, \qquad i \in \{1, \ldots, N\} \tag{2.7}$$

where Δt is a simulation step size and u_i^t is the external force, or acceleration, induced by each vehicle i. This acceleration response for simulated "human" drivers must be chosen such that, in the absence of autonomy, perturbations propagate and grow between adjacent human drivers until stop-and-go oscillations form. A popular model for recreating such behaviors is the *Intelligent Driver Model*, or IDM [174]. Following this model, the acceleration response for any human-driven vehicle i is defined by its bumper-to-bumper space headway h_i, velocity v_i, and relative velocity with the preceding vehicle Δv_i as:

$$u_i^t = f_{\text{IDM}}(v_i, \Delta v_i, h_i) = a \left[1 - \left(\frac{v_i}{v_0} \right)^{\delta} - \left(\frac{s^\star(v_i, \Delta v_i)}{h_i} \right)^2 \right] + \epsilon \tag{2.8}$$

where ϵ is an exogenous noise term designed to mimic stochasticity in human driving behaviors and $s^\star$ is the desired headway of the vehicle denoted by:

$$s^\star(v_i, \Delta v_i) = s_0 + \max \left(0,\ v_i T + \frac{v_i \Delta v_i}{2\sqrt{ab}} \right) \tag{2.9}$$

and s_0, v_0, T, δ, a, b are fixed parameters.

Figure 2.3 depicts the spatio-temporal behavior of IDM-controlled vehicles on the ring road, where N is set to 22 and L is set to 260 m. Within this figure (called a *time-space diagram*) each line segment represents the trajectory following by an individual vehicle in relation to a fixed origin point, and is colored in accordance with the speed the vehicle drove at a given time. Seen here, vehicles following this model do succeed at reconstructing the type of congestion expected within ring roads, with the dark diagonal regions that appear further in time depicting the heart and the stop-and-go wave once it has formed.

Finally, to model the behavior of a single automated vehicle interacting the above dynamical system, we replace the response of the AV with the action $a_t \sim \pi_\theta(a_t | o_t)$ assigned by the policy π_θ. The acceleration response for this vehicle is then

$$u_{AV}^t = a_t \tag{2.10}$$

Rewards For the present task, we are interested in generating driving behaviors that divert vehicles from their oscillatory / stop-and-go behavior and towards a more homogeneous state of traffic. This desirable state, at its limit, is termed the *uniform flow* equilibrium, and for a system of human-driven vehicles following a car-following model f is defined as the point at which move at a constant speed v^* and with constant spacing h^*, such that:

$$f(v^*, 0, h^*) = 0 \tag{2.11}$$

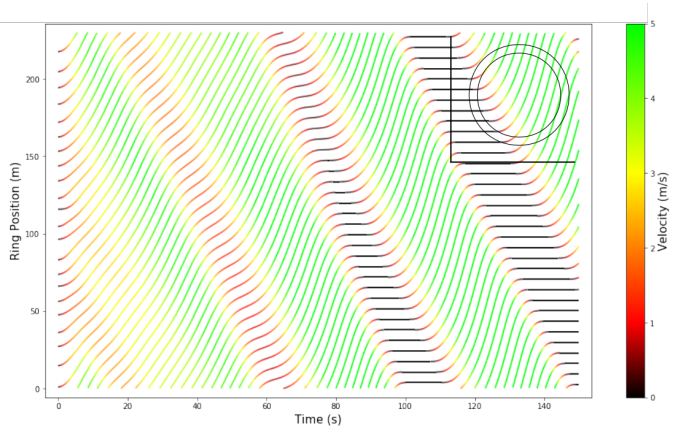


Figure 2.3: Human-driven ring road dynamics

Reward functions are the primary means through which positive behaviors such as these may be incenticized within a learning procedure. The process of assigning appropriate reward functions, however, in nontrivial in many settings. One common tradeoff that emerges here is that between the density of a given reward and the optimally of behaviors it produces. Dense rewards, in particular, offer more direct feedback that accelerate learning within most common learning algorithms, but may result in negative externalities within the learned behavior if not attended to carefully. We will return to this topic in the following chapter, and note that this is a point of concern for this problem as well. However, for this task, we note that optimal, uniform-flow, states of traffic are achieved when all vehicles are moving very fast and none are accelerating greatly. We may accordingly define an appropriate reward function for this problem as:

$$r(s_t, a_t) = \sum_{i=1}^{N} v_i^t + c_1 \cdot |a_t| \tag{2.12}$$

where c_1 is a fixed parameter.

2.2 Reinforcement learning

Having defined Markov decision processes, a framework through which sequential decision-making problems may be framed, we now turn our attention to reinforcement learning (RL), the class of algorithms through which these problems may be solved. As previously stated, the objective of an RL agent within an MDP is to find the parameters for a policy π_θ for which interactions with the MDP attain the largest possible expected return. In RL, this is done through a process akin to trial-and-error, a process that is generally composed of two steps. In the first step, agents, initialized with some choice of parameters[4], navigate

[4]The choice of initial parameters within such a training procedure is a nontrivial and active topic of research [54, 87, 6]. We refrain from discussing this further as it is outside the scope of this research.

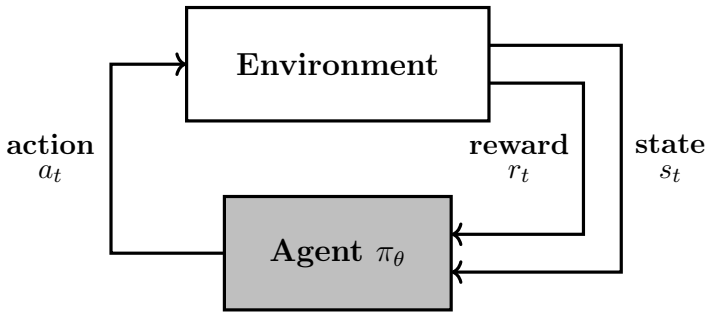


Figure 2.4: An illustration of reinforcement learning.

and explore their environment in a attempt to assign *values* distinguishing which states and actions produce higher expected returns. Next, information gathered in the previous step is used to define directions in the parameter space for which expected returns are predicted to be preferable, and a gradient update step is taken is taken in that direction, i.e.

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta) \tag{2.13}$$

where α is the gradient step size, often referred to as the *learning rate*. The process is then repeated until predefined final conditions are met.

Several methods have been proposed to efficiently estimate the gradient within Eq. (2.13). These methods are differentiated along a wide variety of categories, including via whether prior experiences are recycled [114, 150, 103, 52, 60] or disposed of [199, 147, 146] between gradient update steps, as well as on whether models of the dynamics of the world are generated to help augment the learning procedure [160, 39, 93, 72]. We present below some of the most prominent algorithms employed throughout this document. A more detailed description of each algorithm may be found in the citations provided, and for a more thorough overview of RL, we refer the reader to the following book by Sutton and Barto [161].

2.2.1 Policy gradient methods

Policy gradient methods (also referred to as direct policy gradient or on-policy methods) provide a straightforward approach for estimating gradients for the objective function from collections of trajectories collected on-policy with respect to π_θ. In particular, expanding the expectation in Eq. (2.3), policy gradient methods try to produce estimates for the term:

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \tag{2.14}$$

where $r(\tau)$ is the cumulative discounted return of a trajectory τ. The challenge here arises from the presence of the dynamics term p_θ, a term that is generally unavailable and may difficult to model for arbitrary problems. To compute this gradient in a model-free fashion, Williams et al. [199] demonstrate that, through some reparametrization and an exploitation

of the definition of probabilities of a trajectories within an MDP, the dynamics component p may be disentangled and removed from the remainder of the estimate, resulting in an equivalent representation of the form:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \right) \right] \tag{2.15}$$

The gradient from here is then typically computed via Monte Carlo estimation. That is:

$$\nabla J_\theta(\theta) \approx \frac{1}{N_T} \sum_{i=1}^{N_T} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^{T} \gamma^t r(s_{i,t}, a_{i,t}) \right) \tag{2.16}$$

where N_T is the total number of executed trajectories between gradient update steps. This policy gradient algorithm is called "REINFORCE". Further details on this algorithm may be found in the following citation [199].

Trust Region and Proximal Policy Optimization The above policy gradient algorithm provides a strong benchmark for estimating gradients in the direction of optimal policies, but exhibits several underlying limitations when applied to complex problems. One prominent restriction within this algorithm surrounds the validity of assigned gradients for large step sizes α. This is an important factor within any optimization procedure and plays a critical role within policy gradient methods. Importantly, due to the on-policy nature under which gradients are estimated, assigned gradients may be locally valuable within the vicinity of θ, but become increasingly less relevant once pushed farther beyond this point. Moreover, if excessively large step sizes are taken, important behaviors learned from prior policy updates may be forgotten and then must be relearned, a factor which further hinders the efficiency of this learning procedure.

Most popularly, the above challenge is addressed through the use of trust region constraints. In particular, more advanced policy gradient methods restrict the effects of large learning rates by constraining the mismatch between the behavior of a policy π_θ prior to and after individual policy update. The distance between these two models is quantified via the Kullback–Leibler (KL) divergence, and results in an optimization problem of the form:

$$\begin{aligned} &\arg\max_\theta \quad J(\theta) \\ &\text{subject to} \quad \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right) \right] \leq \delta \end{aligned} \tag{2.17}$$

where θ_{old} is are the parameters prior to a given policy update and δ is a maximum constraint imposed on the KL divergence. This optimization problem, under idealistic settings does ensure monotonically improvements to the expected return between policy updates, but is difficult and costly to compute. To this, multiple algorithms have been proposed to solve efficiently solve the above problem, and solutions to this are at the core of the contributions of both the *Trust Region Policy Optimization* (TRPO) [147] and *Proximal Policy Optimization* (PPO) [146] algorithms, which are used frequently within this document. We refer the reader to the provided citations for more detail on each algorithm.

2.2.2 Actor-critic methods

Another prominent class of reinforcement learning algorithms are what are known as actor-critic methods. Actor-critic methods, in comparison to direct policy gradient methods before them (Section 2.2.1), solve two key problems: 1) the need for a stochastic policy $\pi_\theta(a|s)$, and 2) the need for on-policy samples to compute policy gradients. Instead, actor-critic methods recycle prior experiences, storing them within *replay buffers* of varying attributes, and sampling them whenever new policy updates need to be performed. As a result, methods of this form, also referred to as *off-policy* RL methods, often perform more sample-efficiently than their on-policy counterparts, but rely more heavily on function approximators to compute appropriate policy gradients, and as such must be handled with care.

Actor-critic algorithms, similar to most RL algorithms, follow a two-step approach of estimating the values of different trajectories and then updating the choice of actions by a policy in the direction of higher gains. In this first step, termed the *value iteration* phase, agents use samples collected both on-policy an off-policy to improve the predictive accuracy of an action-value function, or *critic*, $Q_\phi(s, a)$. This function, at its optimum, defines the cumulative discounted return an agent would expect to experience if it were to take an action a within a starting state s, i.e.

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) \middle| s_1 = s, a_1 = a \right] \tag{2.18}$$

The estimates of the critic in the second, or *policy iteration*, step is then used to shift the distribution of actions by the policy, or *actor*, π_θ to those which are predict to produce higher expected returns. We present the details behind both steps below.

Value iteration We begin by introducing the process through which approximated action-value functions may be generated within an offline RL paradigm. Specifically, we here are interested in attempting to parameters ϕ for a function Q_ϕ that approximately matches the distribution of expected returns from a *true* Q-function Q^π. The objective is then to find the parameters ϕ that minimize the loss function:

$$L(\phi) = \mathbb{E}_{s_t, a_t \sim p_\theta} \left[\left(Q_\phi(s, a) - Q^\pi(s, a) \right)^2 \right] \tag{2.19}$$

Within the context of RL, estimates of this form are typically performed through a process termed temporal difference (TD) learning. To incrementally update the predictive accuracy of Q_ϕ, TD learning procedures exploit the recursive relationship between action-value estimates (state, action) pairs ordered sequentially in time. This relationship is know as the Bellman equation, and for deterministic actors results in a sequence of nested functions of the form:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_t, a_t, s_{t+1} \sim p_\theta} \left[r(s_t, a_t) + \gamma \cdot Q^\pi(s_{t+1}, \pi_\theta(s_{t+1})) \right] \tag{2.20}$$

From here, reasonable updates to the predictive accuracy of Q_ϕ are achieved by bootstrapping prior action-value estimates within this nested function, and utilizing experienced

(state, actions) pairs to ground predictions to returns witness via interactions with the environment [195]. The updated objective function for the approximate Q-function then becomes:

$$L(\phi) = \mathbb{E}_{s_t, a_t, s_{t+1} \sim p_\theta} \left[\left(Q_\phi(s_t, a_t) - y_t \right)^2 \right] \tag{2.21}$$

where

$$y_t = r(s_t, a_t) + \gamma \cdot Q_{\phi'}(s_{t+1}, \pi_{\theta'}(s_{t+1})) \tag{2.22}$$

and θ' and ϕ' are delayed target parameters for the actor and critic policies, respectively, and are used to stabilize the training procedure [103].

Policy iteration Once differentiable estimates for the Q-function have been assigned, gradients for the objective function may be defined by computing the gradient of the Q-function for different actions and applying the chain rule to propagate these gradients across the choice of policy parameters as well. The gradient applied to the policy is then:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p_\theta} \left[\nabla_a Q_\phi(s, a)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s) \right] \tag{2.23}$$

The proof behind this derivation may be found here [150] and is at the core of the DDPG [103] and TD3 [52] algorithms, both of which are used regularly throughout this document.

2.3 Imitation learning

Finally, we introduce imitation learning, a process for learning optimal sequential-decision making behaviors from expert demonstrations. Imitation learning techniques, unlike within reinforcement learning, do not rely on reward functions or other signals to evaluate the validity of individual actions. Instead, in imitation learning (Figure 2.5), policies are provided with labeled examples of desirable or optimal responses for given sensory inputs and attempt to derive mappings from states to actions that generally match them. More concretely put, consider a policy π_θ parametrized by θ, and let $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ denote a dataset of (state, action) demonstrations provided by an expert policy π_E. In this setting, the objective of an IL algorithm is find the optimal parameters θ^* that solve the supervised learning problem:

$$\theta^* = \arg\min_\theta \left[\mathbb{E}_{s_i, a_i \sim \mathcal{D}} \left[\mathcal{L}(\pi_\theta(s_i), a_i) \right] \right] \tag{2.24}$$

where $\mathcal{L}(\cdot, \cdot)$ is a distance metric used to compute the difference between predicted and expert actions. For continuous action problems, this loss is typically something similar to a mean-squared error, while for discrete action problems cross-entropy losses are generally used.

Data Aggregation (DAgger) Imitation learning, while similar to other supervised learning techniques, is importantly different as a result of the consecutive nature of prediction, or actions, within tasks. As a result, predictions by a policy are conditioned to past predictions

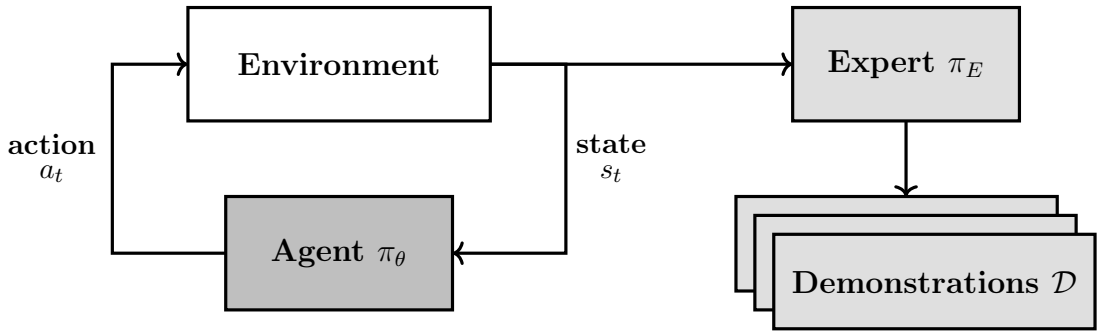


Figure 2.5: An illustration of imitation learning.

and subsequently negate the assumption of independent and identically distributed (IID) samples within standard supervised learning. This causes standard supervised learning to suffer from a phenomenon known as covariate shift, whereby errors by an agent compound as it executes its policy. The effects of this are not uncommon within imitation learning, and occur primarily when demonstrations provided by an expert are sparse or insufficiently diverse. Prominent examples within the field of driving include policies, trained to imitate the driving behaviors of humans, drifting off-road after prolonged periods of time [132, 21] or failing to maintain safe car-following distances [11]. To mitigate this, on-policy techniques in IL such as DAgger [136, 137] correct for covariate shift by iteratively sampling states $\{s_i\}_{i=1}^{N_s}$ from the agent's policy and providing corrective labels within these states from the expert $\{a_i\}_{i=1}^{N_s}$. The new samples are then aggregated to the original dataset $\mathcal{D} = \mathcal{D} \cup \{(s_i, a_i)\}_{i=1}^{N_s}$ and the optimization procedure is repeated. We employ this technique when deriving decentralized, locally observable policies in Part II of the document.

CHAPTER 3

Learning Traffic Regulation Policies in Simulation

Having previously defined reinforcement learning and other techniques for generating sequential decision-making behaviors in simulation, we now turn our focus to the main problem of interest: traffic congestion. In particular, in this chapter we explore methods for reconstructing human-driven factors leading to the proliferation of congestion in simulation, and use such simulations to attempt to answer the question: What behaviors may automated vehicles adopt to dissipate congestion in mixed-autonomy settings?

The following chapter is composed of three sections. In Section 3.1, we introduce the microscopic traffic flow models and tools used to simulate human-driven factors in congestion. Next, in Section 3.2 we discuss how such tools may be readily integrated with existing RL algorithms, and provide results demonstrating the efficacy of RL in solving a difficult mixed-autonomy traffic control task. Finally, in Section 3.3 we present initial attempts at standardizing some key problems in mixed-autonomy traffic, and benchmark the performance of common RL algorithms in solving such tasks. The results from this final section point to important challenges when attempting to adapt RL to the domain to traffic control. We end by introducing two prominent challenges, and for the remainder of this document present models and algorithms that mitigate them in practice.

The content in the last two sections in this chapter is adapted largely from [202, 184].

3.1 Microscopic traffic flow models

Traffic flow models describe the dynamical interplay between multiple agents within a given network. They provide concrete mathematical framings that define the rationale through which agents navigate their surroundings in an effort to reach a desired destination. Examples of common decision-making behaviors captured by such models include car-following, lane changing, route assignment, and others[1]. Models of this form yield a number of underlying benefits to the transportation community. As a means of reconstructing past or typical events in traffic, they provide a venue through which our understanding of the contributing factors

[1]To delve deeper into this topic, we refer the readers to the following book by Treiber and Kesting [177].

to congestion may be unified and tackled in a systematic manner. Examples include the phenomenon of capacity drop near highway on- and off-ramp merges [32, 89] from which theory into optimal ramp metering strategies may be defined [127, 128], as well as other negative effects resulting from suboptimal lane changing and car-following behaviors. Conversely, as a means of predicting future events, models of this form may also be used in a closed-loop manner to identify behaviors that achieve desirable short-term objectives. Examples in this context include model-predictive or optimal control strategies that solve for safety [75, 190] or other metrics often synonymous with some form of efficiency or mobility [97, 57, 181].

In this document, we focus primarily on *microscopic* simulations of traffic, that is, simulations of traffic at the level of individual vehicles. These are in contrast to macroscopic models [102, 135, 129, 8, 207], within which the evolution of traffic is modeled as aggregate quantities (notable flows, densities, and average speeds) defined across spatial sections within a network. The aggregation of vehicles states across segments introduces a gas-kinematic interpretation of traffic whereby individual vehicles are no longer disambiguated, but rather move as fluids in a pipe. This perspective reduces the computational burden of such models, but hinders their fidelity in modelling the effects of individual driving choices on future states of traffic. Conversely, advancements in compute have reached a point whereby many driving behaviors may be explored relatively quickly in microscopic simulation settings and extracted via data-driven methods based on the system-level responses they produce. We will return to the machine learning components in the following section, but begin now by introducing some of the models and tools used within this study.

3.1.1 Car-following models

Car-following models define the longitudinal motion of vehicles driving behind one another in a single lane. Within car-following models, the acceleration a_i of a vehicle i is typically defined as a function $a_i = f(s_i, v_i, v_l)$ of the speed of the ego vehicle v_i, the speed of the lead vehicle $v_l = v_{i-1}$, and the space gap between the two $s_i = x_{i+1} - x_i - l_i$, where l_i is the length of vehicle i. Figure 3.1 provides a visual interpretation for each of these terms. The functional form $f(\cdot)$ is also referred to as an *acceleration* or *follow-the-leader* model, and is designed to capture common human factors and incentives in driving, including maintaining a safe driving distance proportional to the agent's speed, driving at, or attempting to drive at, some fixed desired speed, and responding to local fluctuations in traffic conditions with some reaction time or delay. Examples of prominent car-following models include Newell's car-following model [121], the Optimal Velocity Model (OVM) [10, 74], the Intelligent Driver Model (IDM) [174, 3], among others.

Once a car-following model is defined across multiple vehicles, the motion of a *string* or *platoon* of vehicles is provided by the coupled ordinary differential equations:

$$\begin{cases} \dot{x}_i(t) = v_i(t) \\ \dot{v}_i(t) = f(s_i(t-\tau), v_i(t-\sigma), v_l(t-\kappa)) \end{cases} \qquad i = 1, \ldots, N \tag{3.1}$$

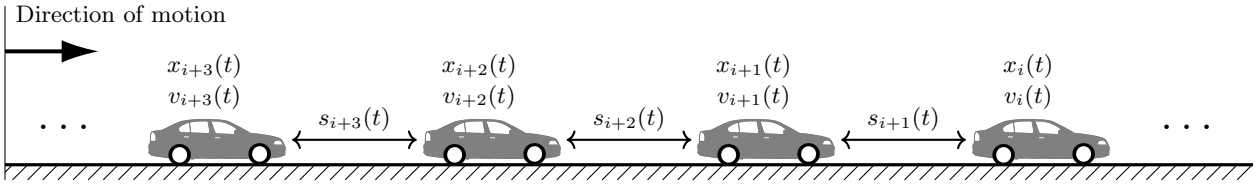


Figure 3.1: Variables defining the motion of vehicles in single lane car-following settings.

where τ, σ, and κ are time delays, and N is the number of vehicles in the platoon. The boundary conditions imposed on the leading vehicle ($i = 1$) may be defined in a number of different manners. Most prominently, this is done either through the use of periodic boundary conditions for circular (ring road) tracks whereby vehicle $i = 1$ follows vehicle $i = N$ [30], or via a simulated trajectory which $i = 1$ vehicle then follows [110, 189, 100].

A dynamical system of this form reaches its *uniform flow* steady-state equilibrium when all vehicles drive at a constant speed v^* and with uniform inter-vehicle spacing s^*. From the perspective of the system of equations presented in Eq. (3.1), this is achieved when the following condition is met:

$$f(s^*, v^*, v^*) = 0 \tag{3.2}$$

for all vehicles within the network. This equilibrium state is characterized by fast driving speeds and reduced instances of sharp accelerations, and as such is considered a desirable state of traffic from the perspectives of mobility and energy-efficiency. In generic highway settings, however, this equilibrium is typically unstable. Instead, small fluctuations arising from heterogeneities in human driving behaviors produce higher amplitude oscillations by following vehicles as they break harder to avoid unsafe settings. This process then repeats itself, increasing the magnitude of such oscillations until, for large enough platoons, a *stop-and-go* wave is formed, i.e a scenario in which drivers regularly and sharply transition between what appear to be free-flow conditions to a state of highly dense and slow-moving traffic. Figure 3.2 provides a visual depiction of such a response. The phenomenon leading to these every-growing oscillations is known an *string instability*. A formal mathematical definition of of this phenomenon can be found in the following article [164].

The phenomenon of string unstable driving can be witnessed both numerically and empirically in a variety of different network structures. In ring road settings, for instance, stop-and-go oscillations within well-defined car-following models present themselves as stable limit cycles that bifurcate from the uniform-flow equilibrium when conditions become sufficiently dense[2] [122, 124, 123]. The formation of stop-and-go waves within ring roads is an important component of car-following behaviors, as it points directly to the role humans play in the proliferation of traffic congestion. In particular, it points to the notion that, even in the absence of external disturbances such as those induced by various network-imposed bottlenecks, humans themselves may still shift their aggregate behaviors to one representative

[2]We provide numerical results demonstrating the formation of stop-and-go waves in ring roads in Chapter 2.1.1.

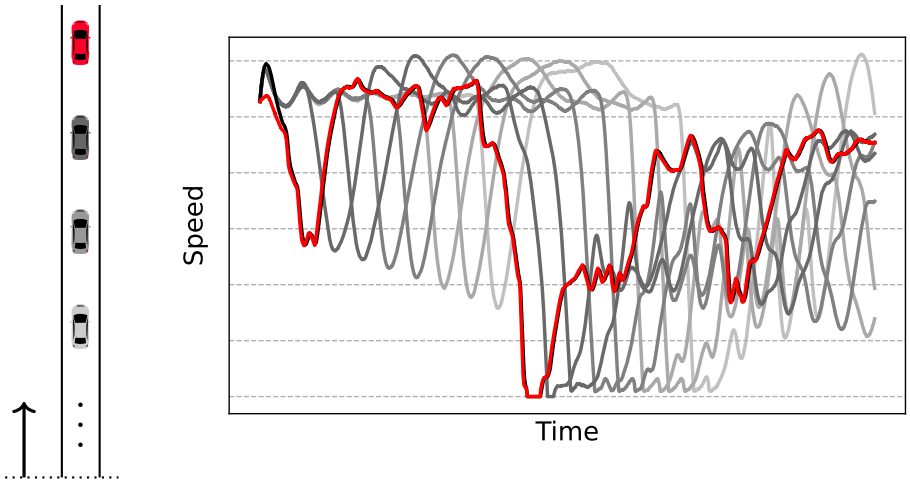


Figure 3.2: Sample string unstable behavior, modeled as a response to a leading vehicle profile (in red) captured under real-world driving conditions [100]. Oscillations by following vehicles continuously grow in magnitude, pointing to the phenomenon resulting in stop-and-go congestion on typical highways.

of traffic congestion. This behavior was empirically reconstructed by the seminal ring road experiment conducted by Sugiyama et al. in 2008 [156] as well, further highlighting the prevalence of string instabilities in human driving, and demonstrating that the emergence of stop-and-go oscillations from such a behavior is no mere simulation artifact.

The notions of string (and similar) instabilities and the effects they have on the growth and propagation of congestion plays an important role in constructing traffic-smoothing policies in simulation, which will be returned to frequently throughout this document.

3.1.2 Additional dynamics and simulation tools

Car-following models alone are often incapable of capturing the richness and diversity of traffic. Instead, in networks with multiple lanes or spatial variations in network structure, other models are needed. In general, most remaining dynamical interactions are captured via lane-changing models for lateral movements and other decision models for discrete-choice situations such as route assignment or navigating into, for instance, an intersection. Models such as these often follow similar structures but share far less consensus in the research community. Lane change models, for one, are typically categorized under *discretionary* lane changes for decisions involving local improvements in utility, e.g. moving to a faster lane, and *mandatory* lane changes when such action are needed to maintain a given route, e.g. to reach an off-ramp. These responses are replicated via a plethora of techniques, ranging from incentive-based models [88, 77, 141], game-theoretic formulations of interactions between adjacent vehicles [167, 191], and, more recently, predictive models trained to mimic large quantities of human data [91, 143, 206]. Similarly, many other decisions are typically modeled

as some variation of a discrete choice model [18] and applied spatially when an agent is near the location where a decision needs to be made.

To reproduce the above and other unmentioned dynamical responses in simulation, we in this document rely mainly on existing microscopic traffic simulation tools. Tools of this nature are varied and widespread within the transportation community, and include proprietary tools such as Aimsun [27], Matsim [186], and Vissim [46], as well as open-source software such as SUMO (Simulation of Urban MObility) [80]. Traffic microsimulators have been broadly accepted in the transportation community as an acceptable prelude to the empirical evaluation of several complex automated tasks such as cooperative adaptive cruise control (CACC) [111] and mixed-autonomy traffic flow control [155], and crucially provide powerful and flexible methods for configuring network structure and control inputs based on user preference. In this document, we use the simulator SUMO due in large part to its flexibility and ease of access and integration across multiple platforms.

3.2 Reinforcement learning in mixed-autonomy traffic

Microscopic traffic flow models and simulation tools provide a rich and exciting landscape for the analysis of different traffic control mechanisms, but are limited in the avenues they offer for search and optimization. Instead, through typical control theoretic studies researchers trade the complexity of traffic flow models (and, thus, their realism) for the tractability of analysis, with the ultimate goal of designing optimal and practical controllers.

Reinforcement learning, as discussed in the previous chapter, has been very effective at solving similarly complex and seemingly intractable problems in other domains. However, previously, no frameworks have existed for the evaluation of the efficacy of reinforcement learning under a variety of traffic control paradigms. In this Section, we introduce, Flow, an open-source framework with aims to combine state-of-the-art microsimulations of traffic with modern reinforcement learning tools. In Section 3.2.1 we introduce the internal structure and components within Flow that enable the flexible composition of different traffic control problems. Then, in Section 3.2.2, we demonstrate the efficacy of such a tool in solving a widely studied mixed-autonomy traffic control problem.

3.2.1 Flow: A modular learning framework

Flow (see Figure 3.3) is a modular learning framework which enables the composition of diverse mixed autonomy traffic scenarios for study with deep reinforcement learning. Scenarios conform to a MDP interface, and are composed of several reusable modules used to define various components of the MDP. In particular, Flow is comprised of the following modules:

Network The network specifies the physical road layout, in terms of roads, lanes, length, shape, roadway connections, and additional attributes. Example include multi-lane circular tracks, or the structure described by importing a map from OpenStreetMap (see Figure 3.4

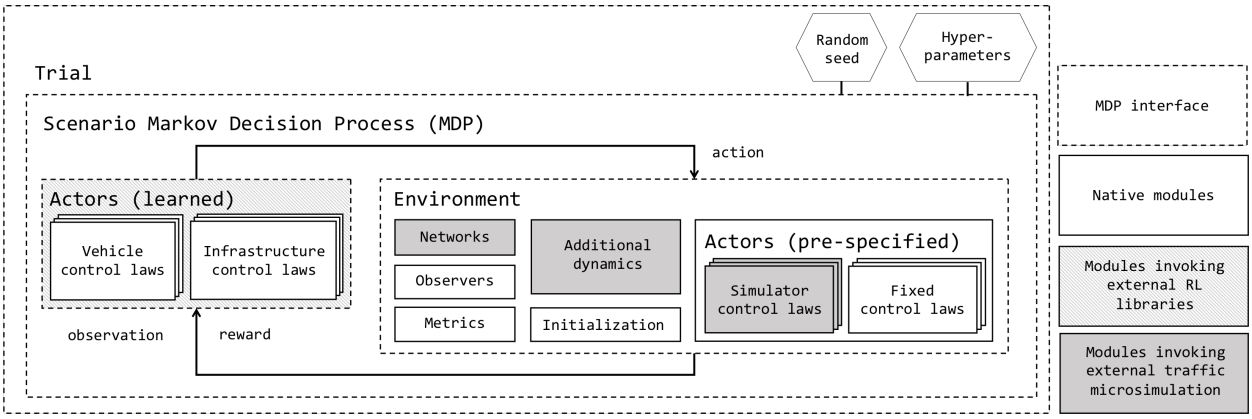


Figure 3.3: Flow process diagram.

for examples of supported networks). Several of these examples will be used in demonstrative experiments in Sections 3.2.2 and 3.3.

Actors The actors describe the physical entities in the environment which issue control signals to the environment. In this document, actors are primarily characterized as automated vehicles; however, other possible actors may include pedestrians, bicyclists, traffic lights, roadway signs, toll booths, as well as other transit modes and infrastructure.

The behaviors of the actors are defined by **control laws**, function mapping from states or observations to actions which, in turn, influences the transitions between states. All actors require a control law, which may be pre-specified or learned. For instance, a control law may represent a human driver, an autonomous vehicle, or even a set of vehicles. That is, a single control law may be used to control multiple vehicles in a *centralized control* setting. Alternatively, a single control law may be used by multiple actors in a *shared parameter control* setting.

Observer The observer describes the mapping from environment state to features observed by the actor(s). The output of the observer is taken as input to the control law, described above. For example, while the state may include the position, lane, velocity, and accelerations of all vehicles in the system, the observer may restrict access to only information about local vehicles and aggregate statistics, such as average speed or length of queue at an intersection.

Dynamics The dynamics module consists of additional submodules which describe different aspects of the system evolution, including vehicle routes, demands, stochasticity, traffic rules (e.g., right-of-way), and safety constraints.

Metrics The metrics describe pertinent aggregated statistics of the environment. The reward signal for the learning agent is a function of these metrics. Examples include the

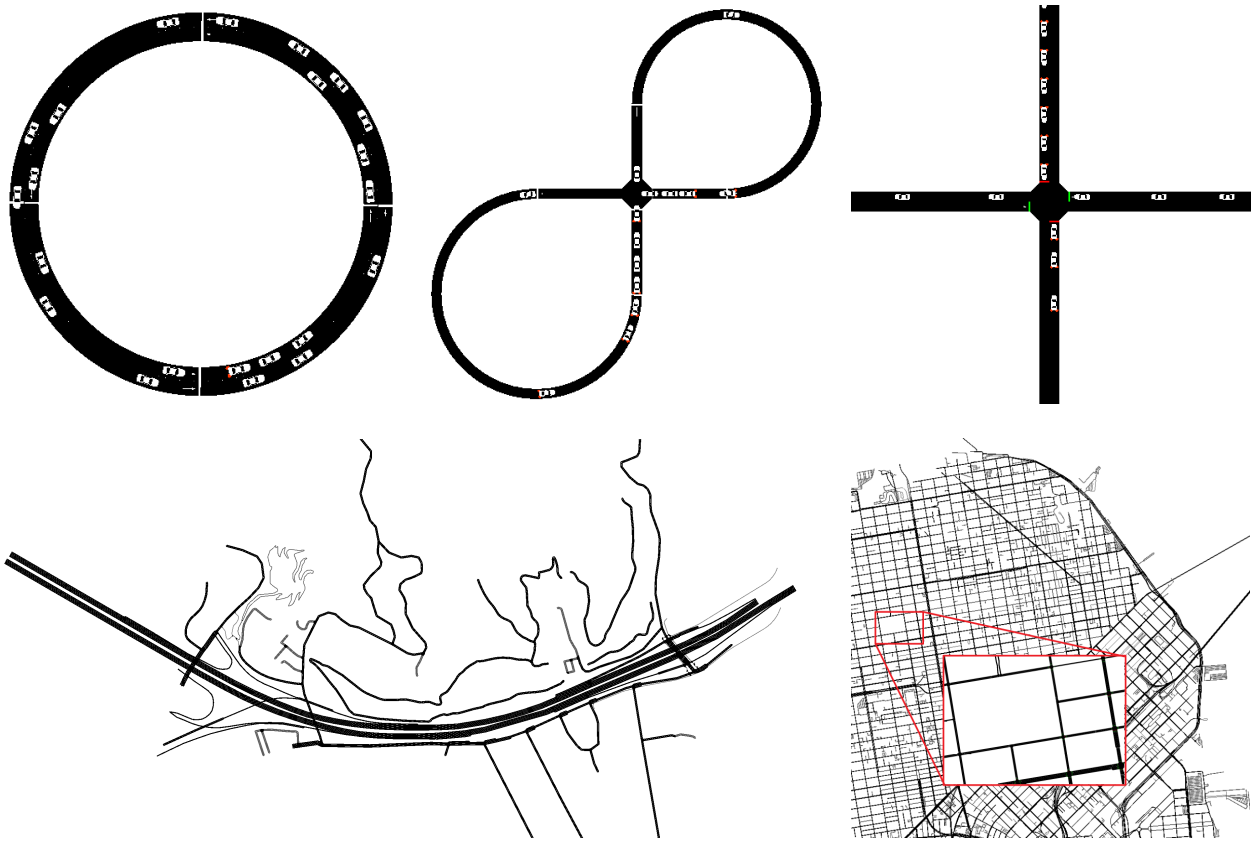


Figure 3.4: Sample network modules supported by the Flow framework. Networks vary from simple, closed-loop systems with relatively fixed dynamics to more complex, multi-modal problems with predefined network structures provided via OpenStreetMap[3].

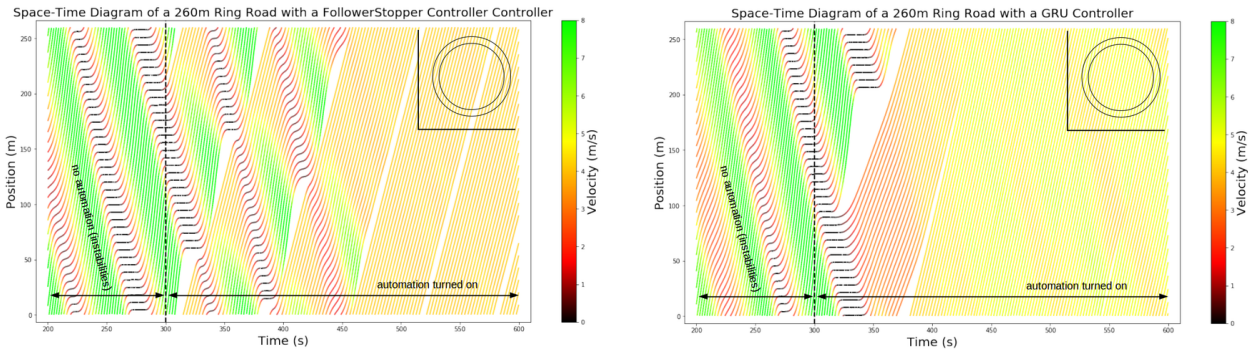
overall (average) velocity of all vehicles and the number of (hard) braking events.

Initialization The initialization describes the initial configuration of the environment at the start of an episode. Examples include setting the position and velocity of vehicles according to different probability distributions.

Architecture and implementation

The implementation of Flow is open source and builds upon open source software to promote access and extension. The open source approach further aims to support the further development of custom modules, to permit the study of richer and more complex environments, agents, metrics, and algorithms. The implementation builds upon SUMO [80] for vehicle and

[3]OpenStreetMap: `https://www.openstreetmap.org/`



(a) Follower Stopper [155]

(b) Reinforcement learning [202]

Figure 3.5: Spatio-temporal response of different mixed-autonomy behaviors in a single lane circular track with a circumference of 260 m.

traffic modeling, Ray RLlib [44, 99] for reinforcement learning, and OpenAI gym [22] for the MDP.

As typical in reinforcement learning studies, an **environment** encodes the MDP. The environment facilitates the composition of dynamics and other modules, stepping through the simulation, retrieving the observations, sampling and applying actions, computing the metrics and reward, and resetting the simulation at the end of an episode. A **generator** produces network configuration files compatible with SUMO according to the network description. The generator is invoked by the experiment upon initialization and, optionally, upon reset of the environment, allowing for a variety of initialization conditions, such as sampling from a distribution of vehicle densities. Flow then assigns control inputs from the different control laws to the corresponding actors, according to an **action assigner**, and uses the TraCI library to apply actions for each actor. Actions specified as accelerations are converted into velocities, using numerical integration and based on the timestep and current state of the experiment.

Finally, Flow is designed to be inter-operable with classical feedback and model-based control methods for evaluation purposes. That is, the learning component of Flow is optional, and this permits the fair comparison of diverse methods for traffic control. We demonstrate the adaptability of Flow to both learn and hand-crafted control laws in the following subsection.

3.2.2 Example: Mixed-autonomy ring road

We now return to the example introduced in Chapter 2.1.1, where a single automated vehicle (AV) attempts the dissipate the formation and propagation of stop-and-go waves in variable-length ring roads. As discussed in Section 3.1.1, this oscillatory (stop-and-go) behavior emerges largely from string unstable interactions between human drivers, which cause perturbations in driving speeds to compound and grow over time. In this setting, AVs

must regulate their spacing in such a manner as to prevent the continued growth of such oscillations.

To reconstruct the above problem in Flow, we assign the network, actors, observers, and so on as described in the previous section and in accordance with the MDP structure outlined in Section 2.1.1. Exact implementation details can be found at: `https://github.com/flow-project/flow-lab/tree/master/flow-framework`. The AVs in our system then execute actions following parameterized control laws whose values are trained using policy gradient methods. For all experiments, we use the Trust Region Policy Optimization (TRPO) [147] method for learning the control law, linear feature baselines as described in Duan, et al. [44], discount factor $\gamma = 0.999$, and step size 0.01. Each of the results presented here are collected from numerical experiments conducted on three Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz processors for six hours. A total of six million samples were collected during the training procedure.

Performance

We compare the performance of the learned control strategy against human drivers' behaviors in fully human-driven settings as well as the behaviors of hand-craft control laws presented in [155]. Videos and additional results for each controller are available at: `https://sites.google.com/view/ieee-tro-flow`.

Figure 3.6 demonstrates the performance of the different learned and hand-crafted controllers. Here, we begin to observe key findings pointing to the efficacy of RL methods in solving mixed-autonomy traffic control tasks. In particular, we find that learned control behaviors (in partially observed settings) match the optimal uniform-flow conditions very closely for all explored densities, thereby eliminating congestion. This is true for both densities in which the policy was trained as well as regions in which the policy is evaluated in a zero-shot manner, pointing to the generalizability of the control law. The hand-crafted controllers, on the other hand, only dissipate stop-and-go traffic at densities less than or equal to their calibration density (less congested settings).

Next, Figure 3.5 shows the space-time curves for all vehicles in the system using a variety of control laws. We observe that, in this particular density, while both control strategies succeed in dissipating stop-and-go congestion, the *FollowerStopper* control law leaves much smaller openings in the network (smaller headways) when compared to the learned control law, as can be seen by the large white portion of the RL plot. The large gaps in this setting emerge from the necessity for acceleration penalties within the reward function design. For this particular reward configuration, we find that large small acceleration penalties do not provide agents with strong enough short-term feedback to explore and identify the long-term benefits of driving at uniform speeds, while large penalties uniformly generate the aforementioned response. This delicate balance between exploration and reward function design motivates the work outlined in Chapter 4.

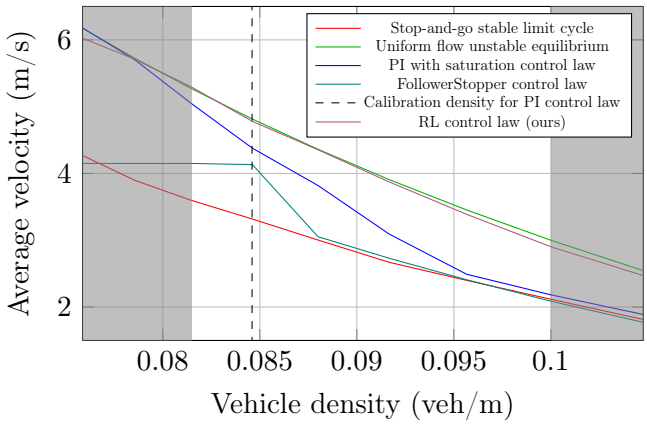


Figure 3.6: The performance of the learned (RL) and hand-crafted (FollowerStopper and PI Saturation) control laws for various vehicle densities are averaged over ten runs at each evaluated density. Also displayed are the performance upper and lower bounds, derived from the unstable and stable system equilibria, respectively.

3.3 Benchmarks for RL in mixed-autonomy traffic

In this section, we extend the numerical analysis conducted in Section 3.2.2 in an attempt to better define the scope and capabilities of RL in solving diverse traffic control problems. We introduce a number of benchmarks in Section 3.3.1 aimed at tackling different mixed-autonomy tasks, and discuss some of the findings and limitations that emerge from such benchmarks in Section 3.3.2.

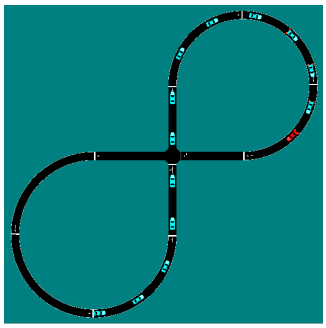
3.3.1 Proposed benchmarks

We construct a suite of benchmarks aimed at tackling different problems within the context of mixed autonomy traffic. These benchmarks, seen in Figure 3.7, are detailed below.

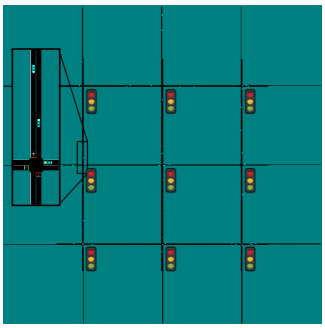
Figure eight: optimizing intersection capacity

The figure eight network (Figure 3.7a) acts as a closed representation of an intersection. In a figure eight network containing a total of 14 vehicles, we witness the formation of queues resulting from vehicles arriving simultaneously at the intersection and slowing down to obey right-of-way rules. This behavior significantly reduces the average speed of vehicles in the network. In a mixed-autonomy setting, a portion of vehicles are treated as CAVs with the objective of regulating the flow of vehicles through the intersection in order to improve system-level speeds. The components of the MDP for this benchmark are defined as follows:

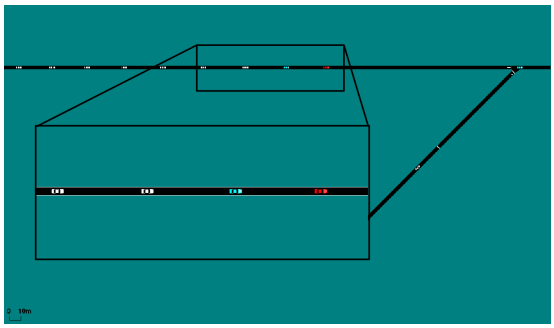
- States: The state consists of a vector of speed and positions for each vehicle in the network, ordered by the position of each vehicle, $s := (v_i, x_i)_{i=0:k-1} \in \mathbb{R}^{2k}$, where k is the number of



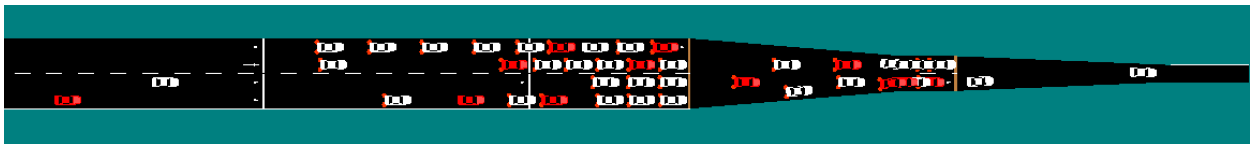
(a) Figure Eight



(b) Grid



(c) Merge



(d) Bottleneck

Figure 3.7: Network configurations for the various benchmarks presented. In each of the figures, vehicles in red are automated (controlled by the RL agent), in blue are human-driven and directly observed in the state space, and in white are human-driven an unobserved.

vehicles. Note that the position is defined relative to a pre-specified starting point.

- Actions: The actions are a list of accelerations for each CAV, $a \in \mathbb{R}^n_{[a_{\min}, a_{\max}]}$, where n is the number of CAVs, and $a_{\min}$ and $a_{\max}$ are bounds on the accelerations.
- Reward: The objective of the learning agent is to achieve high speeds while penalizing collisions. Accordingly, the reward function is defined as follows:

$$r = \max\left(||v_{\text{des}} \cdot \mathbb{1}^k||_2 - ||v_{\text{des}} - v||_2,\ 0\right) \ / \ ||v_{\text{des}} \cdot \mathbb{1}^k||_2 \tag{3.3}$$

where v_{des} is an arbitrary large speed used to encourage high speeds and $v \in \mathbb{R}^k$ is the speeds of all vehicles in the network.

Merge: controlling shockwaves from on-ramp merges

The merge network (Figure 3.7c) highlights the effect of disturbances on vehicles in a highway network. Specifically, perturbations resulting from vehicles arriving from the on-merge lead to the formation of backwards propagating stop-and-go waves, thereby reducing the throughput of vehicles in the network. In a mixed-autonomy setting, a percentage of vehicles in the main highway are tasked with dissipating the formation and propagation of stop-and-go waves from locally observable information. Moreover, given the open nature of the network, the total number of CAVs within the network may vary at any given time. Taking these into account, we characterize our MDP as follows:

- States: The state consists of the speeds and space gaps of the vehicles preceding and following the CAVs, as well as the speed of the CAVs, i.e. $s := (v_{i,\text{lead}}, v_{i,\text{lag}}, h_{i,\text{lag}}, h_{i,\text{lag}}, v_i) \in \mathbb{R}^{n_{\text{RL}}}$. In order to account for variability in the number of CAVs (n_{CAV}), a constant n_{RL} term is defined. When $n_{\text{CAV}} > n_{\text{RL}}$, information from the extra CAVs are not included in the state. Moreover, if $n_{\text{CAV}} < n_{\text{RL}}$ the state is padded with zeros.

- Actions: The actions consist of a list of bounded accelerations for each CAV, i.e. $a \in \mathbb{R}^{n_{\text{RL}}}_{[a_{\min}, a_{\max}]}$. One again, an n_{RL} term is used to handle variable numbers of CAVs. If $n_{\text{CAV}} > n_{\text{RL}}$ the extra CAVs are treated as human drivers and their actions are updated using a human-driver model. Moreover, if $n_{\text{CAV}} < n_{\text{RL}}$, the extra actions are ignored.

- Reward: The objective in this problem is, once again, improving mobility, either via the speed of vehicles in the network or by maximizing the number of vehicles that pass through the network. Accordingly, we augment the reward function presented in Eq. (3.3) as follows:

$$r = \max\Big(||v_{\text{des}} \cdot \mathbb{1}^k||_2 - ||v_{\text{des}} - v||_2,\ 0\Big) \ / \ ||v_{\text{des}} \cdot \mathbb{1}^k||_2 - \alpha \sum_{i \in CAV} \max\left[h_{\max} - h_i(t), 0\right] \quad (3.4)$$

  The added term penalizes small headways among the CAVs, thereby discouraging dense states that lead to the formation of stop-and-go traffic.

Grid: improving traffic signal timing schedules

The grid network (Figure 3.7b) is an idealized representation of a city with a structure similar to Manhattan. This task highlights issues that arise in coordination of traffic light signals, particularly questions of partial observability and the scaling of RL algorithms with action dimension. Solutions to this problem will generate new traffic light control schemes that minimize the average per-vehicle delay while inducing some measure of fairness.

Vehicles enter at the corners of the grid. For simplicity of the problem, vehicles travel straight on their path. Each intersection has a traffic light that allows vehicles to flow either horizontally or vertically. If the light is green, it transitions to yellow for two seconds before switching to red for the purpose of safety.

The components of the MDP for this benchmark are defined as follows:

- States: Speed, distance to intersection, and edge number of each vehicle. The edges of the grid are uniquely numbered so the travel direction can be inferred. For the traffic lights we return 0,1 corresponding to green or red for each light, a number between [0, t_{switch}] indicating how long until a switch can occur, and 0,1 indicating if the light is currently yellow. Finally, we return the average density and speed of each edge.

- Actions: A list of numbers $a = [-1, 1]^n$ where n is the number of traffic lights. If $a_i > 0$ for traffic light i it switches, otherwise no action is taken.

- Reward: The reward presented in Eq. (3.3) is once again used for this task. Here the use of the 2-norm induces fairness, as a more equal distribution of speeds produces a larger reward.

Bottleneck: maximizing throughput in a bottleneck structure

The bottleneck network (Figure 3.7d) is an idealized representation of the lane-reduction bottleneck witnessed on Oakland-San Francisco Bay Bridge. On the Bay Bridge, 16 non-HOV lanes narrow down to eight and subsequently to five. In our model, lanes are instead reduced from $4N$ to $2N$ to N, where N is a scaling factor. This system exhibits the phenomenon known as *capacity drop* [138], where the throughput of vehicles through the bottleneck experiences a sharp drop in magnitude once the inflow of vehicles to it surpass a critical value. Given this inflow-outflow relation, the goal of a learning agent within this task restrict or otherwise modify the flow of vehicles within the bottleneck to avoid the triggering of a capacity drop.

In what follows, we term each distinct segment of road an *edge*. For each edge, users can specify for each edge whether it is observed and/or controlled, and how many segments it is divided into: we refer to this as an *edge-segment*. Although the observation space, action space, and reward can easily be modified, the provided environment operates on the following MDP:

- States: The mean positions and speeds of human drivers for each lane for each edge segment. The mean positions and speeds of the CAVs on each segment. The outflow of the system in vehicles per/hour over the last 5 seconds.
- Actions: For a given edge-segment and a given lane, the RL action shifts the maximum speed of all the CAVs in the segment from their current value. By shifting the max-speed to higher or lower values, the system indirectly controls the speed of the RL vehicles.
- Reward: $r_t = \sum_{i=t-\frac{5}{\Delta t}}^{i=t} \frac{n_{\text{exit}}(i)}{\frac{5}{\Delta t * n_{\text{lanes}} * 500}}$ where $n_{\text{exit}}(i)$ is the number of vehicles that exited the system at time-step i. Colloquially, this is the outflow over the last 5 seconds normalized by the number of lanes, n_{lanes} and a factor of 500. This is to keep the scale of the reward in line with the other benchmarks.

3.3.2 Numerical results and discussion

Experiments were conducted using gradient-based algorithms Trust Region Policy Optimization (TRPO) [147] and Proximal Policy Optimization (PPO) [146] as well as the gradient-free method Evolutionary Strategies (ES) [139] and an implementation of Augmented Random Search (ARS) [107]. For ARS a linear policy is used. For ES we use a deterministic multi-layer perceptron (MLP) with hidden layers (100, 50, 25) and `tanh` non-linearity whereas for PPO and TRPO the MLP is outputs a diagonal Gaussian response. Moreover, in the PPO algorithm, a (256, 256) MLP with tanh non-linearity is used to compute a value function baseline,

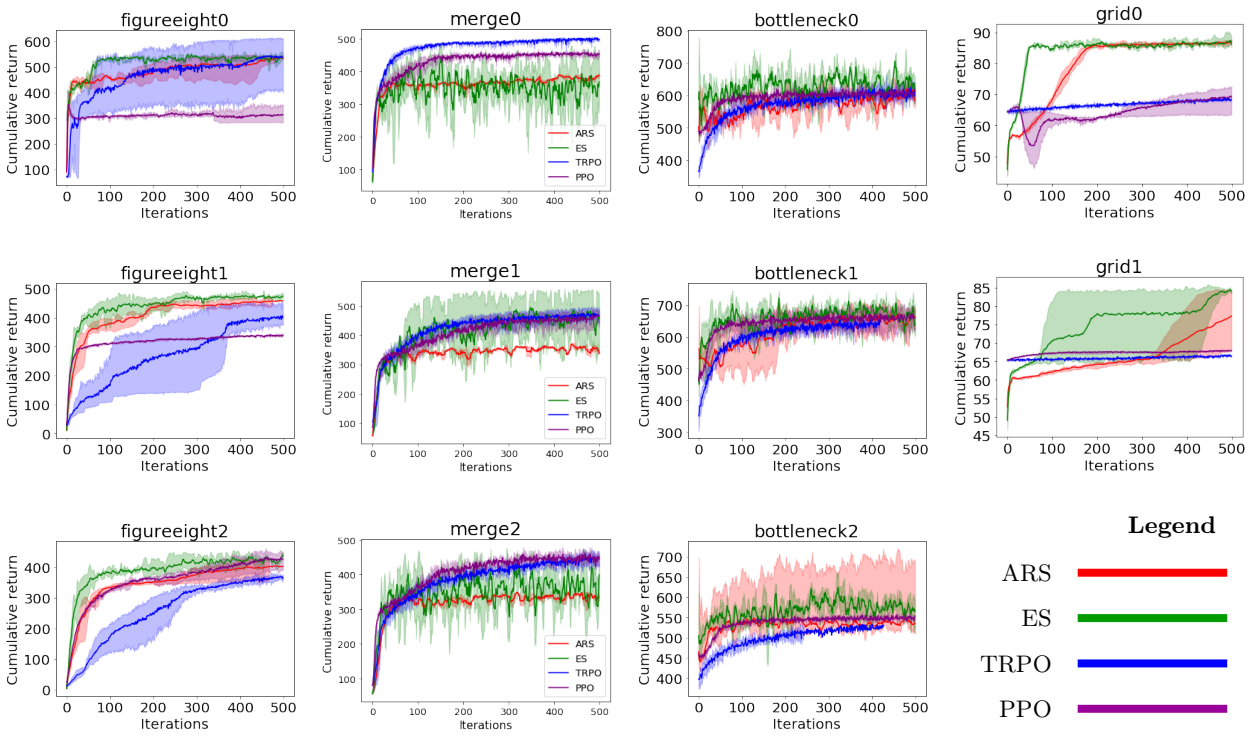


Figure 3.8: Learning curves for each benchmark. An iteration of training in each algorithm consists of 50 rollouts/trajectories. The reported values are undiscounted cumulative returns.

whereas a linear feature baseline is used for the TRPO algorithm. Further implementation details and a broader discussion on findings may be found here [184].

Figure 3.8 depicts the training performance of each algorithm for variations of the aforementioned tasks[4]. The optimal return for each learning algorithm is provided in Table 3.1 as well, alongside a human-driven baseline for comparative purposes. Given the current performance of the different RL algorithms, there is an abundance of open questions that remain for these benchmarks. For each of these problems, it remains to characterize whether the optimal solution has been achieved. Towards this end, we provide some intuition here that the optimal solution has not been achieved from the perspective of speed/delay of the resultant traffic. For the figure eight, CAVs in a fully automated setting (figureeight2) do not succeed in rearranging their spacing so as to pass through the intersection without pause or collision, and in fact *underperform* policies trained in settings with fewer controllable agents (figureeight0). The existence of a more optimal control strategy suggests that, as with the ring road in Section 3.2.2, there is an exploration problem to be solved.

[4]Increasing numbers with each task denoting increasing difficult, generally through the inclusion of larger networks or additional controllable agent, which increase sizes of both state and action spaces. Details on each configuration are available here: `https://github.com/flow-project/flow/tree/master/flow/benchmarks`

Benchmark	ARS	ES	TRPO	PPO	Human
Figure Eight 0	7.31 ± 0.54	6.87 ± 0.08	8.26 ± 0.10	–	4.18 ± 0.09
Figure Eight 1	6.43 ± 0.01	–	5.61 ± 0.55	–	4.18 ± 0.09
Figure Eight 2	5.70 ± 0.01	5.96 ± 0.00	5.03 ± 0.23	–	4.18 ± 0.09
Merge 0	11.3 ± 0.31	13.31 ± 0.54	14.95 ± 0.12	13.66 ± 0.40	7.39 ± 0.55
Merge 1	11.06 ± 0.32	17.29 ± 0.40	13.74 ± 0.23	14.61 ± 0.47	7.39 ± 0.55
Merge 2	11.5 ± 0.54	17.36 ± 0.48	14.14 ± 0.24	14.54 ± 0.31	7.39 ± 0.55
Grid 0	270.2 ± 0.2	271.7 ± 0.6	296.2 ± 2.5	296.8 ± 5.3	280.8 ± 1.5
Grid 1	274.7 ± 1.0	274.3 ± 1.1	296.0 ± 2.0	296.2 ± 2.0	276.8 ± 1.7
Bottleneck 0	1265 ± 263	1360 ± 200	1298 ± 268	1167 ± 264	1023 ± 263
Bottleneck 1	1350 ± 162	1378 ± 192	1375 ± 61	1258 ± 200	1135 ± 319
Bottleneck 2	2284 ± 231	2324 ± 264	2131 ± 190	2143 ± 208	1889 ± 252

Table 3.1: Average optimal return and standard deviation based on performance metrics after 500 training iterations; average is over 40 rollouts. "–" indicates that the experiment did not have a complete number of seeds and thus was not reported.

The brittleness of existing learning methods to problems of increasing size and complexity is further evident within the other presented benchmarks. In the bottleneck benchmark for instance, while policies are capable of arranging vehicles so as to merge smoothly through lane reductions without sharp decelerations when networks are small (bottleneck0), they fail to produce similar gains in bottlenecks with multiple the number of lanes (e.g bottleneck2, for which the optimal reward should, in theory, be double to denote similar gains). Similarly, for the grid, while we do not have a proof that we have not discovered an optimal solution for the grid problems, visualizations of the solutions do not show the platooning that is expected of optimal solutions to heavy traffic inflows. This is a heuristic argument and further characterization of the optimum is worth pursuing. However, we note that, as seen in Table 3.1, policies learned within the grid do not provide noticeable benefits to the mobility of drivers.

Finally, we note that gradient-based methods in particular fail to meaningfully update their behaviors after the first few iterations, as evidenced by the flat training curves within the large grid and bottleneck benchmarks. This suggests that there still remain fundamental advances to be made in learning policies that can tackle the large action and state spaces that arise in trying to control city-scale road networks.

3.4 Chapter Summary

This chapter introduces the topic of microscopic traffic flow modelling and presents Flow, a tool for integrating such models with state-of-the-art reinforcement learning algorithms. It then provides some insights into the efficacy of RL algorithms in solving a multitude of

traffic control problems and introduces some challenges that emerge when adapting the two together. Broadly speaking, these challenges include the following:

1. Policies struggle to find optimal solutions, particularly in partially observable problems or tasks with high degrees of freedom. This, in certain settings, may be addressed through reward engineering; however, optimal reward configurations are nontrivial to assign and often result in negative externalities, as evidenced by the formation of large gaps in the ring road task.

2. In addition, learning methods in difficult problems struggle early within the training procedure, producing little or no gains after the first few iterations of training despite performing on par with human models. This suggests that incrementally updating policies with random starting parameters is insufficient in a multitude of mixed-autonomy traffic control settings.

3. Finally, policies learned in larger scale problems often underperform those produced in analogous simpler tasks. This, while once again pointing to issues of scale, also begs the question: Can said policies in simpler problems be transferred to complex settings to support or enhance learning?

We tackle each of these points in the remaining chapters of this document.

Part I

Learning Across Long Time Horizons

CHAPTER 4

Inter-Level Cooperation in Hierarchical Reinforcement Learning

We begin in this chapter by exploring methods for enabling long term exploration in mixed autonomy settings with sparse or under-specified reward functions. In particular, we focus here on motivating such exploration through the use of hierarchies. Hierarchical reinforcement learning techniques present promising methods for enabling structured exploration in complex long-term planning problems. Such systems, however, struggle when concurrently training all levels within a hierarchy, thereby hindering the flexibility and performance of existing algorithms. In this chapter, we draw connections between these and similar challenges in multi-agent RL, and hypothesize that increased cooperation between policy levels in a hierarchy may help ameliorate hierarchical credit assignment and exploration when exposed to ever-evolving and suboptimal lower-level policies. To validate this hypothesis, we introduce a simple yet effective technique for inducing hierarchical cooperation via loss-sharing among sub-policies. Experimental results on a variety of tasks demonstrate that inducing cooperation results in stronger performing policies and increased sample efficiency on a set of difficult long-time horizon tasks. In particular, we find that the coupling of both hierarchies and cooperative learning between hierarchy layers allows automated vehicles to generate traffic regulation policies without the need for auxiliary rewards functions that redirect learning in undesirable ways. Finally, we demonstrate that policies trained using our method display better transfer to new tasks, highlighting the benefits of our method in learning task-agnostic lower-level skills.

This chapter is adapted from [83]. Videos and code are available at `https://sites.google.com/berkeley.edu/cooperative-hrl`.

4.1 Introduction

To solve interesting problems in the real world, agents must be adept at planning and reasoning over long time horizons. To achieve meaningful goals in robotics problems [130, 44], for instance, agents must first compose lengthy sequences of actions to effectively navigate

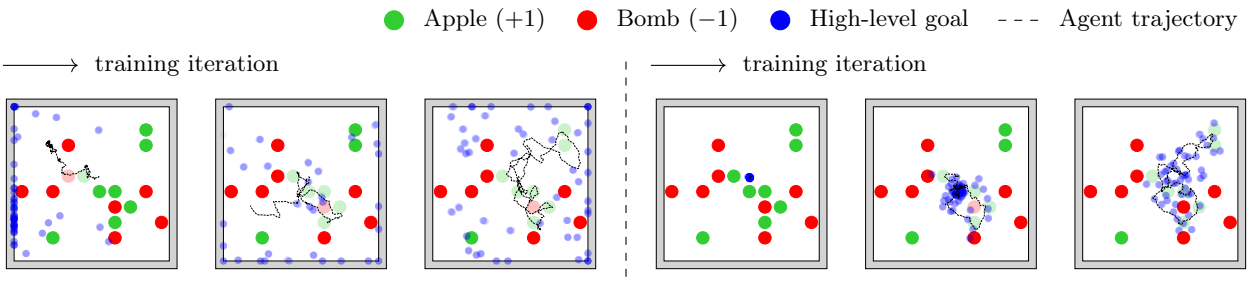
and explore their surroundings. Similarly, in many traffic control settings [201, 184], agents must choose among actions that produce limited but cascading short-term effects on the state of traffic that, if poorly chosen, may destabilize the system. These and similar settings provide limited immediate feedback on the efficacy of individual actions or behaviors, and as such have proven particularly challenging to standard reinforcement learning (RL) in the absence of advanced exploration strategies.

Hierarchical reinforcement learning (HRL) techniques provide natural methods for inducing structured exploration in these difficult, long-horizon tasks. By decomposing challenging tasks into simpler sub-problems using a hierarchy of policies, such methods refine exploration from the perspective of higher-level policies, enabling agents to search for optimal solutions across a range of temporally extended sequences [162]. This simplifies learning for such policies over large timescales, thereby allowing hierarchies, particularly with well-defined lower-level behaviors, to outperform their non-hierarchical counterparts in a variety of interesting and difficult problems [162, 86, 183, 117].

More recently, concurrent learning methods in HRL have garnered particular interest as a means of improving the flexibility and performance of past approaches [95, 117, 9]. By jointly optimizing distributions of useful subgoals by a higher-level policy and actions by the lower level to achieve them, such methods are capable of overcoming suboptimalities that arise from handcrafted or predefined lower-level policies [13] and provide increased flexibility when such sub-policies are insufficiently robust to solve novel tasks [200]. However, simultaneously learning multi-level hierarchical behaviors poses a significant challenge to existing algorithms. Critically, challenges arise from the conditional relationship between the efficacy of individual goals and a policy's ability to achieve them. This, coupled with the ever-evolving nature of lower-level policies, forces hierarchies to solve for optimal goals across a non-stationary target as lower-level policies specialize to new subtasks. This dynamic, as we show in Section 4.5.3, introduces significant biases to the credit assignment procedure when lower-level policies are unable to reach high-efficacy subgoals, further destabilizing training.

Contribution. In this chapter, we demonstrate that motivating cooperation between subagents of a hierarchy can address instabilities that arise from improper credit assignment in concurrent HRL. This approach, as we discuss later, draws insight from the importance of actively promoting positive information sharing in multi-agent, communicative systems [73, 45], and the effects of improved cooperation on the dynamics of learning in mixed cooperative-competitive tasks [168]. To induce cooperation between levels of a hierarchy, our method for *Cooperative HiErarchical Reinforcement learning*, or CHER, penalizes goal-assignment behaviors beyond a lower-level policy's degree of specialization. This penalty unifies the objectives of both policies from the perspective of the higher level. We derive a gradient for this joint objective by exploiting concepts similar to differentiable communication in multi-agent RL [51].

Figure 4.1 depicts the effect of CHER on assigned waypoint goals in an agent navigation task at different stages of training. Seen here, goals are restricted in different stages in accordance with a low-level policy's goal-reaching abilities and expand in the direction of desirable paths as the training progresses. This reduction in the variability of assigned goals



(a) Standard HRL (b) Cooperative HRL

Figure 4.1: Learned higher-level goal distributions and lower-level agent trajectories from normal HRL and our method in an environment whereby agents are incentivized to move towards (green) apples and away from (red) bombs. Our agents develop more reasonable goal proposals that allow low-level policies to learn goal reaching skills more quickly and navigate more intricately through their environments. This improves hierarchical credit assignment, thereby accelerating learning.

disambiguates under-performing from unachievable goals and producing a more informative loss landscape for higher-level policies.

We evaluate the performance of CHER on a collection of HRL environments and two previously unexplored mixed-autonomy traffic control tasks. These experiments provide two useful insights on the role of cooperation in HRL. We find that CHER accelerates and improves the asymptotic performance of learning in various tasks where more precise goals are needed to overcome certain bottlenecks. Next, we find that CHER improves the transferability of learned lower-level skills to novel tasks with differing objectives. This result highlights an additional benefit of learning across a curriculum of dynamically expanding distributions of goals, and suggests that similar cooperative methods could serve an important role in learning task-agnostic policies to improve continual learning [29] in HRL.

4.2 Related work

We explore methods at the intersection of hierarchical RL, multi-agent RL, and communication and cooperation across multiple interacting agents. We cover the most relevant topics in each of these fields to this work.

4.2.1 Multi-agent reinforcement learning

The work presented in this chapter takes inspiration from studies of communication and cooperation in multi-agent RL (MARL). In MARL, communication signals are shared among agents as a means of influencing neighboring agents to achieve a common objective. These messages take multiple forms, and may be shared directly through the observations [51, 115] or

hidden layers of neighboring actors [157], as well as indirectly via a policy's intentions [48, 78]. These messages, however, are devoid of meaning at the early stages of learning. Challenges, as such, emerge due to the ambiguity of shared signals, with agents forced to balance between sending and interpreting messages. HRL has not used similar signals and has focused on one-directional channels with higher-level policies communicating goals to lower-level policies. However, similar difficulties with communication persist, as lower-level policies fail to perform relevant goals, and higher-level policies fail to distinguish undesirable goals from those in which the lower level underperformed. The work presented here aims to provide an algorithmic framework for encouraging the balancing of exploring and interpreting viable goals in HRL, and will hopefully motivate future work on unifying solutions between MARL and HRL.

Several studies have attempted to address the challenge of stationarity and optimization in MARL. These solutions have included centralized training paradigms [105, 49, 203], regularization techniques for grounding communication in interpretable features [115], methods for intrinsically motivating desirable interactions between agents [188, 45, 73], and many others. Inline with intrinsically motivating positive interactions, our work is motivated by Tampuu et al. [168], which observe that loss-sharing among agents in two-player games encourages the formation of cooperative behaviors and reduces overestimation biases in decentralized credit assignment. In contrast, we explore the effects of loss-sharing on policies solving hierarchically coupled objectives whereby cooperation may produce inverse effects on exploration. We discuss this further in Section 4.4.3.

4.2.2 Hierarchical reinforcement learning

Several studies have attempted to address training instabilities in concurrent HRL [9, 95, 117, 119, 59]. In particular, prior studies have tackled the problem of *non-stationarity* in HRL, a challenge notably common to MARL [197, 25, 50] whereby the continually changing nature of decision-making among policy levels restricts of usability of previously acquired samples in value estimation. To tackle this problem, prior studies introduce varying techniques for subgoal relabeling to samples at training time. In [117], for instance, goals at training time are relabeled with goals that are most likely to produce the priorly stored sequence of actions. Moreover, in the work of Levy et al. [95], non-stationarities are addressed by replacing a subset of goals with states achieved in hindsight [5]. We compare our solution to concurrent HRL training against these algorithms in Section 4.5 and find that CHER further improves sample efficiency and policy quality in particularly complex tasks where more precise goal-assignment behaviors are needed.

4.3 Preliminaries

Reinforcement learning problems are generally studied as a *Markov decision problem* (MDP) [17], defined by the tuple: $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, T)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is an n-dimensional state space, $\mathcal{A} \subseteq \mathbb{R}^m$ an m-dimensional action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ a transition probability

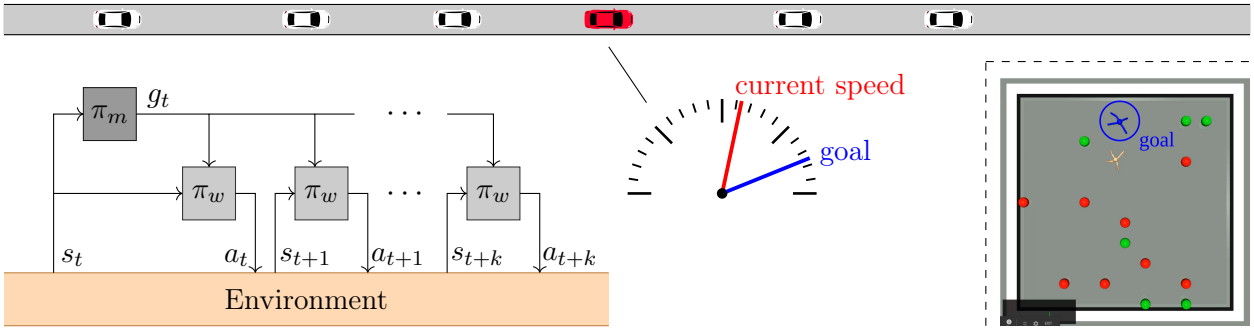


Figure 4.2: Explored hierarchical model. **Left:** A manager π_m issues goals g_t over k steps to a worker π_w. The worker then performs actions a_t to achieve them. **Right:** The goals denote desired states for the worker to traverse. For AV control tasks, we define the goal as the desired speeds for each AV. Moreover, for agent navigation tasks, the goals are defined as desired joint positions and angles.

function, $r : \mathcal{S} \to \mathbb{R}$ a reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_+$ an initial state distribution, $\gamma \in (0, 1]$ a discount factor, and T a time horizon. In a MDP, an *agent* is in a state $s_t \in \mathcal{S}$ in the environment and interacts with this environment by performing actions $a_t \in \mathcal{A}$. The agent's actions are defined by a policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parametrized by θ. The objective of the agent is to learn an optimal policy $\theta^* := \text{argmax}_\theta J(\pi_\theta)$, where $J(\pi_\theta) = \mathbb{E}_{p\sim\pi_\theta}\left[\sum_{i=0}^{T} \gamma^i r(s_i)\right]$ is the expected discounted return.

Goal-conditioned HRL In this chapter, we explore the implications of inducing inter-level cooperation on feudal, or goal-conditioned, hierarchical models in RL [38, 183, 117]. Within such a hierarchy (see Figure 4.2, left), a higher-level manager policy π_m assigns goals $g_t \sim \pi_m(s_t)$ to a *universal* [142] worker policy π_w at the lower level. The worker policy then performs primitive actions $a_t \sim \pi_w(s_t, g_t)$ to achieve these goals for a meta-interval k, at which point a new goal is assigned.

The goals assigned by a manager denote a desired state, typically either within the original state-space or a learned latent embedding of this space [118, 158]. This relationship is defined via the worker's goal-conditioned reward function $r_w(s_t, g_t, s_{t+1})$. For this chapter, we use a reward function similar to the ones presented in [130, 183, 117], in which goals characterize desired deviations in the state from the perspective of an agent. Figure 4.2 depicts examples of such goals for the problems explored within this chapter[1]. For automated vehicle (AV) control problems, we define the goal as the desired speeds for each AV. Moreover, for agent navigation tasks, these goals consist of the agent's desired position and joint angles. The reward function used to instill such a behavior is $r_w(s_t, g_t, s_{t+1}) := -||s'_t + g_t - s'_{t+1}||_2$, where s'_t is the subset of the state for which goals are assigned. The objective of the worker is to find the optimal parameters $\theta^*_w := \text{argmax}_\theta[J_w(\theta)]$ that solve for the goal-conditioned

[1]We provide a more detailed description of the utilized goal-assignment strategies in Section 4.10.2.

Algorithm 1 CHER

Initialize policy parameters $\theta_w, \theta_m,$ memory $\mathcal{D},$ and cooperative term λ
For dynamic updates of $\lambda,$ initialize δ
while True **do**
    for each $t = 0, \ldots, T$ **do**
        if $t \mod k == 0$ **then**
            $g_t \leftarrow \pi_{\theta_m}(g_t|s_t)$ ▷ Sample manager action
        else
            $g_t \leftarrow h(s_{t-1}, g_{t-1}, s_t)$ ▷ goal-transition function
        end if
        $a_t \leftarrow \pi_{\theta_w}(a_t|s_t, g_t)$ ▷ Sample worker action
        $s_{t+1}, r_t^m \leftarrow$ env.step (a_t)
        $r_t^w \leftarrow r_w(s_t, g_t, s_{t+1})$ ▷ Worker reward
    end for each
    $\theta_w \leftarrow \theta_w + \alpha \nabla_{\theta_w} J_w$ ▷ Train worker
    $\theta_m \leftarrow \theta_m + \alpha \nabla_{\theta_m}(J_m + \lambda J_w)$ ▷ Train manager
    if δ initialized **then**
        $\lambda \leftarrow \lambda + \alpha \nabla_\lambda (\lambda\delta - \lambda Q_w)$ ▷ Train λ
    end if
end while

objective [120, 104]:

$$J_w = \mathbb{E}_{s\sim p_\pi}\left[\sum_{t=0}^{k} \gamma^t r_w(s_t, g_t, s_{t+1})\right] \tag{4.1}$$

We consider a concurrent training procedure for the manager and worker policies. In this setting, as the worker solves for optimal goal-conditioned behaviors, the manager jointly searches the space of viable goals and receives a rewards $r_m(s_t)$ extrinsically defined by the underlying task. The objective for the manager policy is to learn an optimal distribution of states to goals $\theta_m^* := \text{argmax}_\theta[J_m(\theta)]$, where J_m is defined in accordance with the original RL objective as:

$$J_m = \mathbb{E}_{s\sim p_\pi}\left[\sum_{t=0}^{T} [\gamma^t r_m(s_t)]\right] \tag{4.2}$$

Notably, within this optimization procedure, value is unassigned to the goal-reaching abilities of worker policies to different goals. This factor, however, plays an important role on credit assignment in concurrent HRL, introducing biases to credit assignment when goals are performed suboptimally performed that misdirect exploration and slow down or hindering learning when such biases remain uncorrected. The absence of such direct feedback motivates the method proposed in the present chapter, which we detail in Section 4.4.

4.4 Cooperative hierarchical reinforcement learning

In this section, we introduce CHER, a method for promoting inter-level cooperation via loss-sharing in HRL. CHER promotes cooperation by propagating losses that arise from random or unachievable goal-assignment strategies. As part of this algorithm, we also present a mechanism to optimize the level of cooperation and ground the notion of cooperation in HRL to measurable variables. The final form of this method is depicted in Algorithm 1.

4.4.1 Promoting cooperation via loss-sharing

Non-cooperative or selfish objectives in otherwise cooperative multi-agent problems may destabilize the dynamics of learning within these problems. We explore this idea in the context of multi-agent systems in Section 4.2.1. We hypothesize that similar disparate objectives in goal-conditioned hierarchies form non-cooperative interactions between goal-assigning and goal-achieving policies, and that these interactions hinder a higher-level policy's ability to estimate the value of goals assigned. To validate this hypothesis, we develop a method for enforcing cooperation in goal-conditioned hierarchies by introducing loss-sharing paradigms based on those presented in [168]. In particular, we focus on the effects of loss-sharing on the goal-assignment strategies of higher-level policies, noting that the objective of a worker policy is, by definition, to cooperate with assigned goals. In line with this, we introduce a weighted form of the worker's return $r_w(s_t, g_t, s_{t+1})$ to the original higher-level task-specific reward function $r_m(s_t)$. The new cooperative reward is:

$$r'_m = r_m(s_t) + \lambda r_w(s_t, g_t, s_{t+1}) \tag{4.3}$$

where λ is a fixed parameter used to balance the effects on cooperation and exploration, and is discussed further in Section 4.4.3. The cooperative objective function for the manager policy is then:

$$J'_m = \mathbb{E}_{s\sim p_\pi}\left[\sum_{t=0}^{T} \gamma^t \left[r_m + \lambda r_w(s_t, g_t, s_{t+1})\right]\right] = J_m + \lambda J_w \tag{4.4}$$

4.4.2 Cooperative gradients via differentiable communication

In this section, we derive an expression for the gradient of the cooperative expected returns:

$$\nabla_{\theta_m} J'_m = \nabla_{\theta_m}\left(J_m + \lambda J_w\right) = \nabla_{\theta_m} J_m + \lambda \nabla_{\theta_m} J_w. \tag{4.5}$$

We derive this solution for deterministic policy gradients algorithms [150], in part for their demonstrated performance and sample-efficiency over other methods [52, 60]. We leave derivations of this objective for other RL methods for future work. Following the results of [150], we can express the first term in Eq. (4.5) as:

$$\nabla_{\theta_m} J_m = \mathbb{E}_{s\sim p_\pi}\left[\nabla_a Q_m(s,a)|_{a=\pi_m(s)} \nabla_{\theta_m} \pi_m(s)\right] \tag{4.6}$$

where Q_m is the manager's Q-function, or critic.

We would next like to compute the derivative of J_w with respect to θ_m. This objective, however, is typically conditioned solely on θ_w, and as such is non-differentiable with respect to θ_m. To compute this gradient, we adopt a trick that redefines the goal states g_t from the perspective of the worker policy as direct connections from the output of the manager policy π_m. As seen in Figure 4.3, this exposes the choice of actions by the worker policies to decisions made explicitly by the manager, thereby allowing gradients to flow between the two. The differentiable variant of assigned subgoals is defined as follows:

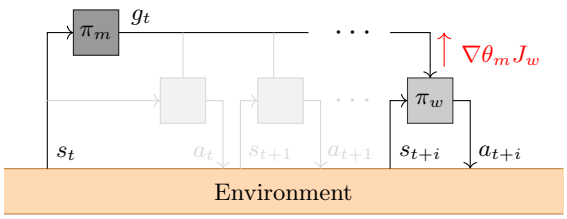


Figure 4.3: At training time, goal states are replaced with direct outputs from the manager.

Definition 1 (*Differentiable goal*)**.** The goal g_t provided to the input of the worker policy $\pi_w(s_t, g_t)$ may be defined as the direct output from the manager policy g_t whose transition function is:

$$g(s_t; \theta_m) = \begin{cases} \pi_m(s_t) & \text{if } t \bmod k = 0 \\ h(s_{t-1}, g_{t-1}(\theta_m), s_t) & \text{otherwise} \end{cases} \tag{4.7}$$

where k is the meta-interval and $h(\cdot)$ is a goal-transition function. For settings where assigned goal remains static between update intervals, this function is $h(g_t) = g_t$. In the work of [117], however, which we attempt to replicate, assigned (relative) goals are updated to maintain a unique absolute position for assigned goals via a goal-transition function $h(s_t, g_t, s_{t+1}) = s_t + g_t - s_{t+1}$.

Having defined the goal-state as a function of the manager policy's parameters θ_m, we now introduce this term to worker's objective function:

$$J_w(\theta_w, \theta_m) = \mathbb{E}_{s\sim p_\pi}\left[\sum_{t=0}^{k} \gamma^t r_w(s_t, g_t(s_t; \theta_m), s_{t+1})\right] \tag{4.8}$$

Finally, we are capable of computing the gradient of the cooperative component of the objective:

Theorem 1 (*Cooperative gradient*)**.** The solution to the deterministic policy gradient for the worker expected return with respect to goals assigned by a manager policy is:

$$\nabla_{\theta_m} J_w = \mathbb{E}_{s\sim p_\pi}\Big[\nabla_{\theta_m} g_t \nabla_g \big(r_w(s_t, g, \pi_w(s_t, g_t)) + \pi_w(s_t, g) \nabla_a Q_w(s_t, g_t, a)|_{a=\pi_w(s_t, g_t)}\big)\Big|_{g=g_t}\Big] \tag{4.9}$$

where $Q_w(s, g, a)$ is an approximation of the expected worker returns.

Proof. We provide a derivation for this gradient expanding on the work of [150] in Section 4.7.

This gradient consists of two terms. The second term computes the gradient of Q_w with respect to the assigned goals, and may be computed via standard autodifferentiation tools.

Moreover, the first term computes the gradient of the worker-specific reward. This reward is a design feature, and as such, may be explicitly computed. When combined with Eq. (4.6), this provides a framework for regulating the anticipated loss by a worker policy via the assigned goals.

4.4.3 Cooperative HRL as constrained optimization

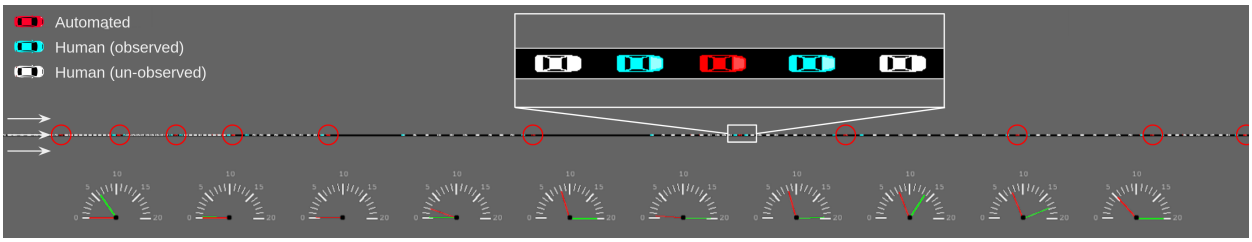
Finally, we discuss the role of cooperation and the weighting term λ on balancing exploration in HRL. This is an important point of differentiation from prior studies on the effects of differentiable communication in fully-cooperative multiagent problems [51]. In contrast, goal-assignment and goal-reaching objectives in HRL are disparate, and become adversarial when goals that produce high extrinsic rewards are difficult to achieve. Cooperation within hierarchies as such introduces competing objectives to higher-level learning as policies balance the standard incentive for exploring goals with high expected returns and the cooperative incentive for disambiguating the effects of an agent's goal-reaching abilities on credit assignment. In Eq. (4.4), this balance is regulated via λ, with higher values for λ deceasing the task objectives importance and therefore the manager's exploration.

Controlling the balance within cooperative HRL given the unknown scale of the worker's and manager's reward functions is complex. To combat this challenge we derive an objective function analogous to Eq. (4.5) that we shown in subsection 4.5.3 improved training efficiency and convergence. We observe that λ acts similar to a Lagrangian term within constrained optimization [19]. Accordingly, maximizing the objective in Eq. (4.5) equivalently solves the constrained optimization problem:

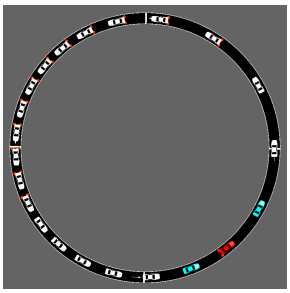
$$\max_{\pi_m} \sum_{t=0}^{T} \mathbb{E}\left[r(s_t)\right] \quad \text{s.t. } \sum_{i=t}^{T} -\mathbb{E}\left[r_w(s_i, g_i, s_{i+1})\right] \leq \delta \; \forall t \tag{4.10}$$

where δ under the original cooperative HRL formulation would be a function of λ. For fixed values of δ, we see here that loss-sharing within hierarchies restricts higher-level exploration to the set of goals for which expected lower-level returns fall within a fixed range, with smaller values for δ producing smaller sets. For waypoint finding goals similar to those presented in this chapter, this is equivalent to constraining goal assignment to within a desirable spatiotemporal distance from the agent, a distance that expands as lower-level policies learn to move to farther waypoints.

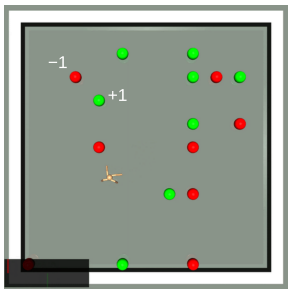
To exploit the interpretability of the δ term for hyperparameter selection, we posit that values for λ may be dynamically assigned to ensure goals adhere to a desirable δ constraint while learning cooperative goal-assignment behaviors. We derive this dual optimization procedure in Section 4.8, and demonstrate how values for δ may be assigned and their effects explored in Section 4.5.4.



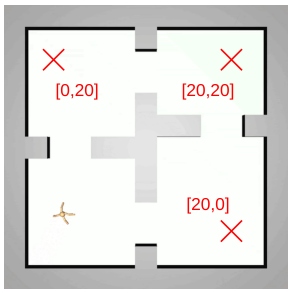
(a) Highway



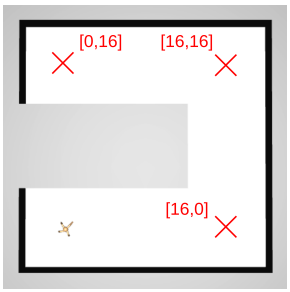
(b) Ring Road



(c) Ant Gather



(d) Ant Four Rooms



(e) Ant Maze

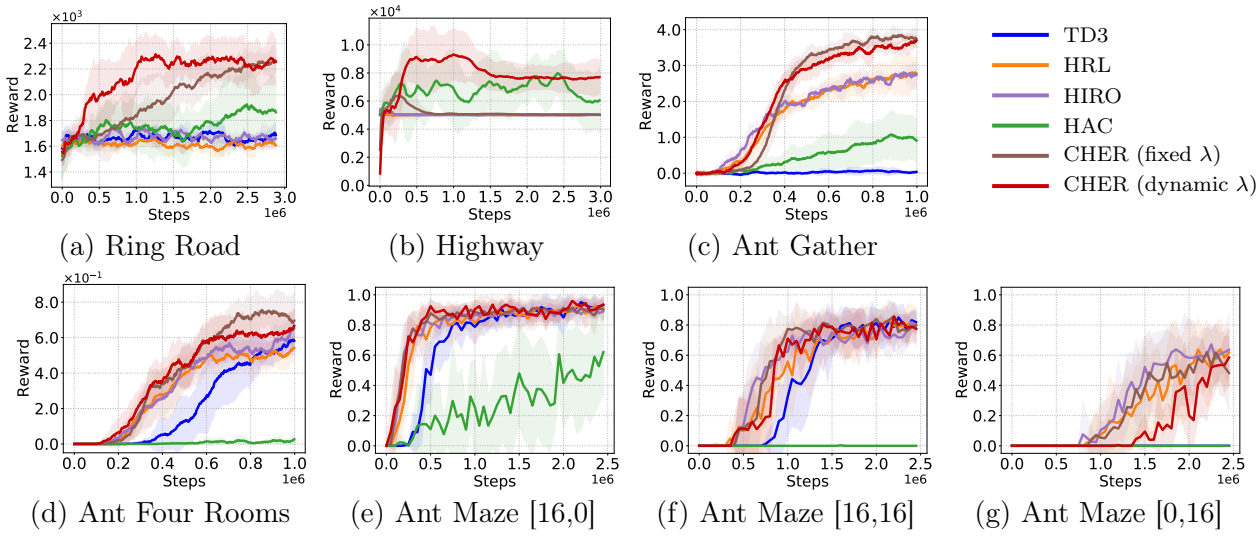
Figure 4.4: Training environments explored here. We compare the performance of various RL algorithms on two traffic control tasks (a, b) and three ant navigation tasks (c, d, e).

4.5 Experiments

In this section, we present the performance of our method over various continuous control tasks. These experiments aim to determine two things: (1) How does cooperation in HRL impact the development of goal-assignment strategies and overall learning? (2) Does the use of cooperation result in a more structured goal conditioned lower-level policy that transfers better to other tasks?

4.5.1 Environments

We study the effects of our algorithm on a variety of difficult long-term planning tasks, see Figure 4.4. These consist of three agent navigation tasks [162, 47, 117] (Figures 4.4c to 4.4e), in which a quadrupedal agent tries to achieve certain long-term goals, and two mixed-autonomy traffic tasks [202] (Figures 4.4b to 4.4a) in which a subset of vehicles act as automated vehicles and attempt to attenuate congestion in the form of stop-and-go traffic. These tasks require a high degree of temporal reasoning from agents as they navigate their surroundings over small timescales and explore viable states which provide (often delayed) positive feedback over large timescales, and as such are ideal for the exploring the effects of hierarchical learning. Further details on each environment is provided in Section 4.9.1.



(a) Ring Road (b) Highway (c) Ant Gather

(d) Ant Four Rooms (e) Ant Maze [16,0] (f) Ant Maze [16,16] (g) Ant Maze [0,16]

Figure 4.5: Training performance of the different algorithms, averaged across 10 runs.

4.5.2 Prior algorithms

We evaluate our proposed algorithm against each of the prior methods defined below. The hyperparameters used for each of these methods and CHER are provided in Section 4.10.1.

TD3 We first compare our method against a flat (non-hierarchical) fully connected network that is trained using the TD3 algorithm [52] with similar hyperparameters.

HRL Next, we compare our algorithm against a hierarchical policy following a more standard optimization scheme (see Section 4.3). This is analogous to the CHER algorithm with λ set to 0.

Finally, to provide a fair comparison of our algorithm against the state-of-the-art, we compare our method to two popular approaches to concurrent HRL. These two algorithms, named HIerarchical Reinforcement learning with Off-policy correction, or **HIRO** [117], and Hierarchical Actor-Critic, or **HAC** [95], are presented in Section 4.2.2. Implementation details for each of the HIRO and HAC algorithms are provided in Sections 4.10.3 and 4.10.4, respectively.

4.5.3 Comparative analysis

Figure 4.5 depicts the learning performance of each explored algorithm on the studied tasks. Here we find that CHER succeeds in both accelerating learning and increasing the asymptotic performance of learned policies in many scenarios. These benefits emerge, as we see in

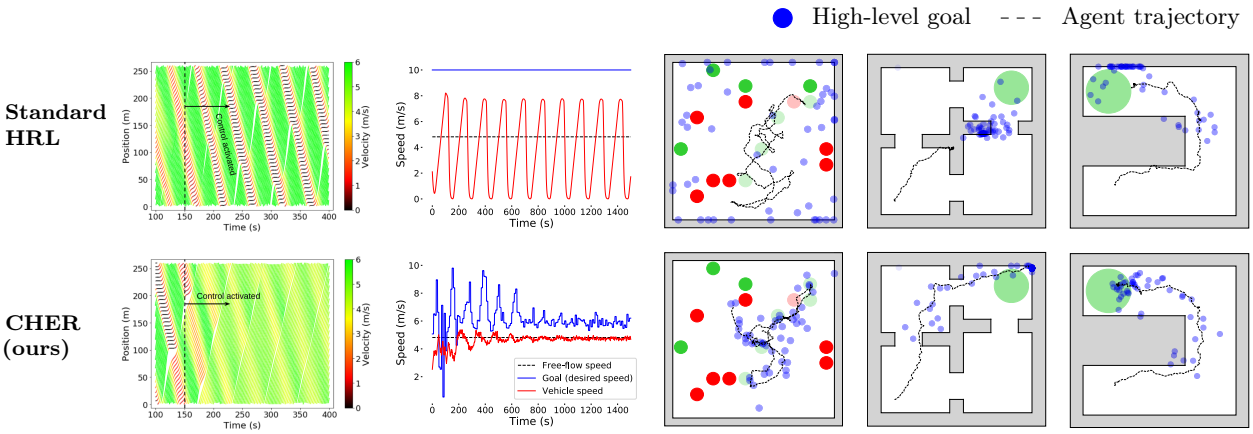
Figure 4.6, in a large part from more the presence of more informative goal-assignment behaviors provide by CHER. For instance, in the Ant Gather task (Figure 4.6b), while standard HRL methods produce highly entropic goal-assignment policies that often lie on the periphery of the map, CHER on the other hand is capable of assigning goals that more closely match an agent's desired trajectory, thereby enhancing the agent's ability to intricately navigate its surroundings. This increased level of control allows the agent to more readily avoid coming in contact with loss-incurring bomb, and produces more regulated movements by the agent which prevent the agent from losing its balances. Video demonstrations of these comparisons may be found in the supplementary material.

We see similar benefits to agents trained via CHER in the traffic control tasks. Here, improvements emerge from corrections by CHER to biases early in hierarchical credit assignment. In the Ring Road setting, for instance, managers trained using standard HRL (Figure 4.6a, top) overestimate the efficacy of assigning high desired speeds when such goals are only loosely correlated with the actions of a lower-level agent. These biases remain uncorrected for the duration of training, and result in the formation of aggressive driving behaviors, that, similar to human driving, contribute to the formation of stop-and-go traffic. In contrast, by accounting for the disparities in assigned and achieved goals, CHER succeeds in assigning desired speeds that more closely match states achieved by the agent, and ultimately, the desired equilibrium speed of the network. This behavior dampens the frequency and severity of stop-and-go traffic (see Figure 4.6a) and improves vehicle throughput. A similar, more profound variant of this behavior is learned in the Highway setting, and can be seen in Figure 4.7.

Finally, we note that within the Ant Maze task, no noticeable benefit is witnessed from CHER. This, once again, may be understood by looking to the behaviors learned by goal-assignment and goal-reaching policies using cooperative and non-cooperative objectives. Seen in Figure 4.6d, while goals assigned by CHER do more closely dictate the agent's movements, the absence of restrictive bottlenecks within this task allows the wide-varying goals assigned by standard HRL to achieve similar trajectories. This is in contrast to the Ant Gather problem, for instance, where small fluctuations in an agent's position may determine whether it comes in contact with an apple or a bomb. This suggests that regularizing around cooperative interactions strengthens learning particularly when stronger understandings of the efficacy of small changes to goals is needed.

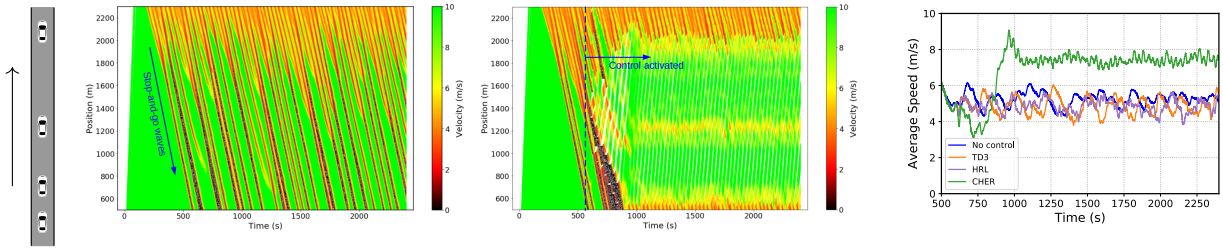
4.5.4 Cooperation tradeoff

In this section, we delve deeper into the balance that emerges between cooperation and exploration, and conceptually demonstrate how this balance may be tuned via the parameter δ (Section 4.4.3) with some prior knowledge. As mentioned in Section 4.4.3, the δ term constrains the manager policy to assign goals within a lower-level policy's degree of specialization, and in particular with respect to the expected returns a specific goal may produce. This constraint shares a relationship with aspect of the lower-level reward function and overall training procedure that can be exploited to define ranges of appropriate values for δ. We demonstrate



(a) Ring Road (b) Gather (c) Four Rooms (d) Maze

Figure 4.6: Agent and goal trajectories for some of the environments studied here. The high-level goals learned via CHER more closely match the agents' trajectories as they traverse the various environments. This results in improved dynamical behaviors by the agent in a majority of the tasks.



(a) Human baseline (b) CHER controller (c) Average speed

Figure 4.7: Performance on the Highway environment. **a)** In the absence of autonomy, downstream congestion results in the propagation stop-and-go waves, seen as the diagonal lines. **b)** The policy learned via CHER allows AVs to form gaps sufficiently large to prevent stop-and-go waves from propagating. **c)** This improves the speeds of vehicles when compared to other methods.

this here for the Ant Gather task. In this task, we notice that expected worker returns under the standard HRL approach (where no explicit cooperation is enough) converge to a value of about -600 (see Figure 4.8g). This informs us that choices of δ are unlikely to produce dynamic values of λ that will enhance cooperation or affect training. As such, we may assign a lower bound to δ of -600. This, coupled with the upper limit of 0 due to the non-positive nature of the goal-conditioned reward, provides us with bounds to explore the choice of this hyperparameter over.

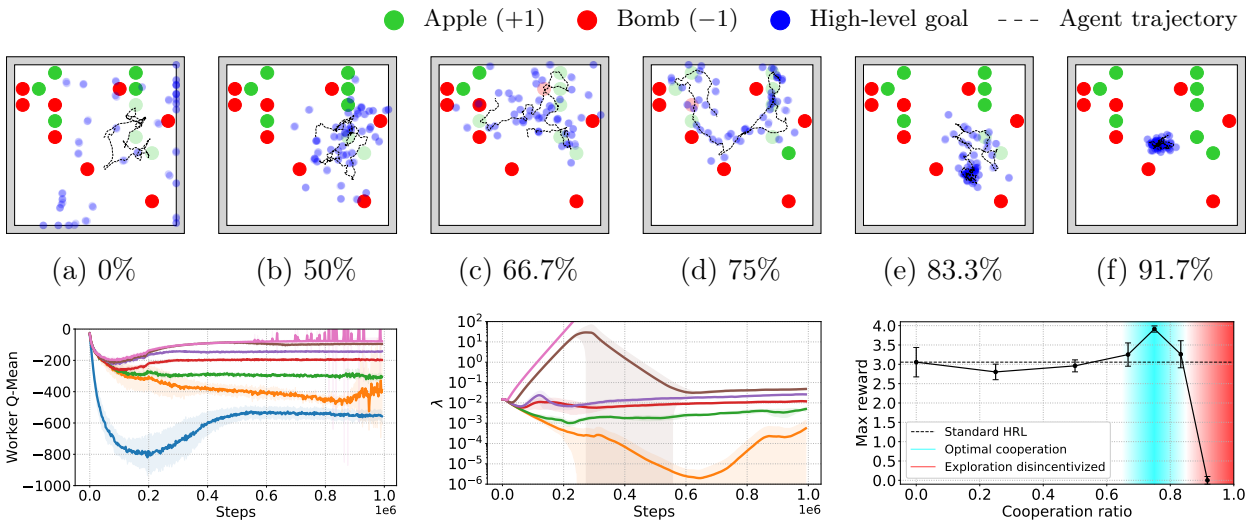
Figures 4.8i to 4.8f depict the effect of varying choices of δ on the performance of the policy. For simplicity, we define the chosen values in relation to the upper and lower bounds defined above. For example, a δ value of -450 is equated to a cooperation ratio of $(-600+450)/(-600-0) = 0.25$, or 25%. As expected, we see that as the level of cooperation increases, the goal-assignment behaviors increasingly consolidate near the path of the agent, until such a point as to disincentivize forward movement. We note however, that as seen in Figures 4.8i, the increasingly informative nature of the assigned goals and the stability associated with such cooperation begins to boost learning, producing behaviors within the vicinity of 75% that outmatch all others. This demonstrates the significance of maintaining a balance between the two. The benefits are also relatively stably maintained within this vicinity, suggesting that the use of appropriate bounds via the δ approach may help reduce sensitivities in the parameter tuning procedure.

Finally, we validate that our approached succeeds in achieving the constraints as assigned by the δ parameter. Figures 4.8g and 4.8h depict the agent's ability to achieve the assigned δ constraint and the dynamic λ terms assigned to achieve these constraints, respectively. For most choices of δ, CHER succeeds in defining dynamic λ values that match the constraint, demonstrating the efficacy of the designed optimization procedure. For very large constraints, in this case for $\delta = -50$, no choice of λ can be assigned to match the desired constraint, thereby causing the value of λ to grow exponentially. This demonstrates a sort of upper bound for the constraint, after which any goal that does not very closely match the agents trajectory is impractical. This ultimately degenerates into a behavior of assigning goals that match the current state, which we see occurs in Figure 4.8f.

4.5.5 Transferability of policies between tasks

Finally, we explore the effects of promoting inter-level cooperation on the transferability of learned lower-level policies between tasks. We here are motivated by the work of [64], which demonstrates that training goal-reaching policies via curricula of increasingly difficult goals accelerates the process of learning diverse skills. As seen in Figure 4.1, our method similarly assigns goals that gradually expand coverage while focusing on trajectories that yield high environmental rewards. As such, similar benefits to generalizability of lower-level skills may emerge here.

To study this, we look to the Ant environments in Figure 4.4 and choose to learn a policy in one environment (Ant Gather) and transfer the learned skills by the worker policy to two separate environments (Ant Maze and Ant Four Rooms). The initial policy is trained within



(a) 0% (b) 50% (c) 66.7% (d) 75% (e) 83.3% (f) 91.7%

(g) Worker expected returns (h) Evolution of dynamic λ (i) Effects of cooperation

Figure 4.8: Effect of varying degrees of cooperation for the Ant Gather environment. We plot the performance of the optimal policy and sample trajectories for various cooperation ratios (see the subcaptions for a-f). Increasing degrees of cooperation improve the agent's ability to assign desired goals that match the agent's trajectory, which subsequently improves the performance of the policy. Large degrees of cooperation, however, begin to disincentivize the agent from moving.

the Ant Gather environment for 1 million steps utilizing either the HRL, HIRO, or CHER algorithms, and weights of the worker policy are then transferred to a novel task and fixed. The manager policy is then retrained to exploit these predefined skills for the new task.

Figure 4.9 depicts the transfer setup and performance of the learned policy in reaching the three corners of each new task. We find that goal-reaching policies trained via CHER are more capable of generalizing to the distributions of goals required to solve novel tasks, attributing to improvements to the success rate of agents particularly when they move to the most difficult to reach corners. This suggests that inducing inter-level cooperation in HRL encourages the formation more robust goal-reaching abilities, a factor that provides benefits to multi-task and lifelong learning.

4.6 Chapter Summary

This chapter proposes connections between multi-agent and hierarchical RL that motivate a novel method of inducing cooperation in hierarchies. We introduce a loss-sharing paradigm that encourages cooperative interactions between goal-assignment and goal-reaching policies, and present a technique for deriving the gradient of such a loss via differentiable goal states. We find that the introduction of cooperation provides a number of underlying benefits when

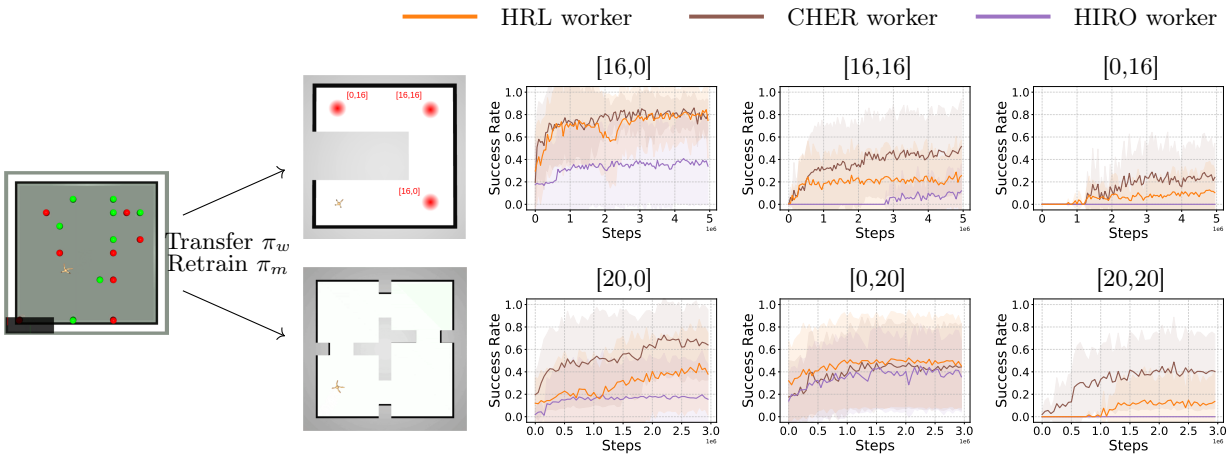


Figure 4.9: Transferability of policies learned via CHER. The policies learned when utilizing the CHER worker significantly outperform HIRO for a wide variety of evaluation points, highlighting the benefit of CHER in learning a more informative and generalizable policy representations.

concurrently optimizing goal-conditioned hierarchies. For one, we find that CHER improves the sample-efficiency and asymptotic performance of learned policies for the majority of explored problems. As we demonstrate, these benefits emerge in part from more informative and precise goal-assignment maneuvers, which produce more intricate and stable behaviors from the perspective of lower-level policies. This behavior also enhances the generalizability of learned lower-level skills when transferred to novel tasks, thereby supporting improved multi-task and lifelong learning.

As a topic of future work, we hope to exploit the effects of promoting inter-level cooperation in multi-level and hierarchies and option-critic [9] style algorithms. Moreover, in the context of promoting positive interactions between levels of a hierarchy, we are also interested in the effects of different forms of intrinsic motivation. Rewards that, for instance, promote novelty [15, 169] or empowerment [45, 73] between levels may help target goal-assignment in such a way as to further promote exploration while augmenting the effects of increased inter-level cooperation.

4.7 Derivation of cooperative manager gradients

In this section, we derive a solution for the derivative of the worker expected return J_w with respect to the manager parameters θ_m. Here, we aim to produce a solution for the deep deterministic formulation of the RL objective. Under this formulation, the estimate for the expected goal-conditioned return for a given initial condition s_t is approximated via a worker value function $V_w(s_t)$ trained alongside the actor. The expected return J_w across a distribution of initial states $\rho(\cdot)$ is then approximated as $J_w = \int_{\mathcal{S}} \rho_0(s_t) V_w(s_t, g(s_t; \theta_m)) ds_t$,

where $g(s_t;\theta_m)$ a differentiable variant of the assigned goal as defined in Eq. (4.7). For the purposes of simplicity, we express this final term as g_t from now on. The derivative of the objective is then:

$$\nabla_{\theta_m} J_w = \nabla_{\theta_m} \int_{\mathcal{S}} \rho_0(s_t) V_w(s_t, g_t) ds_t = \int_{\mathcal{S}} \rho_0(s_t) \nabla_{\theta_m} V_w(s_t, g_t) ds_t \tag{4.11}$$

We begin by computing the partial derivative of the worker value function with respect to the parameters of the manager:

$$\begin{aligned}
\nabla_{\theta_m} V_w(s_t, g_t) &= \nabla_{\theta_m} Q_w(s_t, g_t, \pi_w(s_t, g_t)) \\
&= \nabla_{\theta_m} \left(r_w(s_t, g_t, \pi_w(s_t, g_t)) + \int_{\mathcal{G}} \int_{\mathcal{S}} \gamma p_w(s', g' | s_t, g_t, \pi_w(s_t, g_t)) V_w(s', g') ds' dg' \right) \\
&= \nabla_{\theta_m} r_w(s_t, g_t, \pi_w(s_t, g_t)) + \gamma \nabla_{\theta_m} \int_{\mathcal{G}} \int_{\mathcal{S}} p_w(s', g' | s_t, g_t, \pi_w(s_t, g_t)) V_w(s', g') ds' dg'
\end{aligned} \tag{4.12}$$

where $\mathcal{G}$ and $\mathcal{S}$ are the goal and environment state spaces, respectively, and $p_w(\cdot, \cdot | \cdot, \cdot, \cdot)$ is the probability distribution of the next state from the perspective of the worker given the current state and action.

Expanding the latter term, we get:

$$p_w(s', g' | s_t, g_t, \pi_w(s_t, g_t)) = p_{w,1}(g' | s', s_t, g_t, \pi_w(s_t, g_t)) p_{w,2}(s' | s_t, g_t, \pi_w(s_t, g_t)) \tag{4.13}$$

The first element, p_{w1}, is the probability distribution of the next goal, and is deterministic with respect to the conditional variables. Specifically:

$$p_{w,1}(g' | s_t, g_t, \pi_w(s_t, g_t)) = \begin{cases} 1 & \text{if } g' = g_{t+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

The second element, $p_{w,2}$, is the state transition probability from the MDP formulation of the task, i.e.

$$p_{w,2}(s' | s_t, g_t, \pi_w(s_t, g_t)) = p(s' | s_t, \pi_w(s_t, g_t)) \tag{4.15}$$

Combining Eq. (4.13)-(4.15) into Eq. (4.12), we get:

$$\begin{aligned}
\nabla_{\theta_m} V_w(s_t, g_t) &= \nabla_{\theta_m} r_w(s_t, g_t, \pi_w(s_t, g_t)) \\
&\quad + \gamma \nabla_{\theta_m} \int_{\mathcal{G}} \int_{\mathcal{S}} \left(p_{w,1}(g' | s', s_t, g_t, \pi_w(s_t, g_t)) p_{w,2}(s' | s_t, g_t, \pi_w(s_t, g_t)) V_w(s', g') ds' dg' \right) \\
&= \nabla_{\theta_m} r_w(s_t, g_t, \pi_w(s_t, g_t)) \\
&\quad + \gamma \nabla_{\theta_m} \int_{\mathcal{G} \cap \{g_{t+1}\}} \int_{\mathcal{S}} 1 \cdot p(s' | s_t, \pi_w(s_t, g_t)) V_w(s', g') ds' dg' \\
&\quad + \gamma \nabla_{\theta_m} \int_{(\mathcal{G} \cap \{g_{t+1}\})^c} \int_{\mathcal{S}} 0 \cdot p(s' | s_t, \pi_w(s_t, g_t)) V_w(s', g') ds' dg' \\
&= \nabla_{\theta_m} r_w(s_t, g_t, \pi_w(s_t, g_t)) + \gamma \nabla_{\theta_m} \int_{\mathcal{S}} p(s' | s_t, \pi_w(s_t, g_t)) V_w(s', g_{t+1}) ds'
\end{aligned} \tag{4.16}$$

Continuing the derivation of $\nabla_{\theta_m} V_w$ from Eq. (4.16), we get,

$$
\begin{aligned}
\nabla_{\theta_m} V_w(s_t, g_t) &= \nabla_{\theta_m} r_w(s_t, g_t, \pi_w(s_t, g_t)) + \gamma \nabla_{\theta_m} \int_{\mathcal{S}} p(s'|s_t, \pi_w(s_t, g_t)) V_w(g_{t+1}, s') ds' \\
&= \nabla_{\theta_m} r_w(s_t, g_t, \pi_w(s_t, g_t)) + \gamma \int_{\mathcal{S}} \nabla_{\theta_m} p(s'|s_t, \pi_w(s_t, g_t)) V_w(g_{t+1}, s') ds' \\
&= \nabla_{\theta_m} g_t \nabla_g r_w(s_t, g, \pi_w(s_t, g_t))|_{g=g_t} \\
&\quad + \nabla_{\theta_m} g_t \nabla_g \pi_w(s_t, g)|_{g=g_t} \nabla_a r_w(s_t, g_t, a)|_{a=\pi_w(s_t, g_t)} \\
&\quad + \gamma \int_{\mathcal{S}} \left(V_w(s', g_{t+1}) \nabla_{\theta_m} g_t \nabla_g \pi_w(s_t, g)|_{g=g_t} \nabla_a p(s'|s_t, a)|_{a=\pi_w(s_t, g_t)} ds' \right) \\
&\quad + \gamma \int_{\mathcal{S}} p(s'|s_t, \pi_w(s_t, g_t)) \nabla_{\theta_m} V_w(s', g_{t+1}) ds' \\
&= \nabla_{\theta_m} g_t \nabla_g \Bigg(r_w(s_t, g, \pi_w(s_t, g_t)) \\
&\qquad\qquad + \pi_w(s_t, g) \nabla_a r_w(s_t, g_t, a)|_{a=\pi_w(s_t, g_t)} \\
&\qquad\qquad + \gamma \int_{\mathcal{S}} V_w(s', g_{t+1}) \pi_w(s_t, g) \nabla_a p(s'|s_t, a)|_{a=\pi_w(s_t, g_t)} ds' \Bigg) \Bigg|_{g=g_t} \\
&\quad + \gamma \int_{\mathcal{S}} p(s'|s_t, \pi_w(s_t, g_t)) \nabla_{\theta_m} V_w(s', g_{t+1}) ds' \\
&= \nabla_{\theta_m} g_t \nabla_g \Bigg(r_w(s_t, g, \pi_w(s_t, g_t)) \\
&\qquad\qquad + \pi_w(s_t, g) \nabla_a \left(r_w(s_t, g_t, a) + \gamma \int_{\mathcal{S}} V_w(s', g_{t+1}) p(s'|s_t, a) ds' \right) \Bigg|_{a=\pi_w(s_t, g_t)} \Bigg) \Bigg|_{g=g_t} \\
&\quad + \gamma \int_{\mathcal{S}} p(s'|s_t, \pi_w(g_t, s_t)) \nabla_{\theta_m} V_w(s', g_{t+1}) ds' \\
&= \nabla_{\theta_m} g_t \nabla_g \left(r_w(s_t, g, \pi_w(s_t, g_t)) + \pi_w(s_t, g) \nabla_a Q_w(s_t, g_t, a)|_{a=\pi_w(s_t, g_t)} \right) \Big|_{g=g_t} \\
&\quad + \gamma \int_{\mathcal{S}} p(s'|s_t, \pi_w(s_t, g_t)) \nabla_{\theta_m} V_w(s', g_{t+1}) ds'
\end{aligned}
\tag{4.17}
$$

Iterating this formula, we have,

$$
\begin{aligned}
\nabla_{\theta_m} V_w(s_t, g_t) &= \nabla_{\theta_m} g_t \nabla_g \Big(r_w(s_t, g, \pi_w(s_t, g_t)) + \pi_w(s_t, g) \nabla_a Q_w(s_t, g_t, a)|_{a=\pi_w(s_t,g_t)} \Big) \Big|_{g=g_t} \\
&\quad + \gamma \int_{\mathcal{S}} p(s_{t+1}|s_t, \pi_w(s_t, g_t)) \nabla_{\theta_m} V_w(s_{t+1}, g_{t+1}) ds_{t+1} \\
&= \nabla_{\theta_m} g_t \nabla_g \Big(r_w(s_t, g, \pi_w(s_t, g_t)) + \pi_w(s_t, g) \nabla_a Q_w(s_t, g_t, a)|_{a=\pi_w(s_t,g_t)} \Big) \Big|_{g=g_t} \\
&\quad + \gamma \int_{\mathcal{S}} p(s_{t+1}|s_t, \pi_w(s_t, g_t)) \nabla_{\theta_m} g_{t+1} \nabla_g \Big(r_w(s_{t+1}, g, \pi_w(s_{t+1}, g_{t+1})) \\
&\qquad + \pi_w(s_{t+1}, g) \nabla_a Q_w(s_{t+1}, g_{t+1}, a)|_{a=\pi_w(s_{t+1},g_{t+1})} \Big) \Big|_{g=g_{t+1}} ds_{t+1} \\
&\quad + \gamma^2 \int_{\mathcal{S}} \int_{\mathcal{S}} \Big(p(s_{t+1}|s_t, \pi_w(s_t, g_t)) p(s_{t+2}|s_{t+1}, \pi_w(g_{t+1}, s_{t+1})) \\
&\qquad \nabla_{\theta_m} V_w(s_{t+2}, g_{t+2}) ds_{t+2} ds_{t+1} \Big) \\
&\qquad \vdots \\
&= \sum_{n=0}^{\infty} \gamma^n \underbrace{\int_{\mathcal{S}} \cdots \int_{\mathcal{S}}}_{n \text{ times}} \Big(\prod_{k=0}^{n-1} p(s_{t+k+1}|s_{t+k}, \pi_w(s_{t+k}, g_{t+k})) \Big) \\
&\qquad \times \nabla_{\theta_m} g_{t+n} \nabla_g \Big(r_w(s_{t+n}, g, \pi_w(s_{t+n}, g_{t+n})) \\
&\qquad + \pi_w(s_{t+n}, g) \nabla_a Q_w(s_{t+n}, g_{t+n}, a)|_{a=\pi_w(s_{t+n},g_{t+n})} \Big) \Big) \Big|_{g=g_{t+n}} ds_{t+n} \cdots ds_{t+1}
\end{aligned}
\tag{4.18}
$$

Taking the gradient of the expected worker value function, we get,

$$
\begin{aligned}
\nabla_{\theta_m} J_w &= \int_{\mathcal{S}} \rho_0(s_0) \nabla_{\theta_m} V_w(s_0, g_0) ds_0 \\
&= \int_{\mathcal{S}} \rho_0(s_0) \sum_{n=0}^{\infty} \gamma^n \underbrace{\int_{\mathcal{S}} \cdots \int_{\mathcal{S}}}_{n \text{ times}} \Big[\Big(\prod_{k=0}^{n-1} p(s_{k+1}|s_k, \pi_w(s_k, g_k)) \Big) \nabla_{\theta_m} g_n \\
&\qquad \times \nabla_g \Big(r_w(s_n, g, \pi_w(s_n, g_n)) + \pi_w(s_n, g) \nabla_a Q_w(s_n, g_n, a)|_{a=\pi_w(s_n,g_n)} \Big) \Big] \Big|_{g=g_n} ds_n \cdots ds_0 \\
&= \sum_{n=0}^{\infty} \underbrace{\int_{\mathcal{S}} \cdots \int_{\mathcal{S}}}_{n+1 \text{ times}} \gamma^n p_{\theta_m, \theta_w, n}(\tau) \nabla_{\theta_m} g_n \nabla_g \Big(r_w(s_n, g, \pi_w(s_n, g_n)) \\
&\qquad + \pi_w(s_n, g) \nabla_a Q_w(s_n, g_n, a)|_{a=\pi_w(s_n,g_n)} \Big) \Big|_{g=g_n} ds_n \cdots ds_0 \\
&= \mathbb{E}_{\tau \sim p_{\theta_m, \theta_w}(\tau)} \Big[\nabla_{\theta_m} g_t \nabla_g \Big(r_w(s_t, g, \pi_w(s_t, g_t)) + \pi_w(s_t, g) \nabla_a Q_w(s_t, g_t, a)|_{a=\pi_w(s_t,g_t)} \Big) \Big|_{g=g_t} \Big]
\end{aligned}
\tag{4.19}
$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_n)$ is a trajectory and $p_{\theta_m,\theta_w,n}(\tau)$ is the (improper) discounted probability of witnessing a trajectory a set of policy parameters θ_m and θ_w.

4.8 Cooperative HRL as goal-constrained optimization

In this section, we derive a constrained optimization problem that motivates cooperation between a manager policy π_m and worker policy π_w. We will derive an update rule for the finite horizon reinforcement learning setting, and then approximate the derivation for stationary policies by dropping the time dependencies from the manager policy, worker policy, and the weighting term λ. Our goal is to find a hierarchy of policies π_m and π_w with maximal expected return subject to a constraint on minimum expected distance from goals proposed by π_m. Put formally,

$$\max_{\pi_m} \sum_{t=0}^{T} \mathbb{E}\left[r(s_t)\right] \quad \text{s.t.} \sum_{i=t}^{T} \mathbb{E}\left[r_w(s_i, g_i, s_{i+1})\right] \leq \delta \; \forall t \tag{4.20}$$

where δ is the desired minimum expected distance from goals proposed by π_m. The optimal worker policy π_w without the constraint need not be goal-reaching, and so we expect the constraint to be tight in practice—this seems to be true in our experiments in this chapter. The hierarchy of policies at iteration t may only affect the future, and so we can use approximate dynamic programming to solve for the optimal hierarchy at the last timestep, and proceed backwards in time. We write the optimization problem as iterated maximization,

$$\max_{\pi_{m,0},\pi_{w,0}} \mathbb{E}\left[r(s_0) + \max_{\pi_{m,1},\pi_{w,1}} \mathbb{E}\left[\cdots + \max_{\pi_{m,T},\pi_{w,T}} \mathbb{E}\left[r(s_T)\right]\right]\right] \tag{4.21}$$

subject to a constraint on the minimum expected distance from goals proposed by π_m. Starting from the last time step, we convert the primal problem into a dual problem. Subject to the original constraint on minimum expected distance from goals proposed by $\pi_{m,T}$ at the last timestep,

$$\max_{\pi_{m,T},\pi_{w,T}} \mathbb{E}\left[r(s_T)\right] = \min_{\lambda_T \geq 0} \max_{\pi_{m,T},\pi_{w,T}} \mathbb{E}\left[r(s_T)\right] + \lambda_T \delta - \lambda_T \sum_{i=T}^{T} \mathbb{E}\left[r_w(s_i, g_i, s_{i+1})\right] \tag{4.22}$$

where λ_T is a Lagrange multiplier for time step T, representing the extent of the cooperation bonus between the manager policy $\pi_{m,T}$ and the worker policy $\pi_{w,T}$ at the last time step. In the last step we applied strong duality, because the objective and constraint are linear functions of $\pi_{m,T}$ and $\pi_{w,T}$. Solving the dual problem corresponds to CHER, which trains a manager policy $\pi_{m,T}$ with a cooperative goal-reaching bonus weighted by λ_T. The optimal cooperative bonus can be found by performing minimization over a simplified objective using the optimal meta and worker policies.

$$\min_{\lambda_T \geq 0} \lambda_T \delta - \lambda_T \sum_{i=T}^{T} \mathbb{E}_{g_i \sim \pi^*_{m,T}(g_i|s_i;\lambda_T), a_i \sim \pi^*_{w,T}(a_i|s_i,g_i;\lambda_T)}\left[r_w(s_i, g_i, s_{i+1})\right] \tag{4.23}$$

By recognizing that in the finite horizon setting the expected sum of rewards is equal to the manager policy's Q function and the expected sum of distances to goals is the worker policy's Q function for deterministic policies, we can separate the dual problem into a bi-level optimization problem first over the policies.

$$\max_{\pi_{m,T},\pi_{w,T}} Q_m(s_T, g_T) - \lambda_T Q_w(s_T, g_T, a_T) \tag{4.24}$$

$$\min_{\lambda_T \geq 0} \lambda_T \delta + \lambda_T Q_w(s_T, g_T, a_T) \tag{4.25}$$

By solving the iterated maximization backwards in time, solutions for $t < i \leq T$ are a constant $c_{t:T}$ with respect to manager policy $\pi_{m,t}$, worker policy $\pi_{w,t}$ and Lagrange multiplier λ_t. Dropping the time dependencies gives us an approximate solution to the dual problem for the stationary policies used in practice, which we parameterize using neural networks.

$$\max_{\pi_m,\pi_w} Q_m(s_t, g_t) - \lambda Q_w(s_t, g_t, a_t) + c_{t:T} \tag{4.26}$$

$$\min_{\lambda \geq 0} \lambda \delta + \lambda Q_w(s_t, g_t, a_t) + c_{t:T} \tag{4.27}$$

where the constant $c_{t:T}$ may be dropped during the optimization procedure.

The final approximation we make assumes that, for a worker policy that is maximizing a mixture of the manager policy reward and the worker goal-reaching reward, the goal-reaching term tends to be optimized the most strongly of the two, leading to the following approximation.

$$\max_{\pi_m} Q_m(s_t, g_t) + \lambda \max_{\pi_w} Q_w(s_t, g_t, a_t) \tag{4.28}$$

$$\min_{\lambda \geq 0} \lambda \delta + \lambda Q_w(s_t, g_t, a_t) \tag{4.29}$$

4.9 Environment details

In this section, we highlight the environmental setup and simulation procedure used for each of the environments explored within this chapter.

4.9.1 Environments

Ant Gather In this task shown in Figure 4.4c, an agent is placed in a 20×20 space with 8 apples and 8 bombs. The agent receives a reward of $+1$ for collecting an apple (denoted by the green disks) and -1 for collecting a bomb (denoted by the red disks); all other actions yield a reward of 0. Finally, if an agent falls, defined as the z-coordinate of the agent being less than a certain threshold, the environment is terminated prematurely and the agent receives a return of 0 for the current time step.

Ant Maze For this task, immovable blocks are placed to confine the agent to a U-shaped corridor, shown in Figure 4.4e. The agent is initially placed at position $(0, 0)$, and assigned an (x, y) goal position between the range $(-4, -4) \times (20, 20)$ at the start of every episode. The agent is incentivized to reach these goals via a reward function defined as the negative L_2 distance from this (x, y) position, and is evaluated for its ability to reach the three corners of the network at the positions $(16, 0)$, $(16, 16)$, and $(0, 16)$.

Ant Four Rooms An agent is placed on one corner of the classic four rooms environment [162] and attempts to navigate to one of the other three corners. The agent is initialized at position $(0, 0)$ and assigned an (x, y) goal position corresponding to one of the other three corner ($(0, 20)$, $(20, 0)$, or $(20, 20)$). The agent, similar to the Ant Maze environment, receives reward defined as the negative L_2 distance from this (x, y) position, and evaluated over its ability to reach any of these corners.

Ring Road This environment, see Figure 4.4b, is a replication of the environment presented by [202], but with sparser rewards. A total of 22 vehicles are placed in a ring road of variable length (220-270m). In the absence of autonomy, instabilities in human-driven dynamics result in the formation of stop-and-go traffic [156]. During training, a single vehicle is replaced by a learning agent whose accelerations are dictated by an RL policy. The vehicle perceives its speed, as well as the speed and gap between it immediate leader and follower from the five most recent time steps (resulting in a observation space of size 25). In order to maximize the throughput of the network, the agent is rewarded via a reward function defined as: $r_{\text{env}} = \frac{0.1}{n_v} \left[\sum_{i=1}^{n_v} v_i\right]^2$, where n_v is the total number of vehicles in the network, and v_i is the current speed of vehicle i.

Highway This environment, see Figure 4.4a, is a higher-dimensional and open-network extension of the Ring Road environment. In this problem, downstream traffic instabilities resulting simulated via a slower-moving downstream segment produce congestion in the form of stop-and-go waves. AVs in this setting are once again tasked with improving the throughput of traffic within this network. Within this problem, the states and actions are once again chosen to to match those presented for the Ring Road environment, and similar to the work of [82] are concatenated across all automated vehicles to produce a single centralized policy.

4.9.2 Simulation details

The simulators and simulation horizons for each of the environments are as follows:

- The Ant Maze, Ant Four Rooms, and Ant Gather environments are simulated using the MuJoCo physics engine for model-based control [173]. The time horizon in each of these tasks is set to 500 steps, with dt = 0.02 and frame skip = 5.
- The Ring Road and Highway environments are simulated using the Flow [202] computational framework for mixed-autonomy traffic control. During resets, the simulation is

warmed up for 1500 steps to allow for regular and persistent stop-and-go waves to form. The horizon of these environments are set to 1500 steps. Finally, the simulation step size for the Ring Road environment is set to 0.2 seconds/step, and that of the Highway environment is set to 0.4 seconds/step.

4.10 Algorithm details

4.10.1 Choice of hyperparameters

- Network shapes of $(256, 256)$ for the actor and critic of both the manager and worker with `ReLU` nonlinearities at the hidden layers and `tanh` nonlinearity at the output layer of the actors. The output of the actors are scaled to match the desired action space.
- Adam optimizer; learning rate: $3e-4$ for actor and critic of both the manager and worker
- Soft update targets $\tau = 5e-3$ for both the manager and worker.
- Discount $\gamma = 0.99$ for both the manager and worker.
- Replay buffer size: 200,000 for both the manager and worker.
- Lower-level critic updates 1 environment step and delay actor and target critic updates every 2 steps.
- Higher-level critic updates 10 environment steps and delay actor and target critic updates every 20 steps.
- Huber loss function for the critic of both the manager and worker.
- No gradient clipping.
- Exploration (for both the manager and worker):
  - Initial purely random exploration for the first 10,000 time steps
  - Gaussian noise of the form $\mathcal{N}(0, 0.1 \times \text{action range})$ for all other steps
- Reward scale of 0.1 for Ant Maze and Ant Four Rooms, 10 for Ant Gather, and 1 for all other tasks.
- Number of candidate goals = 10. This is only used by the HIRO algorithm.
- Subgoal testing rate = 0.3. This is only used by the HAC algorithm.
- Cooperative gradient weights (λ):
  - fixed λ: We perform a hyperparameter search for λ values between $[0.005, 0.01]$, and report the best performing policy. This corresponds to $\lambda = 0.01$ for the Ant Gather, Ring Road, and Highway environments, and $\lambda = 0.005$ for the Ant Maze and Ant Four Rooms environments.
  - dynamic λ: We explore varying degrees of cooperation by setting δ such then it corresponds to 25%, 50%, or 75% of the worker expected return when using standard HRL, and report the best performing policy. This corresponds to 25% for the Ant Gather environment, and 75% for all other environments.

4.10.2 Manager goal assignment

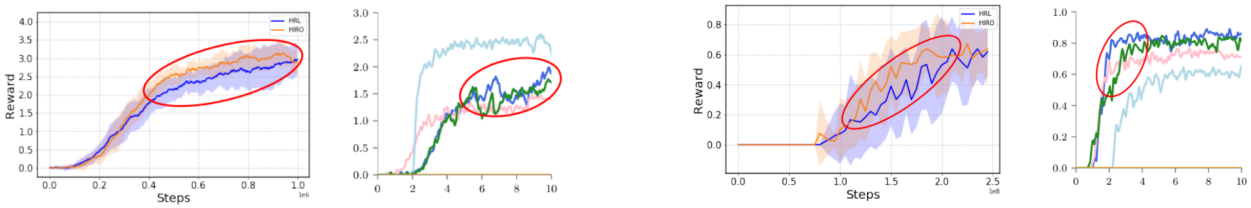
As mentioned in Section 4.10.1, the output layers of both the manager and worker policies are squashed by a `tanh` function and then scaled by the action space of the specific policy. The scaling terms for the worker policies in all environments are provided by the action space of the environment.

For the Ant Maze, Ant Four Rooms, and Ant Gather environments, we follow the scaling terms utilized by [117]. The scaling terms are accordingly ± 10 for the desired relative x,y; ± 0.5 for the desired relative z; ± 1 for the desired relative torso orientations; and the remaining limb angle ranges are available from the ant.xml file.

Conversely, for the Ring Road and Highway environments, assigned goals represent the desired speeds by controllable/automated vehicles. The range of desired speeds is set to $0 - 10$ m/s for the Ring Road environment and $0 - 20$ m/s for the Highway environment.

4.10.3 Reproducibility of the HIRO algorithm

Our implementation of HIRO, in particular the use of egocentric goals and off-policy corrections as highlighted in the original paper, are adapted largely from the original open-sourced implementation of the algorithm, as available in `https://github.com/tensorflow/models/tree/master/research/efficient-hrl`. Both our implementation and the original results from the paper exhibit similar improvement when utilizing the off-policy correction feature, as seen in Figure 4.10, thereby suggesting that the algorithm was successfully reproduced. We note, moreover, that our implementation vastly outperforms the original HIRO implementation on the Ant Gather environment. Possible reasons this may be occurring could include software versioning, choice of hyperparameters, or specific differences in the implementation outside of the underlying algorithmic modification. More analysis would need to be performed to pinpoint the cause of these discrepancies.



(a) Ant Gather (b) Ant Maze

Figure 4.10: Training performance of the original implementation of the HIRO algorithm with ours. The original performance of HIRO, denoted by the right figure within each subfigure, is adapted from the original paper, see [117]. While the final results do not match exactly, the relative evolution of the curves exhibit similar improvements, as seen within the regions highlighted by the red ovals.

4.10.4 Hierarchical Actor-Critic with egocentric goals

To ensure that the hindsight updates proposed within the Hierarchical Actor-Critic (HAC) algorithm [95] are compared against other algorithms on a level playing field, we modify the non-primary features of this algorithm to result in otherwise similar training performance. For example, while the original HAC implementation utilizes DDPG as the underlying optimization procedure, we use TD3. Moreover, while HAC relies on binary intrinsic rewards that significantly sparsify the feedback provided to the worker policy, we on the other hand use distance from the goal.

We also extend the HAC algorithm to support the use of relative position, or egocentric, goal assignments as detailed in [117]. This is done by introducing the goal-transition function $h(\cdot)$ to the hindsight goal computations when utilizing hindsight goal and action transitions. More concretely, while the original HAC implementation defines the hindsight goal $\tilde{g}_t$ at time t as $\tilde{g}_t = \tilde{g}_{t+1} = \cdots = \tilde{g}_{t+k} = s_{t+k}$, the hindsight goal under the relative position goal assignment formulation is $\tilde{g}_{t+i} = s_{t+k} - s_{t+i},\ i = 0, \ldots, k$. This function results in the final goal $\tilde{g}_{t+k}$ before a new one is issued by the manager to equal zero, thereby denoting the worker's success in achieving its allocated goal. These same goals are used when computing the worker rewards in hindsight.

Part II

Exploiting Expert Demonstrations

CHAPTER 5

Learning Energy-Efficient Driving Behaviors by Imitating Experts

In this part, we explore methods for enhancing the data-efficiency of learning procedures within mixed-autonomy settings. The challenges here arise largely from difficulties associated with credit assignment and exploration, particularly as the number of trainable agents within a task increases[1]. These challenges in mixed-autonomy settings manifest themselves early within the training procedure when agents are incapable of performing simple behaviors such as car-following. The inability of leading vehicles to perform logical primitive actions propagates upstream to other learning agents, altering both the distribution of states they witness and as such the rewards they are assigned when performing certain actions. A common behavior witnessed in these settings is the emergence of stopped traffic, for which any behavior (excluding those by the platoon leader) results in agents not moving, and as such no meaningful signal may be extracted. These challenges furthermore grow exponentially as the number of trainable agents within a network increases, thereby rendering learning in large-scale networks neigh impossible without excessive data collection or significant focus on reward engineering or some other means of problem simplification.

In this chapter, we demonstrate that imitation learning techniques can succeed in learning traffic smoothing behaviors in even large-scale networks without the need for excessive data collection. To do so, we construct an expert policy that maps non-local states of traffic to behaviors that, if adopted locally by a subset of vehicles, homogenize the flow of traffic within highway networks. We then describe some of the challenges that emerge when trying to imitate such an expert, and demonstrate that imitation learning techniques can succeed in mapping such a behavior to locally observable features of traffic in a robust and effective manner.

This chapter is adapted from [81]. Results and code are available online at `https://sites.google.com/view/il-traffic/home`.

[1] This is generally true whether trained agent share policy parameters or not, and is an active topic of research in the multiagent RL community

5.1 Introduction

Autonomous driving systems have the potential to vastly improve the quality and efficiency of existing transportation systems. With fast reaction times and socially optimal behaviors, *automated vehicles* (AVs) can improve the road capacity and traffic flow stability of existing networks [187, 166, 155, 201, 82], as well as reduce instances of vehicle collisions and similar incidents brought about by human errors [20]. The promise of such systems, however, is seldom witnessed in practice [58, 33], highlighting the challenge that lies ahead.

For the condition of addressing traffic flow instabilities in highway networks, longitudinal motion planning systems have been at the core of the conversation [209, 153]. Even in *mixed-autonomy* settings, whereby only a subset of vehicles act as AVs, such systems have been demonstrated to provide significant improvements to both network throughput and energy-efficiency. In their seminal work, for instance, the authors of [155] empirically demonstrate that even at low penetration rates, AVs driving at the equilibrium free-flow speed of a single-lane ring road can effectively stabilize traffic conditions for all vehicles involved, reducing system-level fuel consumption by 40% in the process. Such maneuvers, however, rely on prior knowledge of free-flow conditions, or access to non-local sensing and communication apparatuses from which to estimate them [148], making these maneuvers difficult to adapt to arbitrary highway networks.

In this chapter, we demonstrate that *imitation learning* (IL) techniques can help alleviate the need for non-local traffic state estimates in expertly designed AV controllers. Looking to multi-lane highways in particular, we construct a controller inspired by the work of [155] that dissipates the formation of stop-and-go traffic, but relies on downstream information to do so. Next, we apply IL to the controller and validate that such techniques, when properly deployed, can learn patterns in locally observable features that negate the need for such information, resulting in AV driving policies that operate as if they are knowledgeable of the state of downstream traffic. These findings suggest that techniques such as IL can serve an important role in adapting traffic smoothing controllers to limitations that arise in the real world.

The primary contributions of this chapter are:

- We design a controller (serving as the expert) that, through some degree of non-local sensing, dissipates the formation of stop-and-go waves within open highway networks.
- We identify key limitations in both the expert model and training procedure that limit the efficacy of IL and describe methods for addressing them. These include introducing temporal reasoning through the state representation and modifying the expert to allow it to recover from errors.
- Finally, we study the performance and robustness of the imitated policies, demonstrating that the policy succeeds at nearly perfectly matching the performance of the expert without the need for non-local state information.

The remainder of this chapter is organized as follows. Section 5.2 provides an overview of imitation learning and mixed-autonomy traffic. Section 5.3 outlines the problem setup, expert, and IL approach. Section 5.4 presents the findings and results of computational experiments over a wide variety of traffic states and compares the performance of the expert and imitated policies. Finally, Section 5.5 provides concluding remarks and potential avenues for future work.

5.2 Related Work

5.2.1 Traffic congestion and mixed-autonomy traffic

Traffic congestion is a state of traffic characterized by slower speeds, longer trip times, and increased instances of vehicular queuing. The presence of such phenomena degrades the quality and efficiency of existing networks, resulting in reduced fuel efficiency [179] and increased incidents of road rage [65] and vehicle-to-vehicle collisions [109]. As a result, solutions for such phenomena have been of primary interest within the transportation community.

The present study focuses on a form of congestion common to typical highways. Within these settings, congestion appears as *stop-and-go waves*, characterized by periodic, sudden, and at times sharp oscillations in driving speeds. Interestingly, these waves emerge even in the absence of external disturbances. For example, in simplified networks such as ring roads, small fluctuations arising from heterogeneities in human driving behaviors can produce higher amplitude oscillations by following vehicles as they break harder to avoid unsafe settings [156]. This response, termed *string instability* [164], results in the formation of stop-and-go waves as oscillations propagate deeper through a platoon.

The presence of string instabilities in human driving highlights a potential benefit for upcoming automated driving systems. In particular, if properly implemented, AVs can reduce the effects of such instabilities amongst one another [106], and by influencing the behaviors of neighboring vehicles may succeed at mitigating congestion in *mixed autonomy* settings, where only a subset of vehicles are automated. In [155], for instance, the authors demonstrate that stop-and-go waves in ring road settings can be dampened by operating as few as 5% of vehicles at the effective *equilibrium speed* of the network. The control strategy applied to the AVs, termed the *Follower Stopper*, acts as a velocity controller, navigating the speeds of automated vehicles to a predefined desired speed U (set to the equilibrium speed) while maintaining a safe gap with the vehicle ahead. Following this model, the command velocity v^{cmd} of an AV is defined as:

$$v^{\text{cmd}} = \begin{cases} 0 & \text{if } h_\alpha \leq \Delta x_1 \\ v\frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1} & \text{if } \Delta x_1 \leq h_\alpha \leq \Delta x_2 \\ v + (U - v)\frac{\Delta x - \Delta x_2}{\Delta x_3 - \Delta x_2} & \text{if } \Delta x_2 \leq h_\alpha \leq \Delta x_3 \\ U & \text{otherwise} \end{cases} \tag{5.1}$$

	Intelligent Driver Model (IDM)						
Parameter	v_0	T	a	b	δ	s_0	ϵ
Value	30	1	1.3	2.0	4	2	$\mathcal{N}(0, 0.3)$

	Follower Stopper						
Parameter	Δx_1^0	Δx_2^0	Δx_3^0	d_1	d_2	d_3	$a_{\max}$
Value	4.5	5.25	6.0	1.5	1.0	0.5	1

Table 5.1: Model parameters

where $v = \min(\max(v_l, 0), U)$ and Δx_k is defined as:

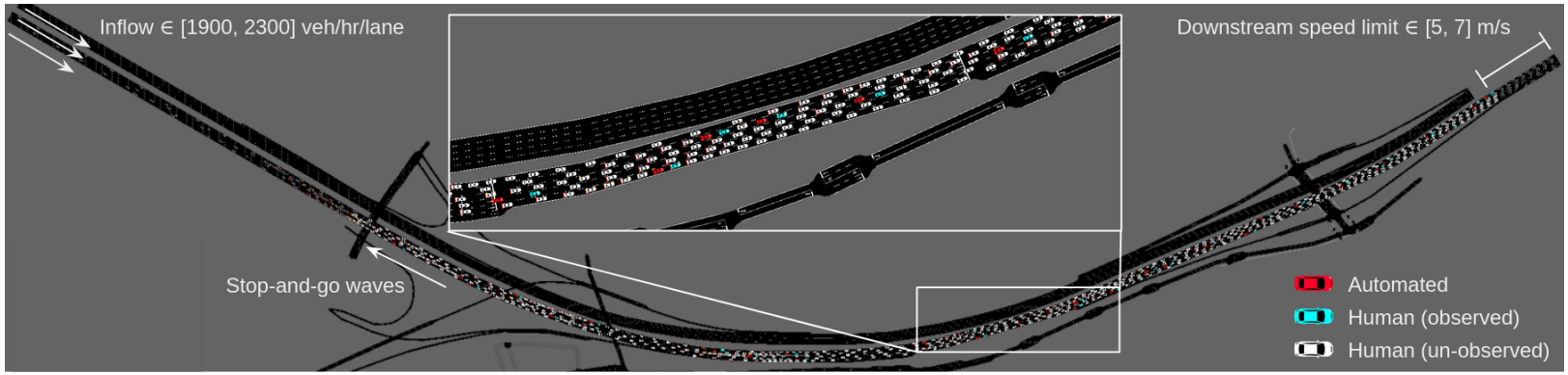
$$\Delta x_k = \Delta x_k^0 + \frac{1}{2d_k}(\Delta v_-)^2, \quad k = 1, 2, 3 \tag{5.2}$$

where $\Delta v_- = \min(\Delta v, 0)$ is the negative arm of difference between the speed of the lead vehicle and the AV. This intermediary desired speed in then converted to approach acceleration or torque commands via some form of lower-level control. In this chapter, we exploit this final controller when designing our congestion-mitigating expert. The parameters used for this model are provided in Table 5.1.

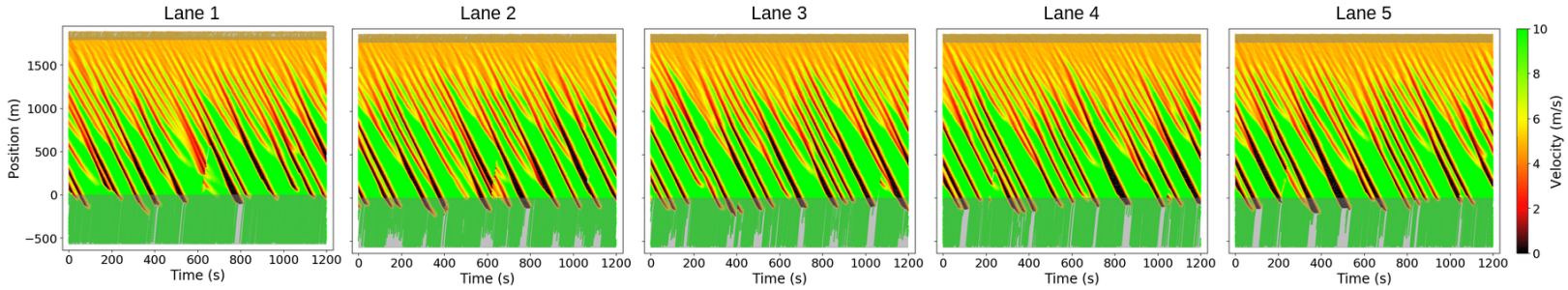
5.2.2 Imitation learning

Imitation learning (IL) refers to a class of algorithms that attempt to develop a policy $\pi_\theta(s_t) : \mathcal{S} \to \mathcal{A}$, parametrized by θ, that matches the behavior of an expert π^* through demonstrations of the performance of the expert within an environment. Let $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ denote a dataset of (state, action) demonstrations provided by an expert. In this setting, the objective of an IL algorithm is to solve the problem: $\theta := \text{argmin}_\theta \left[\mathbb{E}_\mathcal{D} \left[\mathcal{L}(\pi_\theta(s_i), a_i)\right]\right]$, where $\mathcal{L}(\cdot, \cdot)$ is a distance metric between the predicted and expert action. For this work, we use a mean-square error loss function.

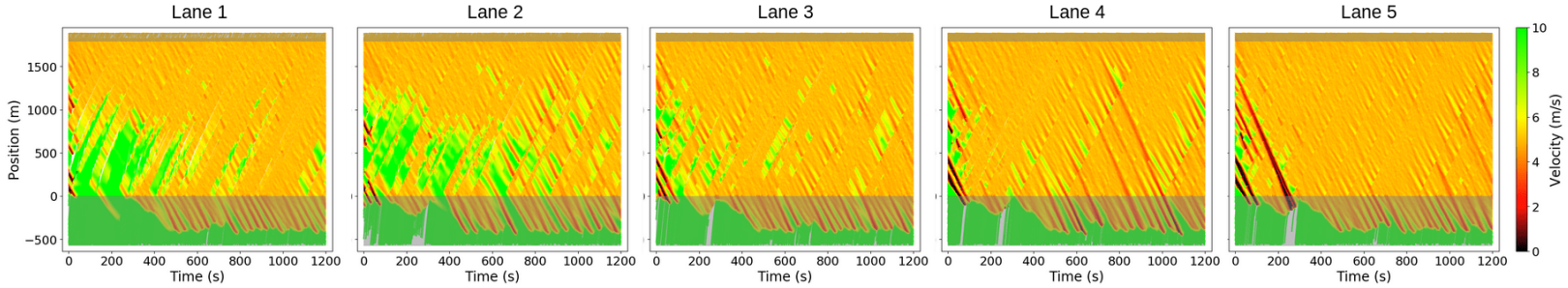
Imitation learning, and its role in autonomous driving, has grown rapidly since its initial deployment via ALVINN [132]. Nowadays, IL methods in autonomous driving have become valuable tools for model compression. In [159], for instance, these methods are used to compress computationally complex AV control strategies based on model-predictive control (MPC) into faster neural network representations. More broadly, IL has been actively being explored as a means of reducing the complexity of ruled-based approaches for autonomous driving into relatively simple end-to-end models trained to replicate the behaviors of human drivers [11, 34]. These findings and similar ongoing work highlight the potential benefits and adaptability of IL within the domain of driving. Taking inspiration from the above studies, we explore the ability of IL to compress behaviors of AV models reliant on knowledge from non-local state information into policies that are more conducive for real-world settings.



(a) Network configuration



(b) Human-driven baseline



(c) Expert controller performance

Figure 5.1: An illustration of the explored network. **Top:** We explore the application of congestion-mitigating strategies on a section of the I-210 network. **Middle:** In the absence of automation, restrictions to the outflow of vehicles from a slower-moving downstream edge result in the formation of stop-and-go waves. **Bottom:** To mitigate congestion, we design an expert inspired by the Follower Stopper. This expert, which exploits downstream information from the network, can succeed in reducing the propagation of waves.

5.3 Experimental Setup

In this section, we detail the considered problem and design an expert controller which, using non-local estimates of traffic, can dissipate the formation and propagation of congestion in the considered setting. We then present an IL paradigm through which the presented controller can be redefined to dissipate congestion in arbitrary traffic conditions using only locally perceptible state information.

5.3.1 Network configuration

We explore the problem of designing congestion-mitigating strategies for AVs in multi-lane highways. The network considered, see Figure 5.1a, is a simulation of a 1-mile section of the I-210 network in Los Angeles, California. This network has been the topic of considerable research in recent years, with various studies aiming to identify and reconstruct the source of congestion within it [55, 42]. This chapter in particular builds on the work of [92], which explores the role of mixed-autonomy traffic systems in improving the energy-efficiency of vehicles within I-210.

Within this network, congestion is generated via an imbalance between the inflow and outflow conditions. In particular, high inflow rates matching peak demand intervals are provided from the leftmost edge, and outflow rates are restricted via a reduced speed limit within the rightmost 100 m. These restrictions serve to model various downstream effects, including from slow-moving or stopping bottlenecks and downstream arterial or merges networks. The overall imbalance increases the density of the network, which coupled with string-instabilities in human-driver dynamics[2] results in the onset of stop-and-go traffic. Figure 5.1b depicts this behavior in the fully human-driven setting.

5.3.2 Dissipating stop-and-go waves via non-local sensing

To dissipate the emergence of stop-and-go waves within this network in mixed-autonomy settings, we seek to adapt to the work of [155] (see Section 5.2.1) to multi-lane highways. To do so, we must first define a distribution of desired speeds that consistently dissipate the formation of waves within the network. For the problem explored within this chapter, we take inspiration from ramp-metering systems for traffic signal control [127, 128]. Through such systems, free-flow conditions in high-demand, throughput-restricted networks are sustained by bounding the inflow of vehicles below a network's breakdown capacity. Similarly, by driving near the outflow conditions of a given network, AVs can restrict the flow of vehicles upstream, thereby preventing the buildup of dense traffic and formation of subsequent stop-and-go waves. We accordingly choose to set the desired speed U of AVs within the Follower Stopper to match the downstream speed of the network.

[2]To model string instabilities within the network, we use the *Intelligent Driver Model* (IDM) [174]. The specific model parameters are in Table 5.1.

Figure 5.1c depicts the performance of the aforementioned controller for an AV penetration rate of 5%. Seen here, the AVs succeed at dissipating the formation of stop-and-go waves, resulting in more uniform driving speeds across the network. This is true for a range of downstream boundaries conditions where stop-and-go congestion occurs and results in significant reductions to energy consumption. This solution, however, requires knowledge of downstream information, which may be unavailable. In the following subsection, we explore methods for subverting this limitation via IL.

5.3.3 Absolving the dependence on non-local states

In this chapter, we aim to determine whether control strategies similar to the one presented previously may be derived from features observable to individual AVs. Specifically, for a given AV α, we seek to design a controller $\pi_\theta(v_\alpha, h_\alpha, v_l)$ which maps actions by the expert Follower Stopper controller $a_{\text{FS}}(v_\alpha, h_\alpha, v_l, U)$ solely to the variables v_α, v_l, h_α, while abolishing the reliance on the non-local desired speed term U. To do so, we explore the use of the IL methods described in Section 5.2 while treating the Follower Stopper as the expert. This process introduces several challenges that must be addressed. We describe the two most prominent challenges and introduce techniques to mitigate them in practice.

Challenge #1: Unobservability of downstream state

The first of these challenges accounts for the absence of non-local observations for the imitated policy. The use of purely local observations confounds the expert's behavior across varying downstream conditions, thereby making specialization to unique downstream events difficult. In response to this challenge, we hypothesize that local historical data can be exploited to estimate traffic flow properties further downstream. In particular, fluctuations in driving speeds from the perspective of an AV, or the act of approaching or moving away from a lead vehicle over time, may provide useful information on the effective equilibrium speed of a given network in relation to one's own. To validate this assumption, we introduce additional temporal information to the state space of the imitated policy. Specifically, we concatenate to the original observation of an AV the past N states, collected at a sampling rate of Δt. For this problem, the following constants are chosen: $N = 5$ and $\Delta t = 2$.

Challenge #2: Recovering from errors

The second of these limitations accounts for features within the expert that are non-conducive for IL. In many driving applications (e.g. lane-keeping [21]), IL methods rely on demonstrations of how to recovery from non-optimal or erroneous trajectories to produce safe and robust behaviors. For longitudinal driving tasks explored here, errors emerge in the form of large gaps. In particular, at the early stages of training, the imitated policy often does not produce actions that maintain the speed desired by the expert, instead slowing down and allowing its leader to outpace it. While this is to be expected, the Follower Stopper does not inform

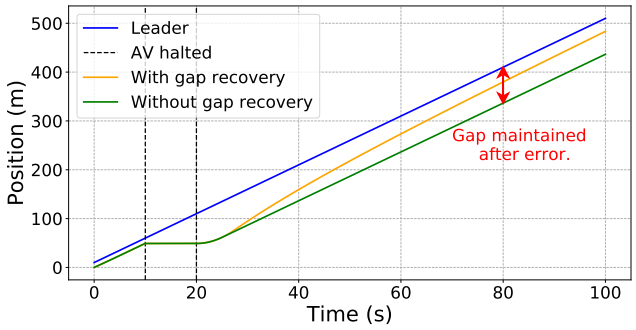


Figure 5.2: Behavior of the Follower Stopper when following a leader with and without gap recovery. The original controller, in green, cannot address gaps after an error is introduced. Conversely, the modified controller, in orange, tends towards its prior trajectory.

policies how to recover from such errors, instead opting to maintain a constant speed despite the magnitude of the gap ahead. As a result, the presence of such errors and the absence of labels that correct for them introduces an incentive to maintain large headways, which may result in additional undesirable behaviors.

In order to address this challenge, we introduce an additive term to the final condition of the expert, thereby informing the AVs to reduce the gap when large values emerge. The modified Follower Stopper command speeds is:

$$v^{\text{cmd}} = \begin{cases} 0 & \text{if } \Delta x \leq \Delta x_1 \\ v \frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1} & \text{if } \Delta x_1 \leq \Delta x \leq \Delta x_2 \\ v + (U - v) \frac{\Delta x - \Delta x_2}{\Delta x_3 - \Delta x_2} & \text{if } \Delta x_2 \leq \Delta x \leq \Delta x_3 \\ U + c(\Delta x - \Delta x_3)^2 & \text{otherwise} \end{cases} \tag{5.3}$$

where c is a constant term that toggles the rate at which recovery occurs, and is set to 0.001 in future experiments.

Figure 5.2 depicts the performance of the original and modified Follower Stopper in the presence of gap-inducing errors. The benefit of this modification becomes evident under this setting, with the modified controller clearing the gap and the original failing to do so.

5.3.4 Simulation and training procedure

Experiments are conducted in Flow [202], an open-source framework designed to enable the integration of machine learning tools with microscopic simulations of traffic. Within Flow, simulations of the I-210 are executed in SUMO [80] with step sizes of 0.4 sec/step and are warmed-started for 3600 sec to allow for the onset of congestion before being run within the training procedure an additional 600 sec. Each simulation is initialized with a random set of boundary conditions, sampled from a distribution of [1900, 2300] veh/hr/lane in increments of 50 for the inflow conditions, and [5, 7] m/s in increments of 1 for the downstream speed

limit. Once the warmup period is finished, 5% of vehicles are replaced with AVs whose actions are sampled either from the expert or policy to mimic a similar penetration rate.

For all experiments in this chapter, we use DAgger [137] for imitating the desired behavior by the expert. We initialize the expert dataset with 30k samples extracted from 20 rollouts of the expert for varying inflow rate and downstream speed limit conditions and, after each training epoch, aggregate the dataset with 7.5k new samples from states visited by the imitated policy. This is repeated for 100 training epoch for a total of 750k samples. Finally, for the choice of policy, a Multi-Layer Perceptron is used with hidden layers $(32, 32, 32)$ and a ReLU non-linearity. This policy is updated using Adam [79] with a learning rate of 0.001 and batch size of 128, and dropout [154] is employed to allow for increased robustness.

5.4 Numerical Results

In this section, we present numerical results for the expert and imitated policy presented in the previous section. Through these results, we aim to answer the following questions:

1. Does the imitated policy succeed in improving the traffic stability and energy-efficiency of vehicles within the network for different traffic conditions?

2. Is the introduction of temporal reasoning and gap recovery to the imitated and expert models, respectively, needed for proper training to occur?

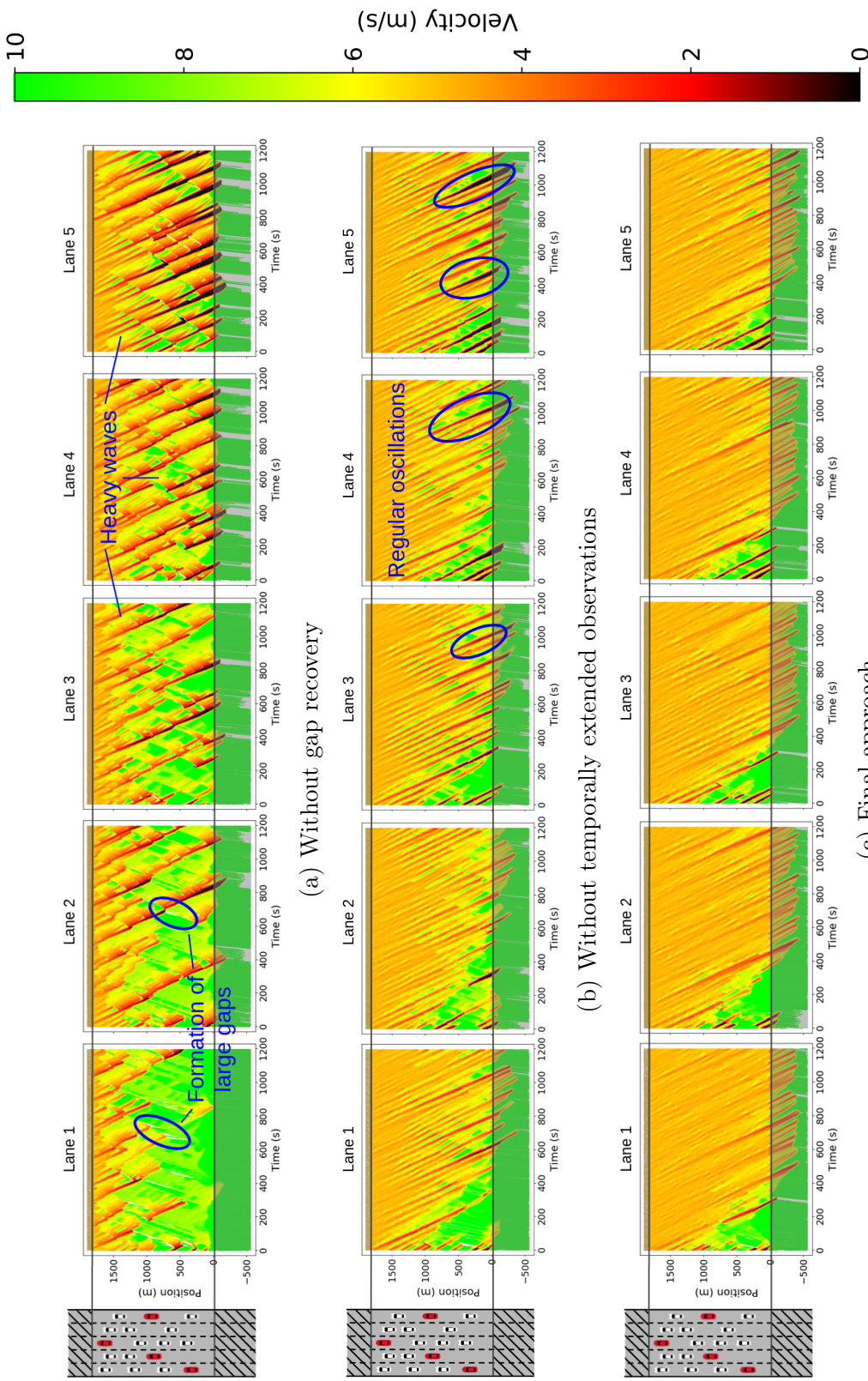
3. Is the imitated policy robust to unseen variations in network and human-driver behaviors?

Imitation learning runs for the various experimental setups are executed over 5 seeds, and the training performance is averaged and reported across all seeds to account for stochasticity between simulations and policy initialization. Further implementation details can be found at: `https://github.com/AboudyKreidieh/il-traffic`.

5.4.1 Performance comparison

Figure 5.3c depicts the spatio-temporal performance of the imitated policy for a unique traffic state. As we can see, the learned behavior largely succeeds in mitigating the frequency and intensity of stop-and-go waves that arise in the fully human-driven setting (see Figure 5.1b), resulting in more uniform driving speeds throughout the network. Moreover, this behavior closely matches that by the expert in Figure 5.1c, suggesting that the imitation procedure was successful.

The performance of the imitated policy in comparison to the expert is further elucidated in Figure 5.4. Within this figure, we compare the ability for both the expert and imitated policy to improve the energy-efficiency of the studied network based on two metrics: 1) a model of energy consumption, in miles per gallon, fitted to Toyota RAV4 emission data [92], and 2) the



(a) Without gap recovery

(b) Without temporally extended observations

(c) Final approach

Figure 5.3: Spatio-temporal comparison of the imitated model for an inflow rate of 2100 veh/hr/lane and downstream speed limit of 5 m/s. **a)** When gap-recovery is not enforced within the expert, AVs learn to form large gaps with their leaders whenever possible, resulting in the emergence of more pronounced waves in adjacent lanes. **b)** When additional temporal knowledge is not provided, the imitated policy does performs well, but fails to dissipate waves as frequently as the expert. **c)** The final approach succeeds in identifying and driving at the effective speed of the network, and in doing so dissipates the vast majority of emerging stop-and-go waves in each individual lane.

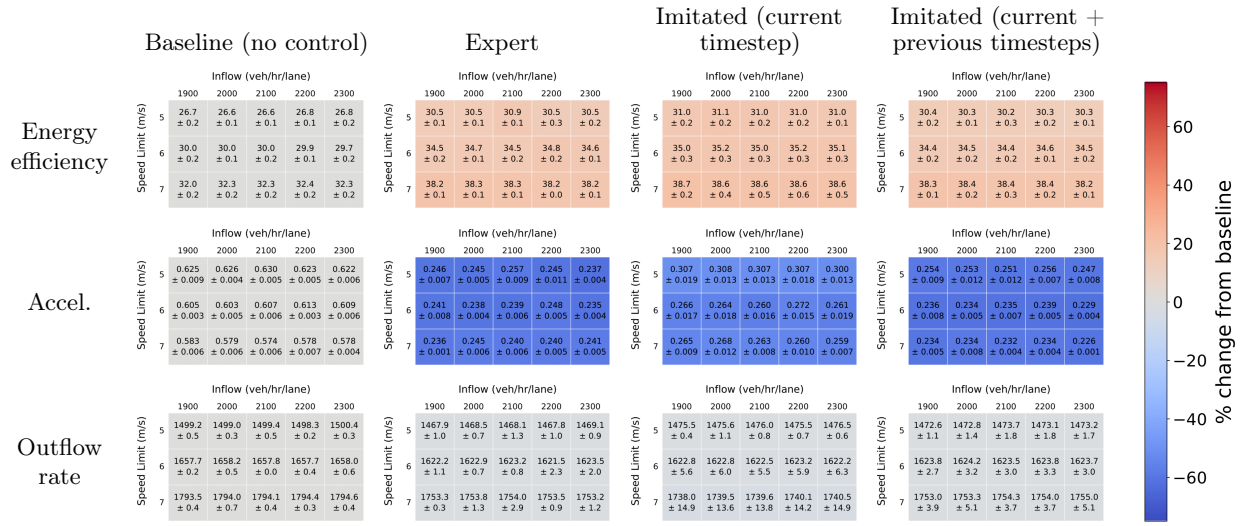


Figure 5.4: Performance of different models for different traffic conditions. Both the expert model and imitated policies provide significant improvements to both energy-efficiency and the frequency and magnitude of accelerations without significantly degrading network throughput. This is true for all sampled inflow / downstream speed limit pairs.

magnitude of accelerations by both the human and automated vehicles within the network. We also compare the throughput of the network (in veh/hr/lane) to ensure any improvements to the energy-efficiency of the network do not emerge at the cost of mobility. As we can see, the energy-efficiency of both the expert and imitated policy match one another very closely, producing values of 34.5 mpg and 34.4 mpg respectively in terms of energy consumption (a 15% improvement over the baseline value of 29.6 mpg), and values of 0.242 m/s^2 and 0.240 m/s^2 respectively in terms of fluctuations in speed (a 60% reduction over the baseline value of 0.604 m/s^2). The two also only experience very minor reductions in network throughput of around 2%, suggesting that the improvements to energy do not come at the cost of mobility. Remarkably, from the perspective of the imitated policy, this behavior emerges in the absence of non-local (downstream) knowledge of the state of traffic, demonstrating the potential for imitation learning to augment and even improve the quality of existing AV control strategies.

5.4.2 Ablation studies

Next, we explore the effect of the various augmentations presented in Section 5.3.3 on the performance of the resultant policy. Figure 5.3 depicts the learned behaviors when a number of the proposed methods are removed from the training procedure. In the absence of gap recovery during the training procedure, the imitated policy produces undesirable large gaps in several lanes as marked by the red circles. These gaps, coupled with the lane-change dynamics of human vehicles, produce distributions of highly dense traffic in adjacent lanes that enable the further propagation of congestion, thereby negating the desired behaviors of

the AVs as congestion mitigators. Alternatively, when gap recovery is enforced through the expert model but only the current time step is perceived through the input observation space, gaps do disappear and waves are further dissolved, however, the regularity at which waves emerge is greater than is experienced when additional temporal information is provided to the imitated model.

The performance gap when temporally extended state information is not provided is further highlighted in Figure 5.4. As seen in this figure, the use of current time-step information does perform well in terms of the studied energy-consumption model and network throughput when compared to the expert model. The disparity, however, is evident in terms of fluctuations in vehicle speeds, producing average acceleration/deceleration values of 0.278 m/s^2, 15% greater than the expert model and imitated model with temporally extended information. This concurs with the increased emergence of waves in Figure 5.3, and in the presence of vehicles with different energy emission properties may result in a degradation in the energy-efficiency of the network in comparison to more fine-tuned models. These findings reinforce the notion that temporal reasoning is needed to properly imitate models that utilize non-local state information.

5.4.3 Robustness to unseen phenomena

Finally, we explore the robustness of imitated behaviors to unseen variations to the task at training time. In particular, we explore two types of variations. For one, we alter the penetration rate of AVs to determine whether the changes to the dynamics associated with such values negatively degrade the performance of the policy. Next, we look to variations in the responsiveness and aggressiveness of the car-following and lane-change models, respectively. Specifically, for the car-following behaviors, we alternate the maximum acceleration, or a term, which controls the human's ability to respond to existing waves. Conversely, for the lane-change model, we modify the lane-change frequency parameter of the model, with larger values of this parameter resulting in more aggressive lane changes by the human drivers.

Figure 5.5 depicts the performance of the baseline, expert, and imitated policies for the aforementioned variations in simulation parameters. For the variations in AV penetration rate, we explore the influence of varying downstream speed limits as well but set the inflow rate to a fixed value of 2100 veh/hr/lane for each run. Moreover, for the variations in human-driver model parameters, we restrict the analysis to a single pair of boundary conditions of 2100 veh/hr/lane and 5 m/s. As we can see from this figure, the imitated policies generalize well to different penetration rates, reducing slightly for low penetration rates of 2.5%, but outperforming the expert at higher penetration rates. For modifications to the human-driver models, a performance gap does begin to emerge between the expert and imitated policy for large changes to the parameters; however, the imitated policy does continue to outperform the human-driven baseline for all regions in which the expert outperforms as well. The gap is also reduced once temporally extended state information is introduced, further highlighting the benefit of this approach.

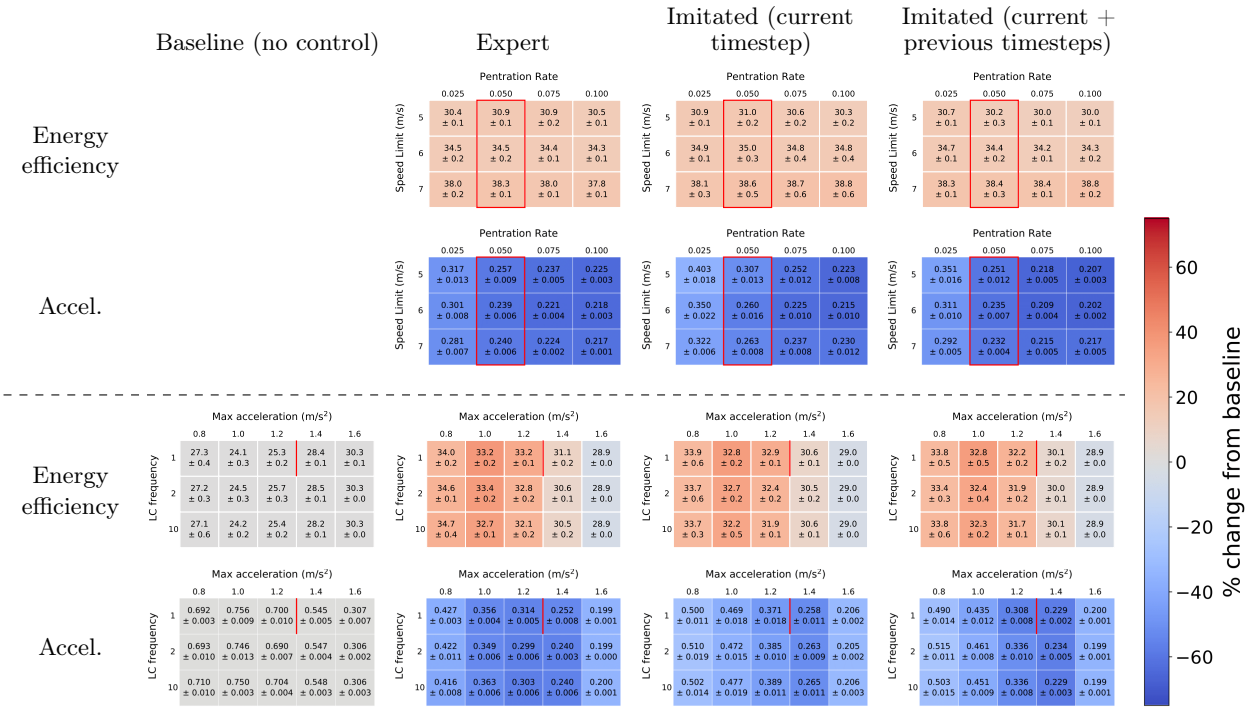


Figure 5.5: Generalizability of the learned policy to different AV penetration rates (top) and human-driver conditions (bottom). The original training regime is bordered in red. We find that the imitated policy generalizes relatively well, continuing to outperform the baseline in all settings. The policy imitated with additional temporal information also succeeds in generally better outside its training scheme.

5.5 Chapter Summary

This chapter explores the problem of designing congestion-mitigating control strategies for automated vehicles in mixed-autonomy highways. For one, it demonstrates that such strategies can be achieved by aligning the speeds of a number of vehicles with the effective speed of downstream traffic, thereby preventing the onset of dense traffic. Next, through imitation learning it demonstrates that such strategies can be mapped to observations that are locally perceivable to individual vehicles, producing a purely decentralized controller that does not require feedback from non-local sources. This final behavior is shown to improve the energy-efficiency of the explored network for values as high as 15% while effectively eliminating the presence of most waves.

As a topic of ongoing work, we aim to determine whether similar control design strategies can be applied to real-world driving settings. In particular, we aim to validate the performance of this control strategy on trajectories of human driving provided by the real world. Dependent on the performance of the model to such a domain shift, we will next determine whether a

similar procedure of iteratively readjusting the model to learn patterns from human driving in the real world can result in a more robust and transferable policy.

CHAPTER 6

Data-efficient Learning for Cooperative Driving through Expert Relabeling

In this chapter, we extend the work presented in Chapter 5. In particular, we expand the expert control strategy employed previously so as to identify desirable speed profiles when the source of the bottleneck is unseen or not undecipherable. We then demonstrate that the previously defined imitation learning procedure continues to generate meaningful results under this approach, and in particular does so even in the presence of highly-stochastic trajectories adopted from real-world data.

6.1 Introduction

Imitation learning techniques have broadly come to serve a critical role in enabling technical advancements in vehicle autonomy. Amid ever-growing complexities and uncertainties associated with automating driving in large urban settings, imitation learning strategies extended the proverbial helping hand, restructuring said challenges to one more tractable within the machine learning community. Instead of hand-crafted systems designed to decipher every possible traffic condition, behavior-planning algorithms were replaced with end-to-end models trained to replicate human behaviors extracted from large driving datasets. This is an approach that dates back to the late 1980s with the advent of the autonomous vehicle ALVINN [132] and has seen a sharp growth in popularity since, with advancements in deep learning motivating researchers to extend such approaches to more complex and interesting problems [21, 11, 62].

Imitation, while a powerful tool for developing robust feedback control strategies from demonstrations, is restricted largely by the nature of demonstrations assigned at training. This is a not uncommon point of contention, and is discussed frequently in the context of data diversity, with researchers primarily focusing on visual factors such as scenery, network structure, and the time-of-day or weather conditions within a given datapoint [43]. A less frequently targeted criticism, however, is the behavioral component of said demonstrations and under what conditions human-labeled responses hinder the performance of learned behaviors.

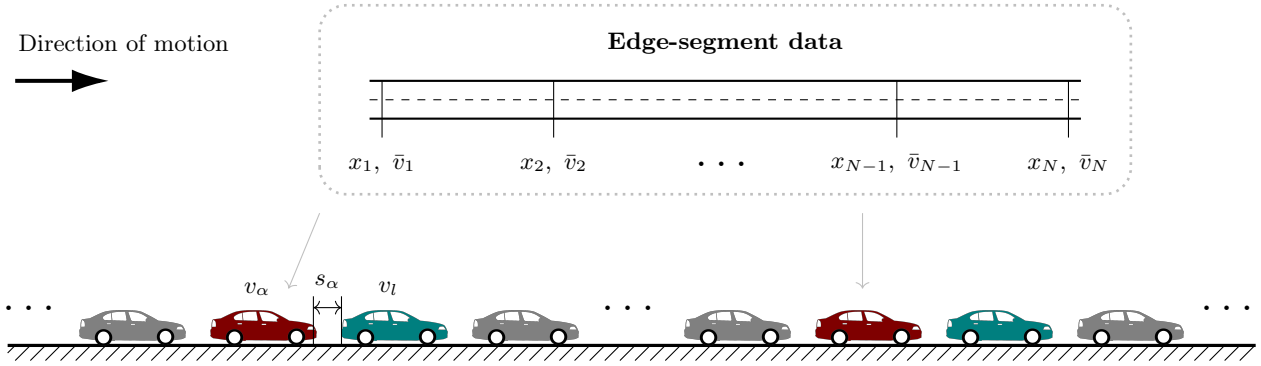


Figure 6.1: We consider the task of traffic flow harmonization in settings where CAVs (in red) are distributed amongst, and interact with, human drivers. Within this setting, we aim to define traffic regulation strategies conditioned on either the local state of traffic (i.e. the ego vehicles' states and that of their immediate vehicles in blue) or the combination of local sensory inputs and edge-segment speeds provided at various locations within the network.

In dense traffic settings, for instance, humans are frequently suboptimal actors, contributing to, as opposed to mitigating, the evolution of congestion. The reasoning behind this varies from network to network, be it capacity drops near bottlenecks [138], instabilities induced by on- and off-ramps [113, 175], or other human-driven phenomena, but propagates through the training procedure, thereby hindering the cooperative aspect of learned responses.

On the other hand, other learning approaches have been explored to induce cooperative driving within automated systems in mixed traffic settings. Reinforcement learning, in particular, has played an important role in shedding light on the style of emergent responses possible by automated vehicles, particular in fully observable or centralized control settings [201, 202]. Learning in large-scale multi-agent networks is nevertheless very difficult. The challenges here arise largely from difficulties associated with credit assignment and exploration, particularly as the number of trainable agents within a task increases. These challenges in mixed-autonomy settings manifest themselves early within the training procedure when agents are incapable of performing simple behaviors such as car-following. The inability of leading vehicles to perform logical primitive actions propagates upstream to other learning agents, altering both the distribution of states they witness and as such the rewards they are assigned when performing certain actions. A common example is the emergence of stopped traffic, for which any behavior (excluding those by the platoon leader) results in agents not moving, and as such no meaningful signal may be extracted. These challenges furthermore grow exponentially as the number of trainable agents within a network increases, thereby rendering learning in large-scale networks neigh impossible without excessive data collection, a significant focus on reward engineering, or some other means of problem simplification [211].

Contribution. In this chapter, we demonstrate that imitation learning techniques, when coupled with appropriate experts, can succeed in learning traffic smoothing behaviors in

large-scale networks without the need for excessive data collection or reward engineering. The key contributions are accordingly as follows:

1. **Expert relabeling in congestion-prone settings.** First, we construct an expert policy that maps non-local states of traffic to behaviors that, if adopted in mixed-autonomy settings, homogenize the flow of traffic within highway networks. Our controller, as we discuss, assigns appropriate speed profiles to vehicles from edge-segment data (Figure 6.1), while using local data to maintain reasonable spacings with leading vehicles. The actions assigned by such an expert can be extracted in hindsight and used as labels in a learning procedure, but are difficult to compute in real time without extensive communication and sensing infrastructures, leading to our second contribution.

2. **Imitation enabling flexibility and decentralization.** Next, we demonstrate that imitation learning techniques can succeed in mapping the above expert behavior to locally observable features of traffic in a robust and effective manner. This result demonstrates that efficient multi-agent learning methods may be generated by combining modern learning approaches with expert responses dictated by traffic flow theory. The mapping of such behaviors to differentiable models further introduces a degree of flexibility to learned responses, thereby enabling policies to serve as warm starts to other learning processes.

The remainder of this chapter is organized as follows. Section 6.2 and 6.3 provide an overview of relevant concepts and the problem setup, respectively. Section 6.4 describes the proposed expert model and provides some insights behind the modeling decisions. Section 6.5 describes the imitation learning procedure and the target networks used to validate the efficacy of our approach. Section 6.6 provides numerical results on the aforementioned tasks. And finally, Section 6.7 outlines some of the key findings and discusses potential topics of future work and challenges that may arise when attempting to adopt similar approaches within cyber-physical systems.

6.2 Preliminaries

6.2.1 Imitation learning

Imitation learning algorithms provide techniques for generating mappings between states and actions from demonstrations of desirable behaviors. Consider a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, and θ are trainable parameters, and let $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$ denote a dataset of (state, action) demonstrations provided by an expert policy π^*. In this setting, the objective of an imitation learning algorithm is to solve the problem: $\theta := \text{argmin}_\theta \left[\mathbb{E}_\mathcal{D} \left[\mathcal{L}(\pi_\theta(s_i), a_i)\right]\right]$, where $\mathcal{L}(\cdot, \cdot)$ is a distance metric between the predicted and expert action[1].

[1]For this work, we use a mean-square error loss, a function commonly used within regression problems

6.2.2 Microscopic traffic flow models

This chapter studies traffic control in the context of microscopic traffic flow models, which attempt to simulate realistic driving behaviors at the level of individual vehicles. Within these models, the acceleration of each individual vehicle of index α distributed longitudinally within a platoon is described by ordinary differential equations of the form:

$$\begin{cases} \frac{\mathrm{d}s_\alpha}{dt} = v_l(t) - v_\alpha(t) \\ \frac{\mathrm{d}v_\alpha}{dt} = f(s_\alpha(t-\tau), v_\alpha(t-\sigma), v_l(t-\kappa)) \end{cases} \tag{6.1}$$

where f is an acceleration function, v_α is the speed of the ego vehicle, v_l is the speed of the vehicle preceding it, s_α is the space headway between the two, and τ, σ, and κ are time delays.

Optimal conditions. In a longitudinal vehicles platoon, the optimal performance of human drivers homogeneously following such a dynamical structure is defined by its *uniform-flow* equilibrium state. At this state, all vehicles move at a constant speed v^* and with constant spacing s^*, such that:

$$f(s^*, v^*, v^*) = 0 \tag{6.2}$$

This equilibrium is characterized by fast driving speeds and reduced instances of sharp accelerations, and as such is considered desirable from the perspectives of mobility and energy-efficiency. In generic highway settings, however, this equilibrium is typically unstable. Instead, under dense enough conditions, small fluctuations in driving speeds propagate and grow in magnitude as following vehicles break harder to avoid unsafe settings. This process then repeats itself until, for large enough platoons, a *stop-and-go* wave is formed, i.e a scenario in which drivers regularly and sharply transition between what appear to be free-flow conditions to a state of highly dense and slow-moving traffic. The phenomenon leading to these every-growing oscillations, known an *string instability* [164], arises largely from human factors in driving, and as such may be addressed by automated systems in mixed-autonomy settings.

Intelligent Driver Model. In this chapter, the behavior of human-driven vehicles within the presented simulations are defined by the Intelligent Diver Model [174], or IDM, a common model for reconstructing realistic driving behaviors in traffic microsimulations. Following this model, the acceleration response of a vehicle α is defined as:

$$f(s_\alpha, v_\alpha, v_l) = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^\star(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right] + \epsilon \tag{6.3}$$

where $\Delta v_\alpha = v_l - v_\alpha$, ϵ is an exogenous noise term designed to mimic stochasticity in human driving behaviors, and $s^\star$ is the desired headway of the vehicle denoted by:

$$s^\star(v_\alpha, \Delta v_\alpha) = s_0 + \max\left(0, v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}\right) \tag{6.4}$$

and s_0, v_0, T, δ, a, b are fixed parameters defined further on within this document. The parameter T in particular, defines the desired time headways by drivers, and as discussed in [174], may be altered spatially or temporally to induce disturbances to vehicles states reminiscent of freeway bottlenecks. In particular, higher desired time headways restrict the number of vehicles that can reside in a freeway segment, thereby simulating the effects capacity drops once imposed on certain regions. We return to this concept in Section 6.5.

6.3 Problem statement

In this chapter, we focus on deriving longitudinal (acceleration) responses for *automated vehicles* (AVs) dispersed within a straight, single lane track. Figure 6.1 provides a visual interpretation of the explored problem setup. Within this setup, we define a mixed-autonomy platoon as a sequence of K subplatoons, each consisting of one leading AV followed by n human-driven vehicles. The platoon as a whole in turn follows a single leading vehicle, which is used to impose boundary condition on the platoon and, as discussed in Section 6.5, may also be used to generate disturbances that shift the platoon from its uniform state. The objective of AVs within this platoon is to disrupt the growth and propagation of disturbances resulting from string unstable interactions between adjacent vehicles, and in doing so shift the motion of vehicles towards their desirable uniform-flow equilibrium state.

To solve this problem, we explore two separate sensing paradigms in which AVs have access to either local or global and local knowledge of traffic state information. We define each of these paradigms as follows:

- **Local sensing.** In local sensing settings, vehicles have access to knowledge generally accessible from onboard sensing devices such as radar. This knowledge consists of the speed v_α of the ego vehicle α, the speed of its immediate leader v_l, and the space headway s_α or time headway h_α between the two. Such a sensing paradigm may safely and readily be adopted to any vehicle capable of assigning adaptive cruise control (ACC) commands, but is limited in its ability to gauge downstream events in traffic such as the propagation of stop-and-go traffic [58].

- **Global sensing.** In global sensing settings, vehicles are equipped with local sensing information and have access to traffic state estimation data defining the aggregate movement of vehicles across large spatiotemporal segments. These variables consist of average (aggregated) speeds $\bar{v}_i$ across multiple positions x_i; $i = 1, \ldots, N$ relative to the origin of the network, and may be in practice be accessed from fixed infrastructures such as loop detectors [28] and cameras [12], as well as Lagrangian or floating point sensors such as probe vehicles [148]. These estimates can be very useful for detecting and responding to downstream events, but may be inaccessible in real-time or unavailable within arbitrary road networks.

6.4 Expert control mechanism

We present an expert feedback control strategy that requires both global and local sensing to harmonize driving speeds amongst vehicles while maintaining safe and appropriate gaps between AVs and their leaders. Previous empirical studies [7, 36, 155, 81] provide a useful insight that traffic may be homogenized near its desirable *uniform* driving speed by operating a subset of vehicles near accurate predictions of said speed. However, under the every-evolving dynamics of a particular network, the actuator must reactively identify desirable speeds that match current spatio-temporal trends while not inhibiting the safety or mobility of the vehicle. To this, the two sensing paradigms may offer a helping hand. Macroscopic traffic state estimates obtained from global sensing may elucidate spatio-temporal patterns that can be exploited by AVs in a largely decentralized manner. This is in part demonstrated in the work of [7], for instance, which devises an optimal speed profile for vehicles provided speed measurements forwards in space and time. Solutions such as these, however, are often studied in the context of fully-observable macroscopic environment, and as such become brittle and unsafe in the presence of inaccurate traffic state estimates and microscopic fluctuations in speed and spacing. As a result, we take microscopic observations via local sensing into consideration to regulate the gap between AV and preceding vehicle and produce a reasonable car-following response. Finally, we also equip the expert with a safety filter to ensure enough gap will be maintained with the preceding vehicle.

With all of the above considerations mentioned, the expert consists a velocity design which including speed synchronization and gap regulation, and a safety filter as E1. (6.5) shows:

$$v_c = \max(0, \min(v_{\text{des}} + k_p(h_\alpha - h_{\text{des}}) + k_d(v_l - v_\alpha), v_{\text{fs}})) \tag{6.5}$$

We describe each subcomponent in further detail below.

6.4.1 Speed synchronization

The desirable uniform driving speed must be achieved without shared communication between adjacent vehicles, as in mixed-autonomy settings human-driven vehicles are incapable of sharing their desired speeds. Instead, we rely on traffic state data to synchronize the driving speeds of automated vehicles. In particular, vehicles are assigned speed profiles contingent on traffic state information, which is shared and common among all AVs. This speed profile may be extracted in a number of different ways, with many convolutional mapping capable of homogenizing the flow at traffic. For this purposes of this study, we consider a uniform kernel, the simplest such mapping. The speed profile at a position x_α is accordingly defined as:

$$v_{\text{avg}}(x_0) = \frac{\int_{x=x_\alpha}^{x_\alpha+w} v(x)dx}{w} \tag{6.6}$$

where $v(\cdot)$ is an interpolation across average (aggregated) speeds within each spatiotemporal segment (Section 6.3), x_α is the position of the subject vehicle, and w is the width of the

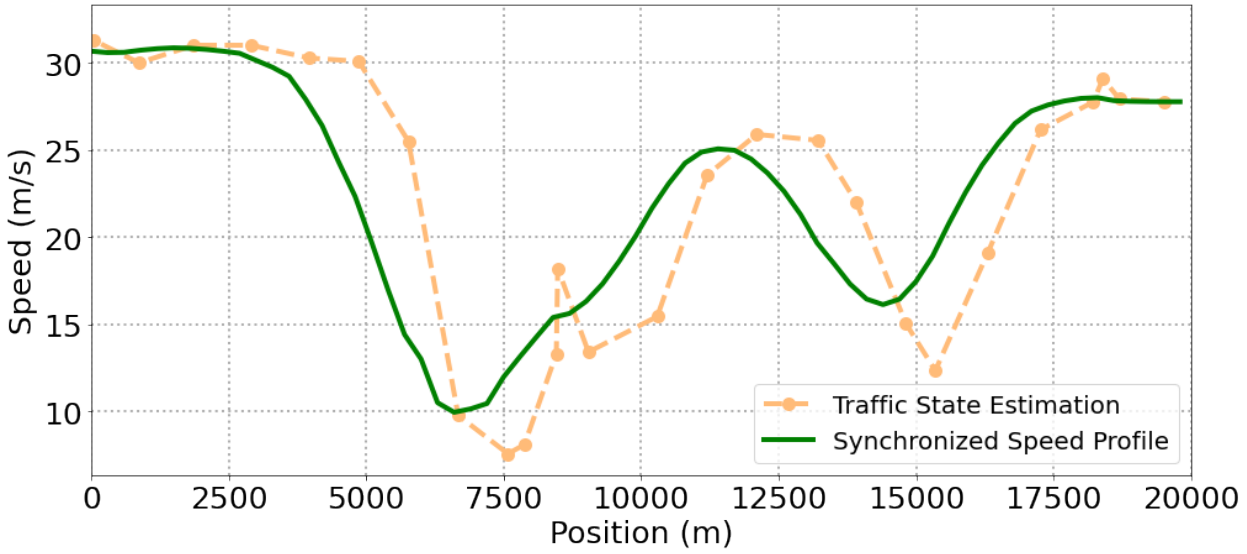


Figure 6.2: Example of the proposed synchronized speed profile for sample traffic state estimates. Average (aggregated) speeds across multiple segments (orange dots) are interpolated to a continuous profile. Uniform kernel as Eq. (6.6) expressed is applied to obtain the synchronized speed profile (green line).

estimation window. Figure 6.2 provides a visual interpretation of our proposed synchronized speed profile for sample traffic state estimates.

The desired speed locally is then estimated contingent on the headway between the subject and preceding vehicle. When the headway is relatively small, we focus on microscopic range, while when the headway is large, we focus on the macroscopic range. The relation between the two is smoothly weighted as follows.

$$v_{\text{des},\alpha} = \begin{cases} v_\alpha & \text{if } \ 0 \leq h_\alpha < 1 \\ (2 - h_\alpha)v_\alpha + (h_\alpha - 1)v_{\text{avg}} & \text{if } \ 1 \leq h_\alpha \leq 2 \\ v_{\text{avg}} & \text{if } \ h_\alpha > 2 \end{cases} \tag{6.7}$$

where v_α and h_α are the speeds and time headways of the subject vehicle, respectively.

6.4.2 Gap regulation

Our expert, in addition to harmonizing its speeds with other AVs, also regulates its spacing in a manner that is both safe and responsive to nearby events such the formation of large gaps. In particular, when provided traffic state information overestimates or underestimates the *actual* state of traffic, additional feedback mechanisms are employed to allow AVs to respond in a manner more reminiscent of adaptive cruise control [205]. The gap regulation part $(k_p(h_\alpha - h_{\text{des}}) + k_d(v_l - v_\alpha))$ in Eq. (6.5) is similar with the design of the ACC vehicle

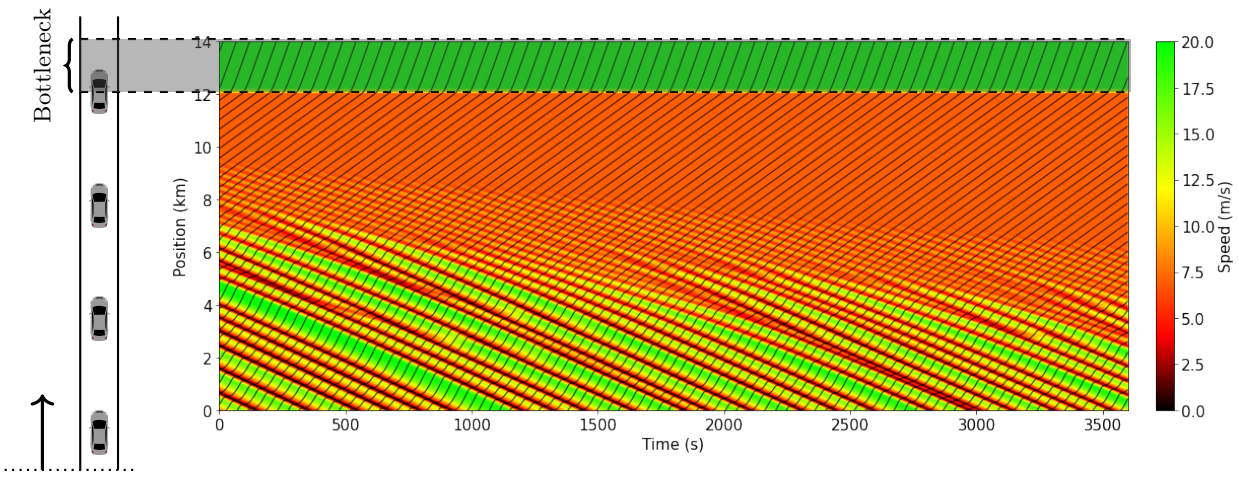


Figure 6.3: An illustration of the bottleneck problem and human-driven dynamics within it. A bottleneck placed near the end of the network restricts the flow of vehicles within this region and subsequent throughput of vehicles downstream. The buildup of dense traffic near the bottleneck also produces a spillover effect, resulting in the propagation of stop-and-go congestion.

model [196], and aims to maintain a desired time headways h_{des} while further reducing the discrepancy in speeds between ego and leading vehicles. The k_p and k_d terms, similar to other PD feedback control methods, specifies intensity of responses by AVs to such fluctuations. This design intends to capture the human-like car following behavior so that the subject vehicle can maintain a reasonable gap from the preceding vehicle. Equipped with the speed synchronization, our controller tries to drive smoothly while maintains reasonable gap with the anticipation of future oscillations in driving speeds.

6.4.3 Safety filter

Finally, to avoid potential collisions with the preceding vehicle from unforeseen driving events, we additionally restrict the magnitudes of assigned speeds by values imposed by a safety filter v_{fs}. The safety filter in our formulation is treated as a generic term to allow for flexible assignment. In this chapter, we adopt a simple method with the idea of maintaining a sufficiently large gaps, both in space an time, subject to the leading vehicles most recent fluctuations in speeds. This failsafe is used both by the expert policy as well as the imitated within the training and evaluation procedures. Details on this safety filter are provided in Section 6.8.2.

6.5 Experimental Setup

In this section, we describe the experimental procedure used to evaluate the efficacy of imitation learning in generating cooperative driving behaviors in congestion-prone settings. We first introduce two scenarios in which platoons, via some network-imposed phenomenon, frequently experience stop-and-go states of traffic, and then continue to provide details on the simulation and learning processes employed.

6.5.1 Scenarios

We design two scenarios aimed at evaluating the efficacy of the expert and learned responses against both static and active sources of congestion. These scenarios involve multiple agents interacting across large timescales and significantly altering the traffic states witnessed by one another, and as such are difficult to solve from a reinforcement learning perspective [101, 100]. These congested states of traffic, however, can be mitigated by the expert through knowledge of downstream inhomeginuoties in driving speeds, and as such are good candidates for evaluating the effects of imitation on generating traffic-regulating behaviors.

Capacity-restricted bottleneck

In this first scenario, we simulate the effects of traffic bottlenecks, static disruptions of vehicular traffic, in both fully-driven and mixed-autonomy settings. Bottlenecks of this nature emerge from various physical phenomena including traffic lights, lane reductions, etc. and reduce the capacity, or number of vehicles flowing through a certain region. This raises the density of traffic upstream, which when coupled with string-instabilities in human-driver dynamics introduce further disruptions in the form of stop-and-go traffic.

To simulate responses of vehicles to such bottlenecks, we assign a restriction to the capacity of traffic beyond a bottleneck region x_{BN}, here set to 12 km. This restriction is imposed by raising the desired time headway T within the model parameters of human-driven vehicles (Section 6.2.2). This, as described in [174], lowers the capacity of traffic with the bottleneck region in a manner inversely proportional to the increase in T, thereby producing the desired effect. Figure 6.3 depicts this behavior in the fully human-driven vehicles within this setting.

Shockwave response

Next, we simulate the response of vehicles to shockwaves, sudden and substantial changes in the state of traffic that act as active or moving bottlenecks [116]. Shockwaves generally emerge from sharp and sudden oscillations in driving speeds occurring forwards in traffic and resulting from, for instance, aggressive lane changing maneuvers, disturbances near on-ramps and off-ramps, or other events. These oscillations are then propagated by human drivers,

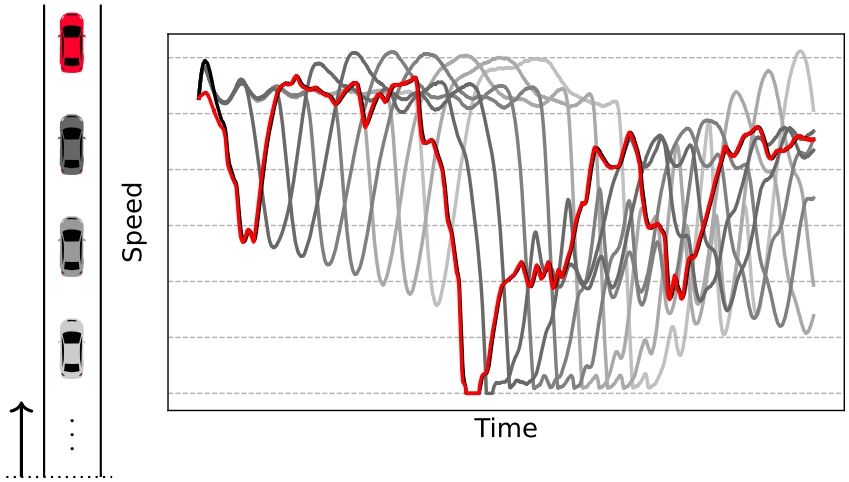


Figure 6.4: Sample response to a leading vehicle profile (in red) captured by the trajectories described in Section 6.5.1. Oscillations by following vehicles grow in magnitude, pointing to the phenomenon resulting in stop-and-go congestion.

who, unable to see the shockwave, maintain close distances with leading vehicles and as such are forced to break heavily when lead vehicles behind to oscillate.

To replicate the effects of realistic and highly stochastic shockwaves in simulation, we make use of the work presented by [100]. In this document, the authors synthesize driving responses to shockwaves in vehicular traffic by extracting trajectories from real-world congested states of traffic witness in Interstate I-24 in Nashville, Tennessee. Shockwaves here are imposed as boundary conditions by a leading vehicles which follow recorded trajectories consisting of position and velocity measurements $\tau := \{(x_1, v_1), \ \ldots, \ (x_T, v_T)\}$ sampled in increments of 0.1 seconds and varying in collection date and severity of congestion witnessed. Platoons of simulated vehicles are then placed behind such a leading trajectory and follow control laws assigned either by a human-driver model or a mixture of human-driver models and automated control strategies provided by the expert or trained policy. Figure 6.4 depicts the behavior of human-driven vehicles following one such trajectory (in red). Seen here, disturbances, experienced as oscillations in driving speeds, expand between subplatoons, thereby pointing to the string unstable nature of driving that propagates and worsens congested events.

6.5.2 Simulations

Simulations were conducted with a step size of 0.1 sec/step and a horizon of length 3600 seconds for bottleneck scenarios and the length of provided trajectories for schockwave scenarios. In each scenario, subplatoons of length $K = 25$ vehicles were assigned to mimic a AV market penetration of 4%. Platoons furthermore vary in length. In particular, for shockwave scenarios, we consider a fixed platoon of length 250 vehicles and containing in part 10 automated vehicles, while for bottleneck scenarios infinite length platoons are assigned

Parameter	Value (trajectory)	Value (bottleneck)
v_0	45 m/s	35 m/s
T	1 s	1.24 s
a	1.3 m/s^2	1.3 m/s^2
b	2.0 m/s^2	2.0 m/s^2
δ	4	4
s_0	2 m	2 m
ϵ	$\mathcal{N}(0, 0.3)$ m/s^2	$\mathcal{N}(0, 0.3)$ m/s^2

Table 6.1: Human model parameters

Parameter	Description	Value
k_p	Proportional gain	2.0
k_d	Differential gain	0.5
h_{des}	Desired time headway	2.0 s
w	Sliding window length for speed estimation	3000 m
$s_{\min}$	Minimum safe space headway	5.0 m
$h_{\min}$	Minimum safe time headway	0.5 s
τ_s	Safety decision-making horizon	5.0 s

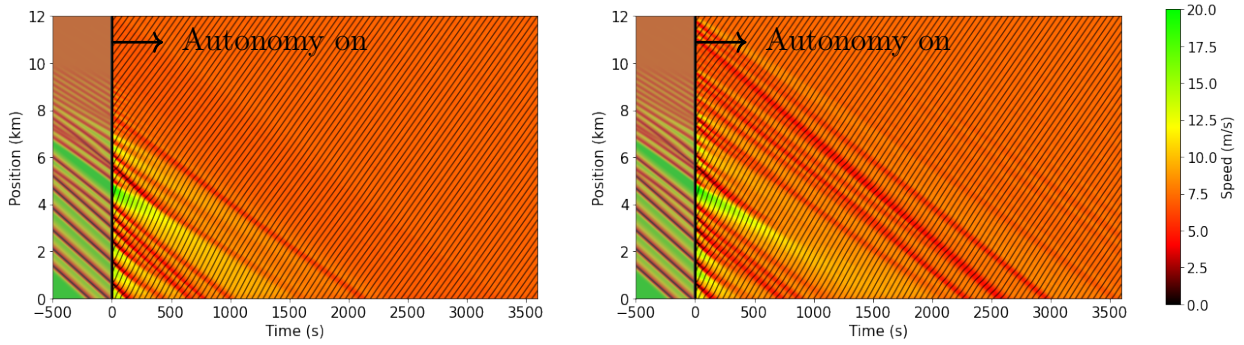
Table 6.2: Expert controller parameters

but evaluated solely from positions 0-12 km. All vehicles outside of the range 0-12 km act as human-driven vehicles, with vehicles beyond 12 km further exhibiting reductions in capacity as a result of their updated model parameters. The model parameters for both human-driven vehicles and expert policies within these simulations are provided in Tables 6.1 and 6.2, respectively.

Prior to training, leading vehicles are initially placed at position 0 and subsequent vehicles are placed behind it with equivalent spacings and speeds to replicate uniform-flow conditions[2]. Vehicles within bottleneck simulations are then further simulated for 3600 seconds initially to allow for the buildup of congested states of traffic prior to simulating mixed-autonomy behaviors. Afterwards, AVs are assigned control laws dictated either by the expert when constructing labeled demonstrations or by the training agent when performing learning.

In assigning expert actions, we collect edge-segment estimates using one of two approaches. For bottleneck scenarios, we directly compute aggregated speeds by return the average speeds

[2]The leading vehicle within bottleneck simulations operate at a constant speed throughout, but exhibit no noticeable effect on the flow of traffic as inter-vehicle spacings and speeds are dictated primarily by the intensity of the bottleneck.



(a) Global sensing (expert) (b) Local sensing (imitation)

Figure 6.5: Spatio-temporal responses by vehicles within the mixed-autonomy bottleneck with a bottleneck time headway of 1.5 s. **Left:** AVs following responses imposed directly by the expert quickly dissipate the propagation of stop-and-go traffic and maintain uniform driving speeds after. **Right:** The learned policy similarly, but more gradually, harmonizes the flow of traffic near the downstream condition imposed by the bottleneck.

of vehicles within each segment in real time. This is possible in this setting as vehicles are distributed throughout the network an may be accessed from the perspective of the simulator, thereby providing feedback on each segment. Alternatively, for shockwave scenarios, provided events downstream of the leading vehicle are not visible, we instead sample said estimates from historical data collected by Inrix [35]. These measurements are collected in segments of length approximately equal to 0.5 miles and are updated in increments of 5 minutes, and provide an added layer of realism to simulated mixed-autonomy responses.

6.5.3 Learning procedure

For all experiments in this chapter, we use DAgger [136, 137], an online imitation learning procedure for generating desired behavior from demonstrations. Labels from expert policies in this setting match the work of [81]. In particular, observations are set as temporal concatenations of local vehicle speeds and headways to capture non-local phenomena via fluctuations in local driving conditions, while actions are defined as relative changes in the desired speed by an expert policy from the driving speeds vehicles current operate under. For both scenario, we initialize the demonstration dataset with samples collected from 10 instantiations of the simulation environments, and then continue data collection after subsequent policy updates, rerunning one new simulation after each update for a total of 50 iterations. Note that this, in itself, is more efficient than classic reinforcement learning strategies, which require dozens of rollouts to effectively perform a single policy update step. Finally, a Multi-Layer Perceptron is used with hidden layers (32, 32, 32), and a `ReLU` nonlinearity. This policy is updated using Adam [79] with a learning rate of 0.001, and dropout [154] is employed to allow for increased robustness.

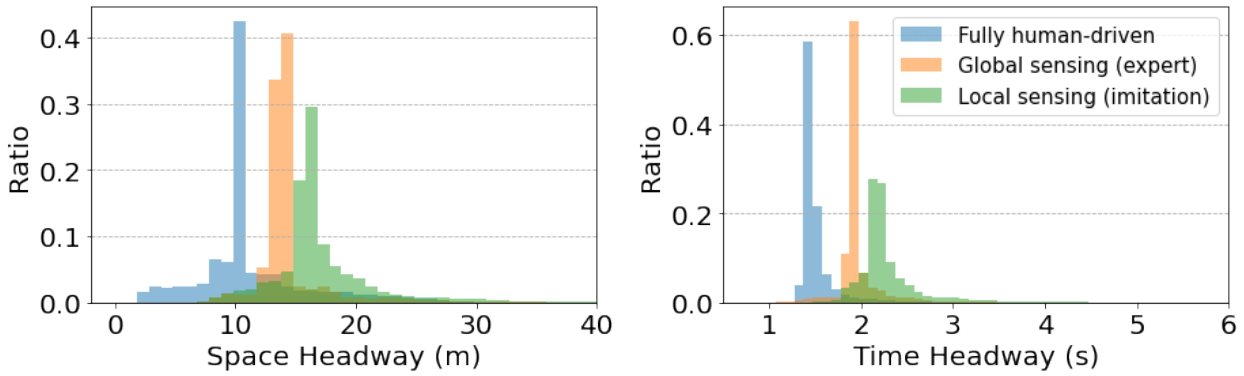


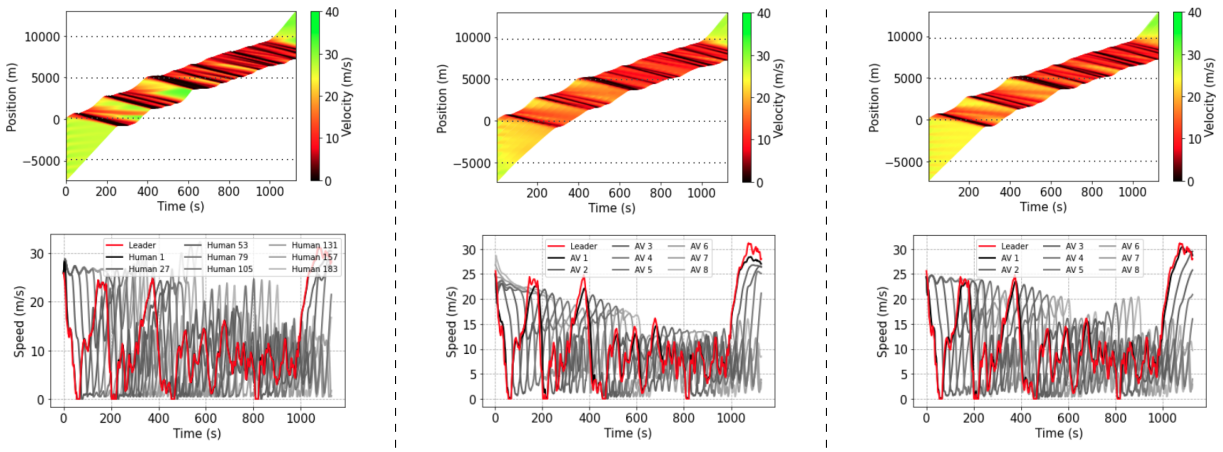
Figure 6.6: Accepted space and time headway by AVs within the bottleneck simulations compared to human drivers (blue).

Bottleneck time headway [s]	Simulation type	Throughput [veh/hr]	Miles-per-gallon (AVs)	Miles-per-gallon (total)
1.4	Human-driven	1858 → 1862	–	42.32
	Global Sensing	1917 → 1874	57.93 (+36.90%)	58.15 (+37.41%)
	Local Sensing	1857 → 1876	57.06 (+34.83%)	58.63 (+38.54%)
1.5	Human-driven	1740 → 1709	–	43.10
	Global Sensing	1756 → 1712	52.20 (+21.11%)	52.21 (+21.14%)
	Local Sensing	1748 → 1715	52.51 (+21.83%)	52.22 (+21.16%)
1.6	Human-driven	1638 → 1592	–	41.95
	Global Sensing	1633 → 1593	47.40 (+12.99%)	47.37 (+12.92%)
	Local Sensing	1631 → 1594	47.37 (+12.92%)	47.28 (+12.71%)
1.7	Human-driven	1512 → 1500	–	39.47
	Global Sensing	1530 → 1500	43.56 (+10.36%)	43.56 (+10.36%)
	Local Sensing	1530 → 1500	43.51 (+10.24%)	43.44 (+10.06%)
1.8	Human-driven	1458 → 1425	–	38.71
	Global Sensing	1455 → 1424	40.49 (+4.63%)	40.50 (+4.65%)
	Local Sensing	1456 → 1425	40.46 (+4.52%)	40.37 (+4.29%)

Table 6.3: Performance of human-driven and mixed-autonomy simulations within bottleneck simulations. Throughputs are computed near the start of the network (at 1 km) and bottleneck (at 9 km) to measure both the effects of control strategies on the mobility of vehicles through the bottleneck and whether any alterations emerge at the cost of restricting traffic upstream.

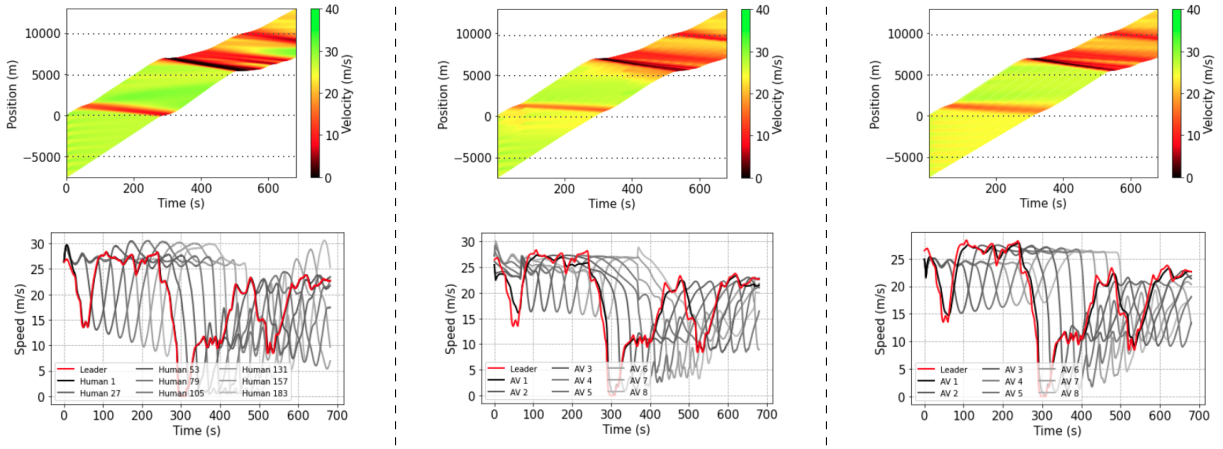
			Distance traveled (km)	Miles-per-gallon (AVs)	Miles-per-gallon (total)
Light/Moderate		Human-driven	13.71	–	45.41
	Experiment 1	Global Sensing	13.57 (−1.02%)	49.89 (+9.87%)	50.85 (+11.98%)
		Local Sensing	13.51 (−1.46%)	51.12 (+12.57%)	51.10 (+12.53%)
		Human-driven	13.86	–	39.45
	Experiment 2	Global Sensing	13.79 (−0.51%)	42.55 (+7.86%)	42.68 (+8.19%)
		Local Sensing	13.59 (−1.95%)	42.40 (+7.48%)	43.23 (+9.58%)
		Human-driven	14.58	–	40.36
	Experiment 3	Global Sensing	14.46 (−0.82%)	44.93 (+11.32%)	44.73 (+10.83%)
		Local Sensing	14.38 (−1.37%)	44.57 (+10.43%)	44.03 (+9.09%)
		Human-driven	14.09	–	40.46
	Experiment 4	Global Sensing	14.02 (−0.50%)	48.21 (+19.15%)	49.84 (+23.18%)
		Local Sensing	13.89 (−1.42%)	47.52 (+17.45%)	48.22 (+19.18%)
		Human-driven	13.23	–	44.19
	Experiment 5	Global Sensing	13.12 (−0.83%)	46.14 (+4.41%)	45.94 (+3.96%)
		Local Sensing	12.81 (−3.17%)	52.58 (+18.99%)	48.81 (+10.45%)
		Human-driven	14.24	–	39.79
	Experiment 6	Global Sensing	14.17 (−0.49%)	46.24 (+16.21%)	46.64 (+17.22%)
		Local Sensing	14.09 (−1.05%)	44.41 (+11.61%)	45.75 (+14.98%)
		Human-driven	14.48	–	38.65
	Experiment 7	Global Sensing	14.34 (−0.97%)	48.12 (+24.50%)	48.61 (+25.77%)
		Local Sensing	14.25 (−1.59%)	47.41 (+22.66%)	47.03 (+21.68%)
		Human-driven	14.03	–	41.19
	Average	Global Sensing	13.92 (−0.73%)	46.58 (+13.10%)	47.04 (+14.21%)
		Local Sensing	13.79 (−1.70%)	47.14 (+14.46%)	46.88 (+13.83%)
Heavy		Human-driven	13.32	–	36.67
	Experiment 8	Mixed-autonomy	13.29 (−0.23%)	42.95 (+17.13%)	41.75 (+13.85%)
		Local Sensing	13.14 (−1.35%)	41.11 (+12.11%)	40.65 (+10.85%)
		Human-driven	13.10	–	36.34
	Experiment 9	Global Sensing	13.04 (−0.46%)	52.48 (+44.41%)	48.72 (+34.07%)
		Local Sensing	12.96 (−1.07%)	45.81 (+26.06%)	44.75 (+23.14%)
		Human-driven	10.70	–	30.97
	Experiment 10	Global Sensing	10.64 (−0.56%)	38.48 (+24.25%)	37.55 (+21.25%)
		Local Sensing	10.43 (−2.52%)	33.48 (+8.10%)	35.60 (+14.95%)
		Human-driven	12.37	–	34.66
	Average	Global Sensing	12.32 (−0.40%)	44.64 (+28.78%)	42.67 (+23.12%)
		Local Sensing	13.30 (−1.59%)	45.04 (+15.79%)	44.92 (+16.37%)
		Human-driven	13.53	–	39.23
Overall Average		Global Sensing	13.44 (−0.64%)	46.0 (+17.26%)	45.73 (+16.57%)
		Local Sensing	13.30 (−1.67%)	45.04 (+14.82%)	44.92 (+14.50%)

Table 6.4: Performance of both fully human-driven and mixed-autonomy simulations, individually and average across 10 runs for each leading trajectory sampled from I-24. Figures 6.8 and 6.7 correspond to experiments 1 and 9, respectively.



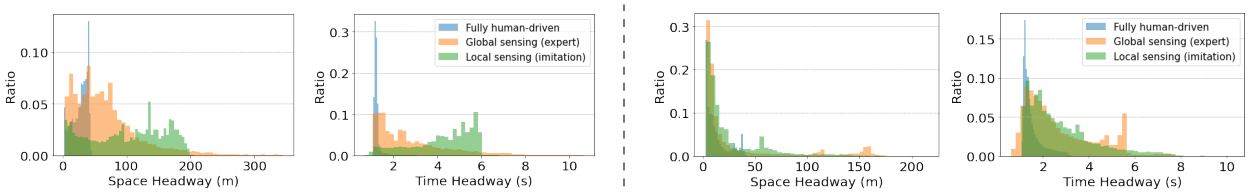
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.7: Performance of vehicles in the presence of sharp and frequent oscillations in driving speeds. Automated vehicles whose actions are supplied both by the global sensing expert as well as the imitated policy do well in reducing the intensity of oscillations by following vehicles, with the expert policy slightly outperforming that which imitates it.



(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.8: Performance of vehicles in the presence of moderate/infrequent oscillations in driving speeds. Automated vehicles in both settings succeed in reducing the magnitude of oscillations in driving speeds resulting from aggressive human driving behaviors. The imitated policy, however, unlike the expert policy does not slow down in anticipation of the shockwave, but rather incrementally alleviates its effects by accepting larger headways and slowing down in a smoother fashion.



(a) Moderate / infrequent oscillations

(b) Heavy / frequent oscillations

Figure 6.9: Accepted space headway and time by human-driven and automated vehicles within the shockwave simulations. As we can see, automated vehicles primarily maintain reasonable headways that distributionally match those of human-driven vehicles but are willing to adopt larger gaps when required to avoid future anticipated congestion.

6.6 Numerical Results

In this section, we present numerical results for the proposed controller and imitation learning procedure across each of the previously described environments. Through these results, we demonstrate 1) that imitation learning procedures are capable of extracting behaviors from global sensing experts that, while distributionally different under certain settings, on aggregation match the behaviors of such experts in critical moments, and 2) that such learned longitudinal responses significantly improves the energy-efficiency of vehicles in comparison to human car-following dynamics.

6.6.1 Evaluation metrics

We evaluate the response of vehicles of vehicles in each setting across the following metrics:

Energy-efficiency. To analyze the performance of the proposed controller in terms of energy efficiency, we adopt a semi-principled energy model that has a physics-based component. This model is described in Section 6.8.1. The energy consumption obtained from the model will be converted into Miles-Per-Gallon (MPG) as the metric to indicate energy efficiency.

Throughput. Next, to evaluate our controller on mobility improvement, throughput can be a good metric to indicate the actual amount of traffic flowing in our target segment. However, in the shockwave scenario, since the simulation experiment will end if it reaches the end of the leading trajectory, regulation on controlled vehicles may reduce the throughput near and upstream of these vehicles. For fixed regions, measuring the distanced traveled can be an equivalent representation of measuring the traffic flow. Therefore, we use controlled vehicles' travel distance as a representation of the throughput.

Proximity to leader. Close proximity may denote unsafe driving behaviors while large distances between vehicles may denote reductions in throughput and may encourage cut-ins

and cut-outs by following vehicles. We use time headway and space gap as the metrics to measure the proximity to leader.

6.6.2 Generalization of learned control laws

For both scenarios, we ensure that both the learned and expert responses are robust to traffic conditions by varying the response and severity of traffic conditions between simulations.

Bottleneck.

In the bottleneck scenario, we sampled different desired time headway T from [1.4, 1.8] in the IDM model to represent different level of traffic congestion. As the bottleneck time headway increases, the capacity of the simulated bottleneck should decrease, referring to a more severe congestion.

Shockwave.

For the shockwave scenario, among the drives recorded within Interstate I-24, we select trajectories collected during morning peak demand intervals (6am - 7am) and exhibit some degree of sharp oscillations in driving speeds. This amounts to a total of 10 varying trajectories, seven of which, we note, observe what may deemed as *light-to-moderate* congestion (e.g. Figure 6.8), whereby vehicles alternate between free-flowing and congested states of traffic, while the remaining three exhibit more *severe* forms of congestion (e.g. Figure 6.7), whereby driving speeds are consistently slow and stop-and-go behaviors are frequent.

6.6.3 Performance

To make it easier to compare the performance of our proposed controller, baseline experiments that replace AV with IDM simulated human-driver vehicle are conducted. The expert control strategy that equipped with global sensing, which utilizes downstream traffic states to acquire designed velocity, is also evaluated in each experiment for comparison purposes.

Bottleneck

As Table 6.3 shows, among all cases, our expert control strategy with global sensing can obtain significant energy improvement while maintaining the equivalent level of throughput. The imitation learning controller with only local sensing successfully matches the expert behavior of harmonizing the traffic flow as Figure 6.5 describes, and with similar energy efficiency and throughput, while for the proximity to leader, it tend to be more conservative with larger gaps as Figure 6.6 illustrates.

Shockwave

Table 6.4 depicts the average performance of the system on all 10 utilized trajectories for the metrics we described above. We can see significant improvements on energy efficiency at little cost to the throughput of the system. Comparing to the baseline, the imitation learning controller provides on average 14.50% savings to energy consumption with only 1.67% reduction on distance travelled by the controlled vehicle. This is true for both heavy (Figure 6.7) and moderate/light (Figure 6.8) forms of congestion, with benefits from each policy being achieved through slightly different mechanisms. In particular, as seen in Figure 6.9, the imitation learning controller maintains both space headway and time headway spread in a wider range, leaving a more conservative gap with the preceding vehicle within moderate oscillations.

With the knowledge of the congestion at downstream, the controlled vehicles should deliberately leave more gap to avoid sharp deceleration, as a result, drive in a smoother speed and save energy consumption. This behavior propagates to other vehicles immediately upstream of the automated vehicles as well, resulting in more uniform driving speeds throughout the platoon. This is for instance true in Figure 6.8, where AVs dampen the magnitude of oscillations experienced by consecutive vehicles within the platoon. In contrast, oscillations in driving speeds in fully human-driven setting are amplified by trailing drivers within the platoon, resulting in the subsequent formation of stop-and-go traffic.

6.7 Chapter Summary

This chapter demonstrates the efficacy of imitation learning systems in deriving traffic regulation policies for decentralized, mixed-autonomy car-following systems. In it, we construct a simple expert policy in which labeled actions exploit knowledge of global states of traffic to assign centralized speed profiles that AVs may adhere to restrict the propagation of stop-and-go traffic. We then demonstrate that automated systems trained to imitate such signals succeed in doing so near-perfectly without access to global states and traffic.

6.8 Appendix

6.8.1 Energy model

To analyze the performance of the proposed controller in terms of energy efficiency, we adopt a semi-principled energy model that has a physics-based component [92]. The model takes as inputs the instantaneous vehicle speed v, acceleration a, and road grade θ, and outputs engine speed, engine torque, fuel consumption, gear, transmission output speed, wheel force, wheel power, and feasibility of the given (v,a,θ) with respect to engine speed and engine torque. In our training process, we take Toyota RAV4 as the prototype vehicle and simplify

the energy model to a fitted polynomial model of the form

$$g(v,a) = max(f(v,a),\beta) \tag{6.8}$$

where

$$\begin{aligned} f(v,a) = {} & C_0 + C_1 v + C_2 v^2 + C_3 v^3 \\ & + p_0 a + p_1 a v + p_2 a v^2 \\ & + q_0 a_+^2 + q_1 a_+^2 v \end{aligned} \tag{6.9}$$

and $a_+ = \max(a,0)$, and β is the minimum fuel rate, which is not necessarily zero because different vehicles have different criteria for enacting a fuel cut.

The energy consumption obtained from the model is converted into Miles-Per-Gallon (MPG) as the metric to indicate energy efficiency.

6.8.2 Safety filter details

To avoid any potential collisions with the preceding vehicle, we add a safety filter to the proposed controller. Our safety filter design is inspired by the simple idea that the gap between the preceding vehicle and the subject vehicle should be larger than the minimum space gap and the headway between the two vehicles should also be larger than the minimum time headway. We start with the time headway requirement and add a heuristic in terms of the space gap requirement. The upper bound of the velocity determined by safety filter design is calculated by Eq. (6.10)

$$v_{\text{fs}} = \frac{s - s_{\min} + v_l \tau_s + \frac{1}{2} a_l \tau_s^2 - \frac{1}{2} v \tau_s}{h_{\min} + \frac{1}{2}\tau_s} \tag{6.10}$$

where s is the space gap between the preceding vehicle and the subject vehicle; $s_{\min}$ is the minimum space gap between the two vehicles; τ_s is the decision making horizon; a_l is the acceleration of the preceding vehicle[3]; $h_{\min}$ is the minimum time headway between the two vehicles and the rest of the variables follow the definitions from the above equations.

The detailed design procedure is as follows: Eq. (6.11) shows the requirement on time headway at time $t = t + \tau_s$.

$$\frac{x_l(t+\tau_s) - x(t+\tau_s)}{v(t+\tau_s)} \geq h_{\min} \tag{6.11}$$

where $x_l(t+\tau_s)$ is the position of the preceding vehicle at time $t+\tau_s$; $x(t+\tau_s)$ is the position of the subject vehicle at time $t+\tau_s$; $v(t+\tau_s)$ is the speed of the subject vehicle at time $t+\tau_s$ and the rest of the variables follow the definitions from the above equations. Following

[3]While multiple estimates for this acceleration may be assigned in practice, we take average on the previous 5 seconds acceleration of the preceding vehicle as the estimation in the simulations conducted in this chapter.

simplified dynamics of motion, the position of each the ego and preceding vehicles at time $t + \tau_s$ is:

$$x_l(t + \tau_s) = x_l(t) + v_l(t)\tau_s + \frac{1}{2}a_l(t)\tau_s^2 \tag{6.12}$$

$$x(t + \tau_s) = x(t) + v(t)\tau_s + \frac{1}{2}a(t)\tau_s^2 \tag{6.13}$$

where $a(t)$ is the target decision variable, and for a fixed desired speed across the decision-making horizon may be expressed as:

$$a(t) = \frac{v_{\text{des}} - v(t)}{\tau_s} \tag{6.14}$$

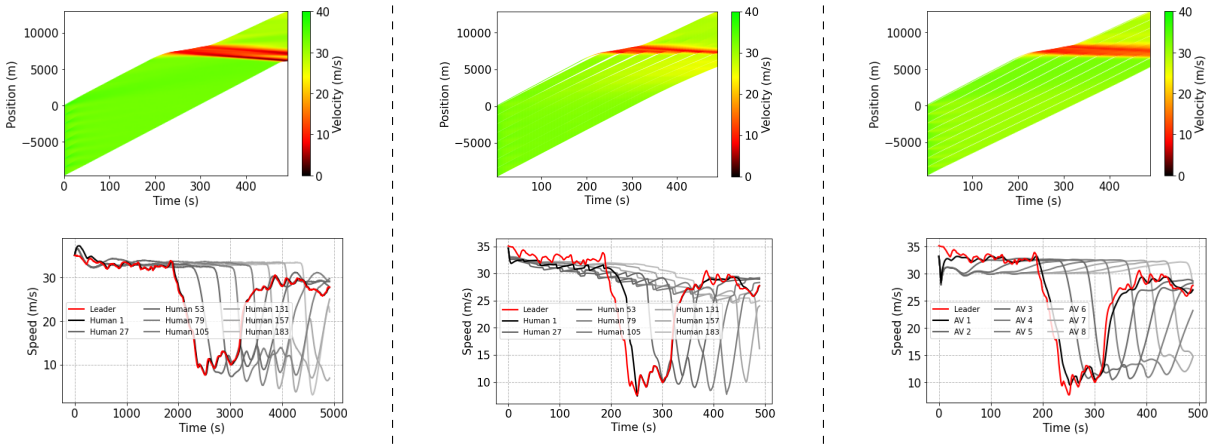
Plugging in Eq. (6.12), Eq. (6.13), and Eq. (6.14) to solve Eq. (6.11), we can get an initial upper bound of v:

$$v \leq \frac{s + v_l\tau_s + \frac{1}{2}a_l\tau_s^2 - \frac{1}{2}v\tau_s}{h_{\text{min}} + \frac{1}{2}\tau_s} \tag{6.15}$$

We also want to take the space gap requirement into consideration and get a slightly tighter upper bound. To achieve this, we add an heuristic by replacing the s in Eq. (6.15) with s - s_{min} which leads to our final design of the safety filter velocity as Eq. (6.10) shows.

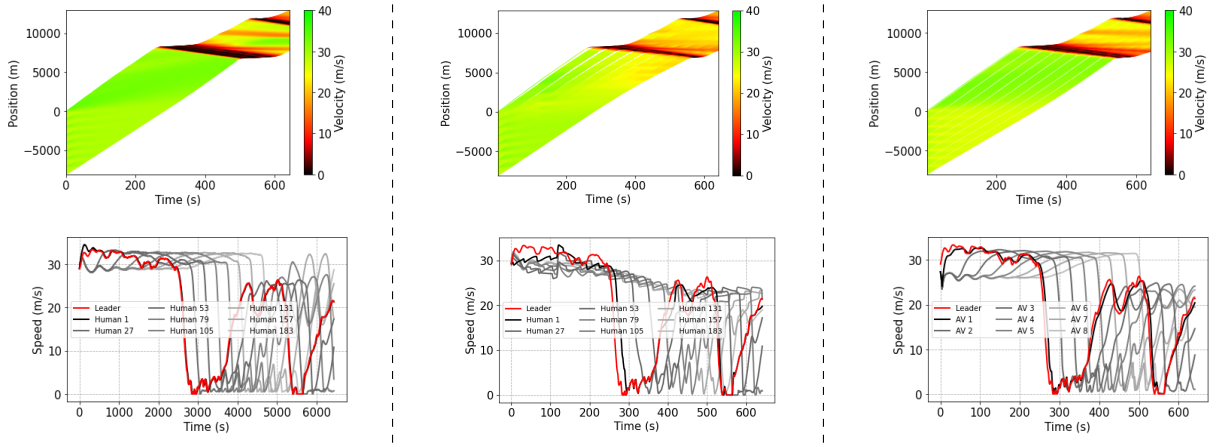
6.8.3 Additional results: Shockwave response

Similar to the subsection prior, for the purposes of completeness, Figures 6.10 to 6.17 provide illustrations of each of the fully-human driven and mixed-autonomy simulations for all remaining leading trajectories samples from Interstate I-24. As seen in these figures, the policy learned via imitation does in fact succeed at providing a close approximation to the behavior by the expert across all settings, and in doing so restricts the growth in perturbations by following vehicles for a majority of the time.



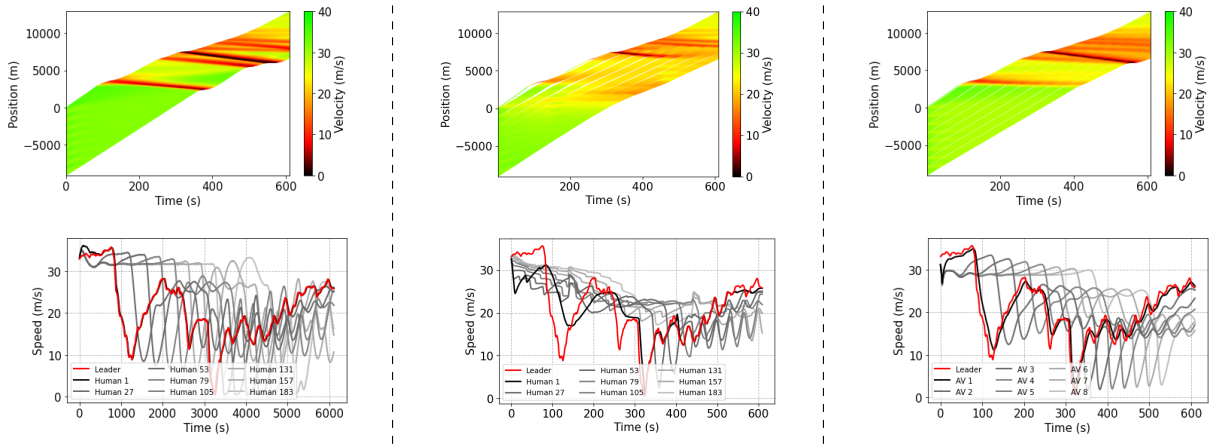
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.10: Simulation results corresponding to Experiment 2 in Table 6.4



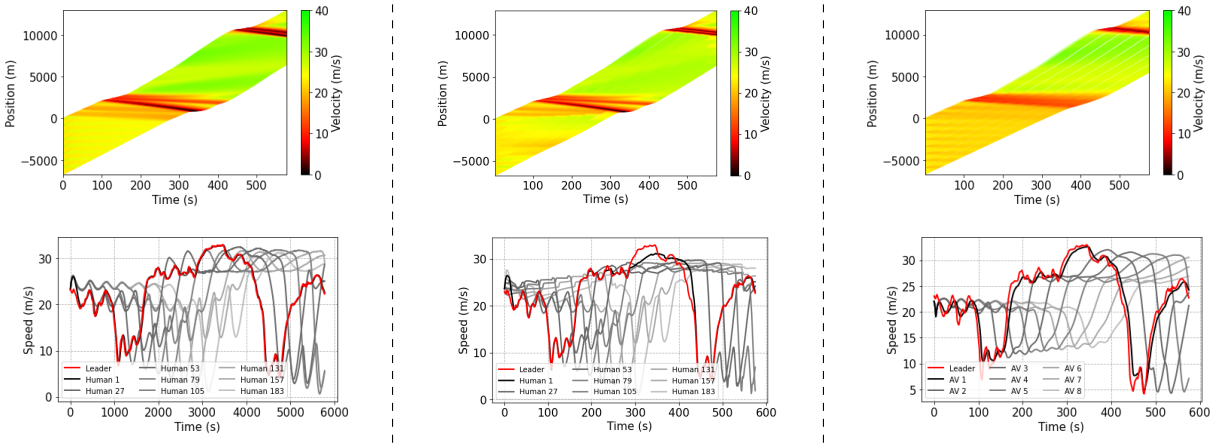
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.11: Simulation results corresponding to Experiment 3 in Table 6.4



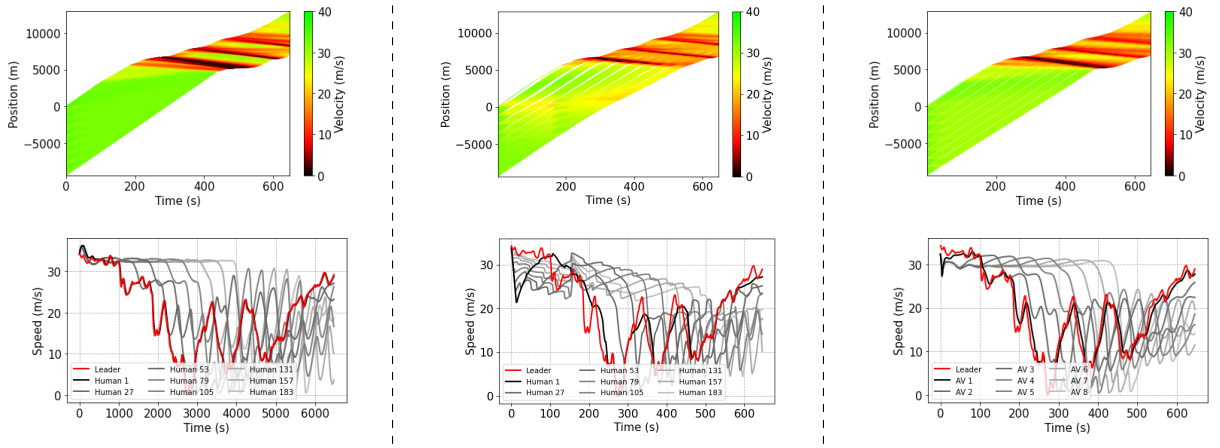
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.12: Simulation results corresponding to Experiment 4 in Table 6.4



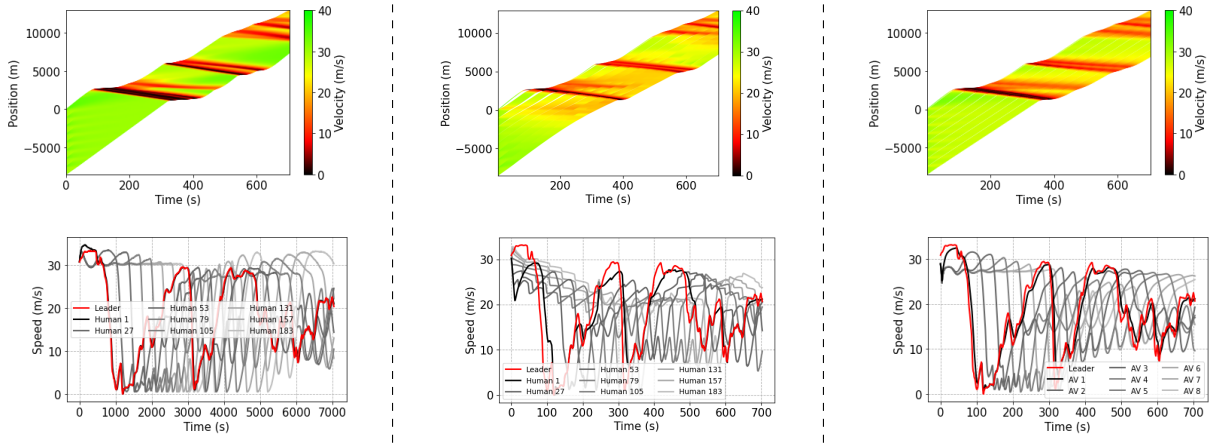
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.13: Simulation results corresponding to Experiment 5 in Table 6.4



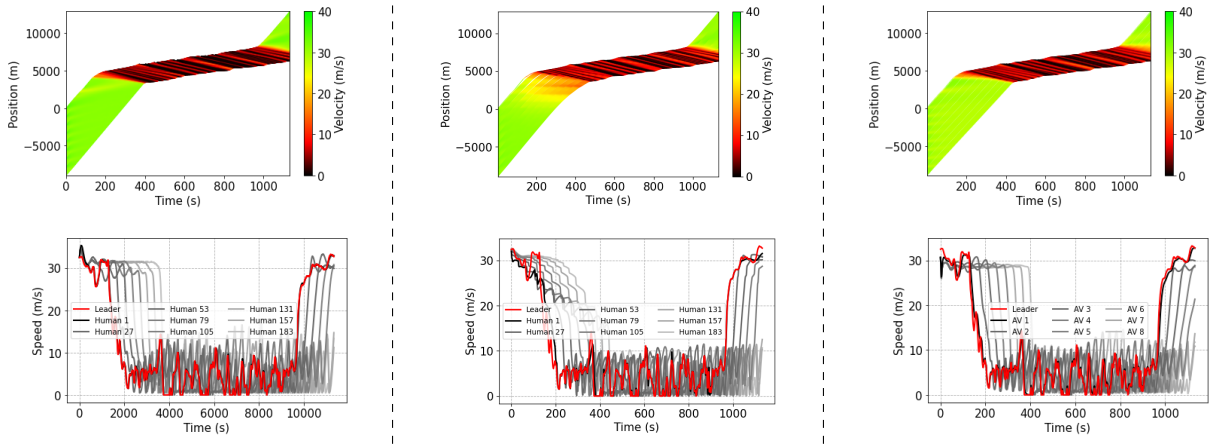
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.14: Simulation results corresponding to Experiment 6 in Table 6.4



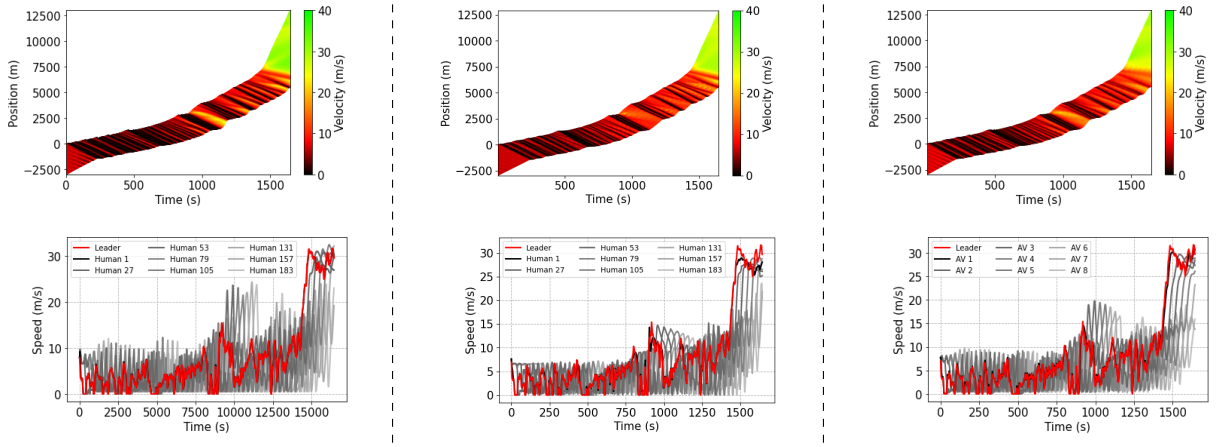
(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.15: Simulation results corresponding to Experiment 7 in Table 6.4



(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.16: Simulation results corresponding to Experiment 8 in Table 6.4



(a) Fully human-driven (b) Global sensing (expert) (c) Local sensing (imitation)

Figure 6.17: Simulation results corresponding to Experiment 10 in Table 6.4

Part III

Generalizing to Complex-to-model Dynamics

CHAPTER 7

Dissipating Stop-and-go Waves in Closed and Open Networks via Deep Reinforcement Learning

In this part, we present approaches for alleviating limitations associated with the computational cost of training RL algorithms in mixed autonomy settings through the use of simplified auxiliary tasks. As mentioned in Chapter 3.3.2, generating microscopic simulations of real-world traffic networks is a slow and costly endeavour. This bottleneck arises primarily from the use of general-purpose traffic microsimulators such as SUMO, which are designed to recreate a broad range of traffic settings as opposed to efficiently generating millions of samples for a narrow set of tasks. The process of simulating stop-and-go waves in a single lane ring road, however, is a relatively light procedure, simply requiring the computation of a sequence of integrals of a number of car-following and AV models at every timestep This makes training on a simulated ring road network highly desirable from a deep RL perspective. A natural question however arises: Are policies learned on a ring road transferable to more realistic representations of traffic?

In this chapter, we explore the transferability of policies learned on a closed ring road network to an open highway networks with instabilities arising from an on-ramp merge. Through this study, we demonstrate that, when trained on a ring road networks exhibiting similar dynamics and network densities as its target network, learned policies can in fact perform well on structurally different networks without any exposure to the network. This result suggests that a considerable portion of training can be performed before exposure more computationally expensive, albeit accurate, models of traffic.

This chapter is adapted from [82]. Videos of the results are available at: `https://sites.google.com/view/itsc-dissipating-waves`.

7.1 Introduction

Traffic congestion is a severe problem in road networks across the world. In the United States alone, the cost of traffic congestion was estimated to be 305 billion USD in 2017, costing the average driver in large cities around 2000 USD. Developing new road infrastructure provides a

natural means of coping with the ever growing demand for mobility, but is expensive and time consuming, rendering it infeasible in most situations. Instead, considerable research in the area of *intelligent transportation systems* (ITS) has been performed to achieve a more efficient road usage, thereby increasing the capacity of road networks. This has led to significant improvements in traffic control methods such as traffic signal control, ramp metering, variable speed limits, and adaptive cruise control (ACC).

Over the past years, RL has led to a considerable amount of successes in performing control and strategy-driven tasks, such as playing Atari games at superhuman levels [114], and outperforming champions in the challenging strategy game Go [152]. This has prompted researchers in the transportation community to apply RL techniques on a multitude of intelligent transportation system tasks including traffic signal timing [96, 198, 2], ramp metering [16, 144], and variable speed limit control [144, 98].

Recently, RL has also been used in conjunction with state-of-the-art microscopic traffic simulations tools to train automated vehicles to improve traffic conditions through vehicle to vehicle interactions. In [202, 201], automated vehicles were trained using deep RL to improve system-level traffic flow conditions in a variety of closed and looped network settings, where the actions of a single vehicle can quickly propagate and affect the performance of all other vehicles in the network. In a variable length ring road, for instance, an RL agent with only local observability and controlling approximately 5% of automated vehicles learned to successfully dissipate stop-and-go waves within the network, thereby allowing the human-driven vehicles to travel at their optimal speeds. However, the applicability of the proposed controllers to *open*, as opposed to closed, network traffic scenarios is not addressed. This discrepancy is important, as it unclear whether a small percentage of automated vehicles interacting in a setting where they have significantly less control can in fact improve traffic.

The present chapter expands on the work described in [202]. The key contributions of this chapter are as follows:

- Using deep reinforcement learning, this chapter presents a traffic control strategy that may be employed by a series of connected automated vehicles to dissipate the effects of stop-and-go waves on open single lane road networks. This controller succeeds at eliminating nearly all stop-and-go waves in simulation with as little as 10% automated vehicle penetration.

- We also demonstrate that, through deep reinforcement learning, control strategies developed to improve traffic conditions in closed networks can be transferred and fined-tuned to handle realistic open network problems.

The remainder of the chapter is organized as follows. Section 7.2 provides an overview of RL and transfer learning, and discusses characteristic features differentiating the formation and propagation of stop-and-go waves in closed and open networks. Section 7.3 outlines the RL and transfer learning problem formulation for dissipating the formation and propagation of stop-and-go waves in open straight highway networks. Finally, Section 7.4 presents the

findings and results of a number of computational experiments conducted over various automated vehicle penetration rates.

7.2 Preliminaries

7.2.1 MDPs and reinforcement learning

Reinforcement learning problems are generally studied as a discrete-time Markov decision problem (MDP) [17], defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, T)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is an n dimensional state space, $\mathcal{A} \subseteq \mathbb{R}^m$ an m dimensional action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ a transitional probability function, $r : \mathcal{S} \to \mathbb{R}$ a bounded reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_+$ an initial state distribution, $\gamma \in (0, 1]$ a discount factor, and T a time horizon. For partially observable tasks, which conform to the interface of a *partially observable Markov decision process* (POMDP), two more components are required, namely Ω, a set of observations, and $\mathcal{O} : \mathcal{S} \times \Omega \to \mathbb{R}_+$, the observation probability distribution.

In a Markov decision process, an *agent* receives sensory inputs $s_t \in \mathcal{S}$ from the the environment and interacts with this environment by performing actions $a_t \in \mathcal{A}$. The agent's actions are defined by a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ parametrized by θ. Common policies used in continuous control tasks include artificial neural networks with multiple hidden layers [63] and recurrent neural networks capable of storing internal memory from previous states [68, 31].

The objective of the agent is to learn an optimal policy:

$$\theta^* := \text{argmax}_\theta \eta(\pi_\theta) \tag{7.1}$$

where $\eta(\pi_\theta) = \sum_{i=0}^{T} \gamma^i r_i$ is the expected discounted return across a trajectory $\tau = (s_0, a_0, ...)$, $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t|s_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$, for all t. In the present chapter, these policy parameters are iteratively updated using policy gradient methods [163].

7.2.2 Transfer learning between MDPs

Transfer learning techniques in reinforcement learning provide methods of leveraging experiences acquired from training in one task to improve training on another [170]. These tasks may differ in the agent-space (the observation the agent perceives or the actions it may perform), and in the MDP-space (such as the transition probability $\mathcal{P}$). For instance, in the classical cartpole control problem, where a cart moving left and right attempts to balance a pole vertically, the task may be modified in the second stage of training by increasing the gravitational force applied to the pole. Common transfer learning practices include sharing policy parameters θ and state-action pairs $< s, a, r, s' >$ between tasks. For a survey of transfer learning techniques, we refer the reader to [170, 126].

7.2.3 Stop-and-go waves in closed and open networks

The present chapter studies traffic control in the context of microscopic (car-following) models, where the dynamics of each individual vehicle of index α in a network is described by ordinary and delayed differential equations of the form:

$$\begin{cases} \frac{\mathrm{d}h_\alpha}{dt} = v_l(t) - v_\alpha(t) \\ \frac{\mathrm{d}v_\alpha}{dt} = f(h_\alpha(t-\tau), v_\alpha(t-\sigma), v_l(t-\kappa)) \end{cases} \tag{7.2}$$

where f is an acceleration equation, $v_\alpha(t)$ is the speed of the vehicle, $h_\alpha(t)$ is its headway with the leading vehicle l, and τ, σ, and κ are time delays. These models form the basis for the transitions of the MDPs studied in this chapter.

The optimal performance of a system of human-driven vehicles following a homogeneous car-following model is characterized by its uniform equilibrium flow. At this equilibrium, all vehicles in the network move at a constant speed v^* and with constant spacing h^*, such that:

$$f(h^*, v^*, v^*) = 0 \tag{7.3}$$

Highway traffic does not naturally remain within its uniform equilibrium flow, but rather experiences the formation of backwards propagating waves sometimes causing part of the traffic to come to a complete stop. This behavior is often attributed to inherent instabilities in human driving dynamics. Specifically, linear string stability formalizes how small disturbances brought about by lane changes, noise, etc. propagate to vehicles upstream and expand until a jam is formed [66].

Numerous articles have studied the nonlinear traffic properties of the formation and propagation of stop-and-go waves in the context of closed-network ring roads [155]. Considering a system of homogeneous vehicles in a closed single lane highway of variable length, the authors of [122] deduced the existence of two Hopf bifurcation points for densities in which the uniform flow equilibrium loses stability. These findings denote the existence of stable "stop-and-go" limit cycles within closed networks of certain densities. This is further illustrated in [156], in which field experiments performed with 22 vehicles in a 230 m ring road demonstrated the formation of traffic jams, even in the absence of external perturbations to the network.

Closed-network analysis of microscopic traffic dynamics such as the ones described in the previous section have shaped the way we attempt to counteract stop-and-go traffic; therefore, an understanding of the transferability of the subsequent designed controllers to open network traffic is paramount. In terms of the MDPs these networks produce, the primary disparity arises from the assumption of periodic boundary conditions at the start and end of the highways, whereby the state of the last vehicle in the network affects the actions of the first. The relaxation of this boundary condition, coupled with the concept of convective stability which asserts that traffic waves can only travel upstream [194], negates the existence of stable stop-and-go dynamics in finite-length open networks, as any wave that forms eventually propagates out of the network. Instead, persistent congested patterns within convectively unstable open networks is attributed to periodic perturbations brought about by bottleneck

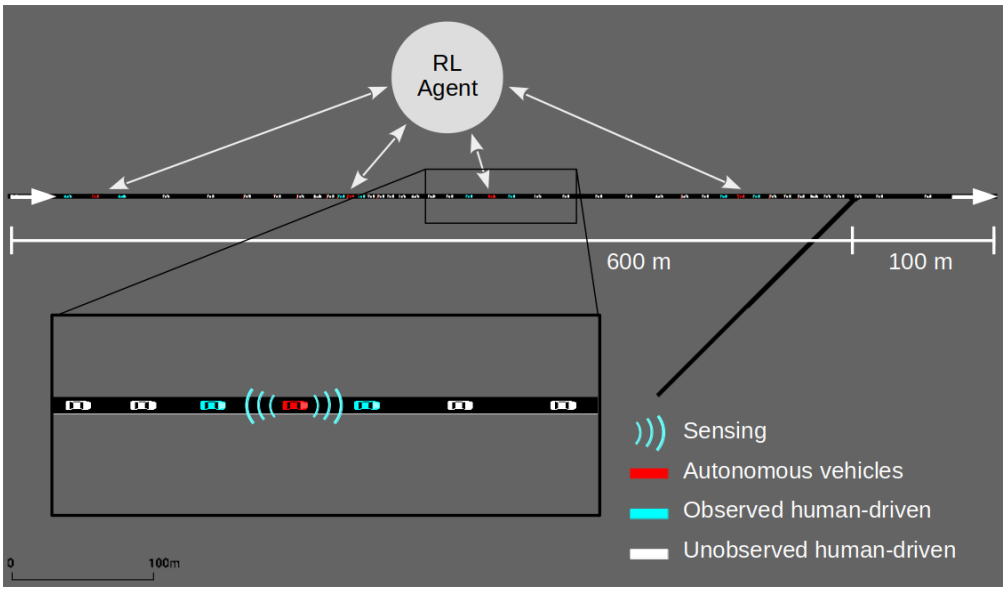


Figure 7.1: Open network highway network of length 700 m and inflow rate 2000 veh/hr with an on-ramp of inflow rate 100 veh/h. Perturbations from the on-merge lead to the formation of stop-and-go waves. CAVs with a centralized controller are trained via RL to dissipate these waves.

structures such as on-ramps, lane closers, etc. [113, 175]. This results in a problem that is more difficult to solve than the one experienced in closed network geometries, and accordingly highlights the potential benefits of originally generating control strategies in closed network settings and attempting to transfer the knowledge.

7.3 Experimental Setup

In this section, we present an experimental setup for studying the effect of mixed autonomy on open networks via deep RL, and building on recent studies [202, 201], propose similar ring road representations of the problem to assess the transferability of control strategies generated for closed networks.

7.3.1 Problem setup

This chapter is concerned with the problem of mixed-autonomy traffic stabilization: specifically, how a small percentage of automated vehicles stabilize stop-and-go traffic in congested highways networks.

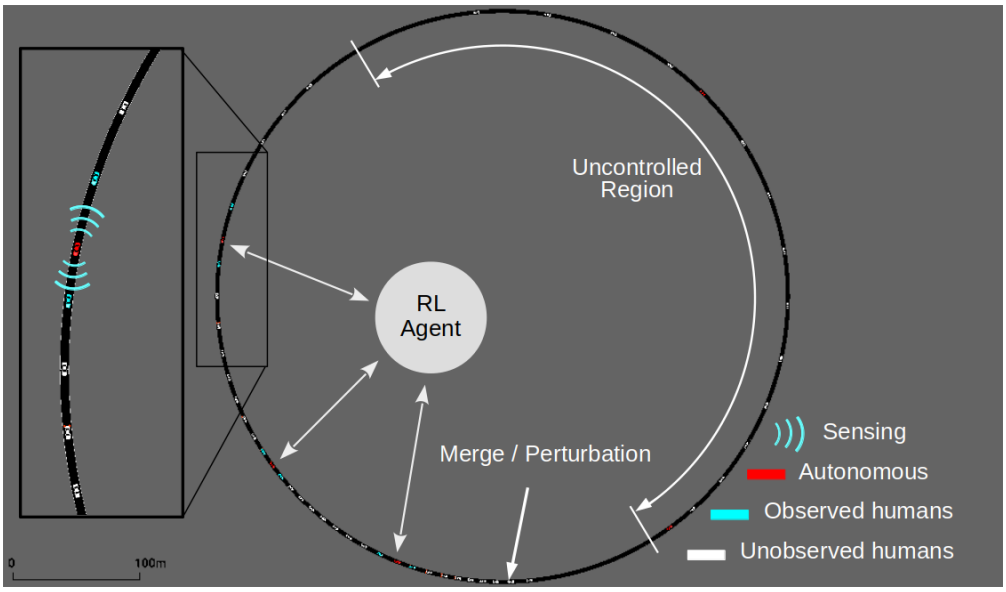


Figure 7.2: Closed network highway of length 1400 m with 50 vehicles and a 700 m controlled region. Periodic perturbations are induced to vehicles near a fixed point, mimicking the effects of an on-merge. A centralized controller issues commands to CAVs only within a controllable region.

Network configuration

A scenario is proposed to study the effects of automated vehicle in open networks that exhibit stop-and-go behavior (see Figure 7.1). This scenario consists of a single-lane highway network with an on-ramp used to generate periodic perturbations to sustain congested behavior. The scenario is characterized by a variable highway length L_h in which the system dynamics are influenced by the presence of automated vehicles, as well as highway and merge inflow rates, denoted by q_h and q_m respectively. For the purpose of this study, the following network parameters are used: $L_h = 700$ m, $q_h = 2000$ veh/hr, $q_m = 100$ veh/hr. Note that the on-ramp inflow rate is much smaller than the primary highway inflow rate, and is designed to be a fixed source of perturbations rather than a realistic on-ramp inflow.

Human-driven vehicles

The longitudinal dynamics of human-driven vehicles in the network are provided by the *Intelligent Driver Model* (IDM) [174], a microscopic car-following model in which the accelerations of a vehicle α are defined by its bumper-to-bumper headway h_α, velocity v_α, and relative velocity with the preceding vehicle $\Delta v_\alpha = v_l - v_\alpha$, via the following equation:

$$f(h_\alpha, v_l, v_\alpha) = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{h_\alpha}\right)^2\right] \tag{7.4}$$

where s^* is the desired headway of the vehicle, denoted by:

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + \max\left(0, v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}\right) \tag{7.5}$$

where s_0, v_0, T, δ, a, b are given parameters calibrated to model highway traffic [178]. In order to simulate stochasticity in driver behavior, exogenous Gaussian noise of $\mathcal{N}(0, 0.2)$ is added to the accelerations, calibrated to match findings in [176]. Finally, right-of-way dynamics near the merge are specified by the simulator for both human and autonomous vehicles.

Automated vehicles

In order to model the effect of $p\%$ CAV penetration on the network, every $\frac{100}{p}$th vehicle is replaced with an automated vehicle whose actions are sampled from a centralized (single-agent) RL policy.

Observations and actions

The observation space of the learning agent consists of locally observable network features. This includes the speeds $v_{i,\text{lead}}$, $v_{i,\text{lag}}$ and bumper-to-bumper headways $h_{i,\text{lag}}$, $h_{i,\text{lag}}$ of the vehicles immediately preceding and following the automated vehicles, as well as the ego speed v_i of automated vehicle i.

The action space consists of a vector of bounded accelerations a_i for each automated vehicle i. In order to ensure safety, these actions are further bounded by failsafes provided by the simulator at every time step.

In an open network, the number of vehicles, and accordingly the number of automated vehicles, fluctuates as vehicles enter and exit the network; however, the RL agent continuously issues a list of actions of fixed size n and requests a list of observations of fixed size m. In order to consolidate potential mismatches between the size of the state/action spaces and the number of AVs, we use zero padding.

Reward function

We choose a reward function that promotes high system-level speeds. Let $v_i(t)$ and $h_i(t)$ be the speed and time headway of vehicle i at time step t, respectively. The reward function is defined as follows:

$$r = ||v_{\text{des}}|| - ||v_{\text{des}} - v(t)|| - \alpha \sum_{i \in AV} \max\left[h_{\max} - h_i(t), 0\right] \tag{7.6}$$

The first two terms encourage proximity of the system-level velocity to a desired speed v_{des} while maintaining a positive reward value to penalize prematurely terminated simulation rollouts caused by vehicle collisions. The second term, on the other hand, is a penalty used to

identify local features of congested traffic (namely small time headways). In order to ensure that this term does not affect the global optimum, the penalty is ignored when time headways are smaller than a threshold value $h_{\max}$, and a gain α is used to diminish the magnitude of the penalty. For this problem, the following constants are chosen: $v_{\text{des}} = 25$ m/s, $h_{\max} = 1$ s, $\alpha = 0.1$.

7.3.2 Transfer learning from closed network policies

We wish to understand through deep reinforcement learning whether control strategies developed in a ring can be transferred and fined-tuned to improve traffic in realistic open network settings. In order to do so, a ring road setup similar to the straight highway depicted in Section 7.3.1 is designed (see Figure 7.2). The ring has a circumference of 1400 m and a total of 50 vehicles, approximately matching the densities in straight highway simulations. In order to reconstruct the effects of the on-merge, vehicles closest to an arbitrary fixed point are periodically perturbed with a frequency equal to that on the merge inflow rate F_m. Finally, in order to account for variability in the number of AVs, observation and action data in only acquired from and provided to automated vehicles within a controllable region of length. In all other regions of space, the AVs act as human-driven vehicles.

During the RL training process, automated vehicles are initially trained in the ring road with the same actions, observations, and rewards described. Then after a predefined number of iterations, the network is replaced with the previously described straight highway network, and training is continued.

7.3.3 Simulations

Experiments are implemented in Flow, an open-source computational framework for running deep reinforcement learning experiments in traffic microscopic simulators [202]. Flow enables the systematic creation of a variety of traffic-oriented RL tasks for the purpose of generating control strategies for autonomous vehicles, traffic lights, etc. All results presented in this chapter are reproducible from the Flow repository at: `https://github.com/flow-project/flow`.

Simulations are executed in the state-of-the-art traffic micro-simulator SUMO [80] with simulation time steps of 0.2 s and a total duration of 3600 s. The RL agent is provided updated state information and generates new actions in increments of 1 s, with the actions repeated for the next five consecutive simulation steps.

For all experiments in this chapter, we use the *Trust Region Policy Optimization* (TRPO) [147] policy gradient method for learning the control policy, linear feature baselines as described in [44], discount factor $\gamma = 0.999$, and step size 0.01. For most experiments, a diagonal Gaussian MLP policy is used with hidden layers (32, 32, 32) and a `tanh` non-linearity.

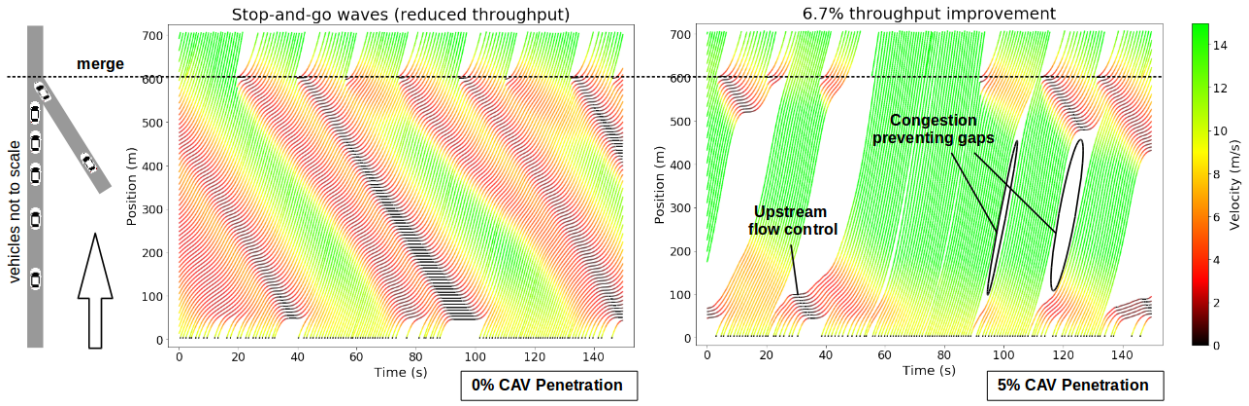


Figure 7.3: **Left**: In the absence of autonomy, waves form and propagate through the network without any regulation. **Right**: In the presence of mixed autonomy, the automated vehicles learn to temporary reduce inflows by slowing down near the start of the network in order to prematurely terminate propagating waves. In addition, the automated vehicles at times create safety gaps between themselves and the downstream wave.

7.4 Numerical Results

We illustrate the results for CAV penetration rates ranging from 0% to 10%. RL training runs for the various experimental setups were executed over five seeds, with training performance presented over all seeds. Moreover, excluding Figure 7.6, all reported performance values are averaged over 10 simulations in order to account for stochasticity between simulations. Videos of the results are available at: `https://sites.google.com/view/itsc-dissipating-waves`.

7.4.1 Performance benefits of mixed autonomy traffic

Figure 7.6 presents the effect of different CAV penetration rates on key congestion performance factors. In terms of mobility, we witness a 13% increase in throughput as the portion of automated vehicles in the network increases from 0% to 10%, with vehicles on average moving at almost twice the speed. Moreover, in terms of energy efficiency, we see that stop-and-go waves within the network as virtually eliminated at penetration rates of 10%, with smaller penetration rates also resulting in less frequent and smaller waves in the network.

7.4.2 Spatio-temporal dynamics of automated vehicles

Figure 7.3, as well as the videos mentioned at the start of this section, provide a spatio-temporal representation of the effect of autonomy on the dynamics of the network. In the absence of automated vehicles, perturbations induced by the on-merge periodically result

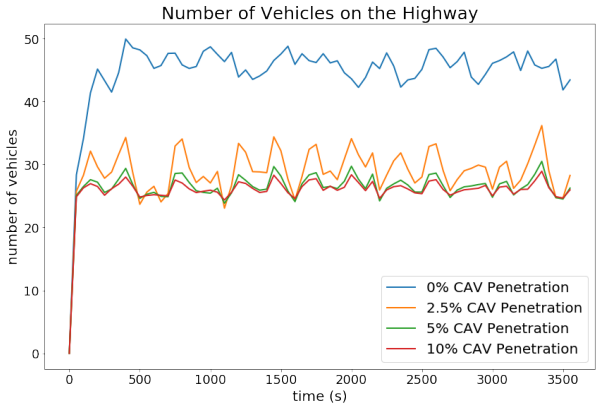


Figure 7.4: The automated vehicle acts as a ramp meter in the controllable region of traffic, effectively maintaining the density at approximately 0.36 veh/m. This strategy is less effective under smaller penetration rates.

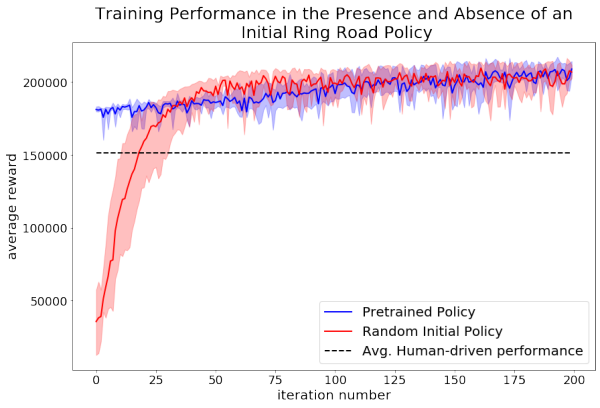


Figure 7.5: RL training performance for a policy trained from a random initial state (red) and a ring road policy (blue). The policy trained on the ring road outperforms human-driven dynamics on the highway.

in the formation of traffic destabilizing stop-and-go waves. This congestive behavior has a significant effect on the number of vehicles passing through the network, dropping the throughput from a free-flow value of 2000 veh/hr to an observed value of 1604 veh/hr.

In the presence of mixed autonomy, automated vehicles near the left of the network slow down or stop in the event of a formation of a wave near the merge, thereby temporarily blocking off traffic from entering the network and contributing to the propagation of the downstream jam. For larger CAV penetration rates, this behavior occurs further downstream, with the automated vehicles slowing down in unison for shorter periods of time, resulting in fewer and shorter lived waves in the network. Finally, once the upstream jam is partially cleared, the automated vehicles then resume regular safe car following behavior.

The learned policy for the automated vehicles share many similarities with ramp metering. As can be seen in Figure 7.4, the automated vehicles succeed in regulating the density of traffic in the straight highway below its critical value. This control strategy is more stable at high CAV penetration rates.

Remarkably, the ramp metering and flow synchronization behaviors discussed in previous paragraphs emerge in the absence of knowledge on the location of the merge or any existing stop-and-go waves. Instead, it is likely that the centralized/single-agent structure of the problem allows vehicles to coordinate actions whenever a lead vehicle acquires jam-like observations such as small headways for heavy fluctuations in speed. This demonstrates the potential of V2V communication in mitigating traffic congestion.

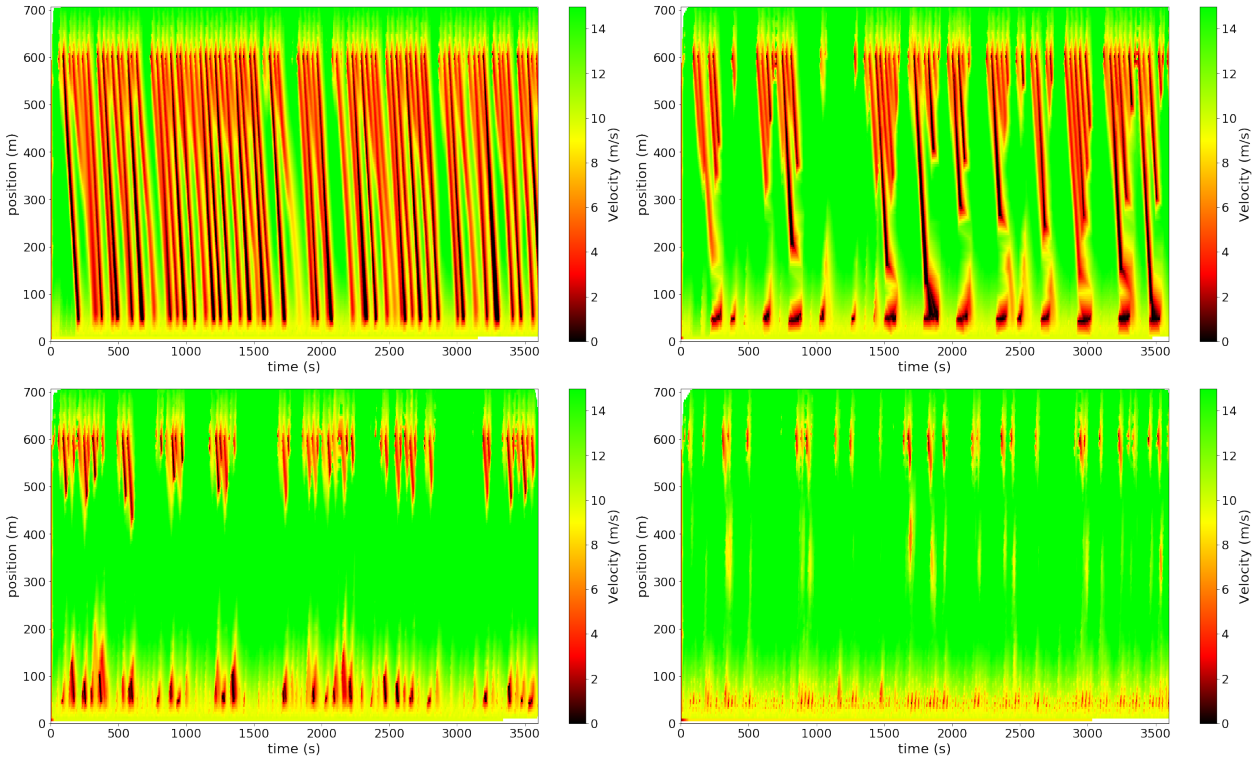


Figure 7.6: Spatio-temporal representation of vehicles dynamics within a periodically perturbed highway. In the absence of automated vehicles, the network exhibits properties of convective instability, with perturbations propagating upstream from the merge point before exiting the network. As the percentage of autonomous penetration increases, the waves are increasingly dissipated, with virtually no waves propagating from the merge at 10% autonomy. **Top left**: 0% CAV penetration, **Top Right**: 2.5% CAV penetration, **Bottom left**; 5% CAV penetration, **Bottom right**: 10% CAV penetration.

7.4.3 Transfer learning performance

Figure 7.5 presents the training performance of the RL agent for an CAV penetration rate of 10%. Comparing the transfer learning method mentioned in Section 7.3.2 against purely training the RL agent in the straight highway, we find that the policy learned on the ring road initially outperforms human-driven dynamics in the straight highway network, thereby acting as a “warm start" to the RL training process, which then continues to optimize the controller parameters for the straight highway. Notably, during the fine-tuning stage, we do not see a sharp drop in training performance, which would indicate incompatibility of the ring road policy for the straight road network. This suggests that the MDP structures presented in closed and open networks are sufficiently similar for control strategies developed to be somewhat interchangeable.

7.5 Chapter Summary

This chapter presents a deep RL approach to generating stop-and-go wave regulating controllers in realistic network geometries. This method is demonstrated to achieve near complete wave dissipation with only 10% autonomous penetration. In addition, penetration rates as low as 2.5% are revealed to contribute greatly to reductions in the frequency and magnitude of formed waves. Finally, a study of controllers generated in closed network scenarios exhibiting otherwise similar densities and perturbing behaviors confirms the transferability of closed network policies to open network tasks, and presents the potential role of transfer reinforcement learning in fine-tuning the parameters of these policies. In future work, we hope to utilize this interchangeability to solve much larger and numerically more difficult highway network tasks from primitives learned on ring roads and single lane merges.

Part IV

Final Remarks

CHAPTER 8

Conclusion and Future Prospects

Reinforcement learning strategies hold the promise of redefining a number of our approaches to solving sequential decision-making problems. The adoption of data-driven methods to task solving, coupled with the highly expressive nature of deep neural networks, allows solutions to such problems to no longer be restricted by human ingenuity and model-based formulations designed with simplicity and interpretability in mind. Instead, modern search and optimization frameworks can help guide our understanding of the intricacies of certain problems, and often succeed in introducing new and novel ways of solving them in a manner that both augments and enhances our understanding of the original problem to begin with. This is true for two-player games such as Chess [26] and Go [151, 152] as well as biological applications such as protein folding [180], and will likely continue to permeate to other fields as well. However, much research must still be conducted in attempting adopt these techniques to each new classes of problems in order to develop a thorough understanding of when, where, and why they may fail to produce meaningful results.

In this document, we explore methods for adapting modern reinforcement learning and machine learning techniques to the problem of mixed-autonomy traffic, whereby a subset of vehicles act as automated entities and subsequently attempt to regulate the flow of traffic through their actions. As part of this research, we present Flow, a flexible learning framework designed to enable the composition of different traffic control problems for analysis in the context of reinforcement learning. We demonstrate through their framework that learning-based solutions for mixed-autonomy traffic are viable, and in fact succeed in producing interesting and generalizable results on a number of problems. However, similar algorithms often fail when applied to larger scale tasks. In this document, we focus on challenges associated with long-term reasoning as well as attempting to learning through computational expensive or inaccurate models. To these challenges we explore the use of hierarchies (Chapter 4) for enabling structured exploration through sparsely defined reward signals, and demonstrate the learning may be made more efficient by coupling supervised learning techniques with expertly defined traffic regulation strategies (Chapter 5 and 6) and well as learning in simplified representations of a target network (Chapter 7).

The work presented here takes some steps towards developing reinforcement learning

strategies that can enhance our understanding of viable traffic control solutions and produce feedback control mechanisms that may be implemented in field experiments. However, a number of challenges must still be addressed to develop controllers that consistently and with a high degree of confidence act as would be desired in generic real-world settings. These challenges, some of which we list below, highlight a number of exciting directions for future exploration.

Compositional models. Our work on the adoption of hierarchies for enabling long-term learning focuses primarily of end-to-end learning, whereby policy layers are initialized randomly and must coordinate with one another to generate meaningful results. However, the usefulness of hierarchies is not restricted to end-to-end learning. Instead, through such models, multiple learned lower-level skills may be coupled and sampled at temporally extended intervals to solve different traffic-oriented problems as they emerge. For instance, policies trained to coordinate vehicles through an on-ramp merge [82] versus a lane-reduction bottleneck [185] may be combined through an options-style framework [162] and chosen based on nearby network structure or other learned probabilities [9]. In this way, policies learned in simplified representations of traffic may potentially be effectively integrated into a single unified model for traffic regulation. Much research, however, must still be conducted on identifying the learned priors needed to solve every variation of traffic control problem, as well as on identifying methods for smoothly integrating such lower-level skills with one another and defining the conditions under which each should be utilized.

Data collection and simulation. In this document, we explore applications of reinforcement learning to traffic regulation problems simulated via microscopic traffic flow models. Such models historically have enabled a number of technological advancements within the field of intelligent transportation systems, but were not intended to capture the full range and diversity of behaviors symbolic of human-style driving. These unreplicated behaviors, however, may be beneficial within the context of RL-based applications, as through the subtleties of human interactions interesting learned behaviors may emerge. As such, many advancements in machine learning applications for traffic control may be further enabled by developing more intricate models of human driving. To this problem, data-driven solutions may once again play an important role, and are commonly explored, primarily in the context of imitation learning. However, as has been in the case in other fields, additional data on human driving behaviors are required. In particular, data on human driving responses are needed from dense traffic settings whereby suboptimalities in human behaviors may be addressed by automated vehicles in mixed traffic settings.

Sim-to-real considerations. Finally, we note that simulation-centric methods to learning will always suffer from what is known as a "sim-to-real gap", whereby differences between the simulated and physical world invalidate certain aspects of the learned response. As such, much research must still be conducted in reducing the gap between the two (as described above)

as well as generating robust learning methods for enabling learned responses to generalize to variations between both settings. Simple injections of exogenous noise to both observed states and executed actions are one such solution [71], however more in depth adaptations of adversarial learning [131] or domain randomization [172] suited for the problem at hand are more likely needed.

⋆ ⋆ ⋆

We hope the work presented in this thesis provides useful building blocks for future research revolving vehicle autonomy and role that automated vehicles and machine learning systems can and should play in the future of transportation and mobility.

Bibliography

[1] ""Phantom auto" to Be Operated Here". In: *The Free Lance-Star* (June 1932).

[2] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. "Reinforcement learning for true adaptive traffic signal control". In: *Journal of Transportation Engineering* 129.3 (2003), pp. 278–285.

[3] Saleh Albeaik et al. "Limitations and improvements of the intelligent driver model (idm)". In: *SIAM Journal on Applied Dynamical Systems* 21.3 (2022), pp. 1862–1892.

[4] Florent Altché and Arnaud de La Fortelle. "An LSTM network for highway trajectory prediction". In: *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE. 2017, pp. 353–359.

[5] Marcin Andrychowicz et al. "Hindsight experience replay". In: *Advances in neural information processing systems* 30 (2017).

[6] Marcin Andrychowicz et al. "What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study". In: *ICLR 2021-Ninth International Conference on Learning Representations*. 2021.

[7] Behrang Asadi, Chen Zhang, and Ardalan Vahidi. "The role of traffic flow preview for planning fuel optimal vehicle velocity". In: *Dynamic systems and control conference*. Vol. 44182. 2010, pp. 813–819.

[8] AATM Aw and Michel Rascle. "Resurrection of" second order" models of traffic flow". In: *SIAM journal on applied mathematics* 60.3 (2000), pp. 916–938.

[9] Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The option-critic architecture". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 2017.

[10] Masako Bando et al. "Dynamical model of traffic congestion and numerical simulation". In: *Physical review E* 51.2 (1995), p. 1035.

[11] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst". In: *arXiv preprint arXiv:1812.03079* (2018).

[12] William Barbour et al. "Interstate 24 MOTION open road testbed". In: ().

[13] Andrew G Barto and Sridhar Mahadevan. "Recent advances in hierarchical reinforcement learning". In: *Discrete event dynamic systems* 13.1 (2003), pp. 41–77.

[14] Norman Bel Geddes. *Magic motorways*. New York: Random house, 1940.

[15] Marc Bellemare et al. "Unifying count-based exploration and intrinsic motivation". In: *Advances in neural information processing systems* 29 (2016).

[16] Francois Belletti et al. "Expert Level Control of Ramp Metering Based on Multi-Task Deep Reinforcement Learning". In: *IEEE Transactions on Intelligent Transportation Systems* 19 (2018), pp. 1198–1207.

[17] Richard Bellman. "A Markovian decision process". In: *Journal of mathematics and mechanics* 6.5 (1957), pp. 679–684.

[18] Moshe E Ben-Akiva, Steven R Lerman, Steven R Lerman, et al. *Discrete choice analysis: theory and application to travel demand*. Vol. 9. 1985.

[19] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[20] Myra Blanco et al. *Automated vehicle crash rate comparison using naturalistic data*. Tech. rep. Virginia Tech Transportation Institute, 2016.

[21] Mariusz Bojarski et al. "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316* (2016).

[22] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[23] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The 2005 DARPA grand challenge: the great robot race*. Vol. 36. Springer, 2007.

[24] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*. Vol. 56. springer, 2009.

[25] Lucian Busoniu, Robert Babuska, and Bart De Schutter. "Multi-agent reinforcement learning: A survey". In: *2006 9th International Conference on Control, Automation, Robotics and Vision*. IEEE. 2006, pp. 1–6.

[26] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. "Deep blue". In: *Artificial intelligence* 134.1-2 (2002), pp. 57–83.

[27] Jordi Casas et al. "Traffic simulation with aimsun". In: *Fundamentals of traffic simulation*. Springer, 2010, pp. 173–232.

[28] Chao Chen et al. "Freeway performance measurement system: mining loop detector data". In: *Transportation Research Record* 1748.1 (2001), pp. 96–102.

[29] Zhiyuan Chen and Bing Liu. "Lifelong machine learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12.3 (2018), pp. 1–207.

[30] Fang-Chieh Chou, Alben Rome Bagabaldo, and Alexandre M Bayen. "The lord of the ring road: a review and evaluation of autonomous control policies for traffic in a ring road". In: *ACM Transactions on Cyber-Physical Systems (TCPS)* 6.1 (2022), pp. 1–25.

[31] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[32] Koohong Chung, Jittichai Rudjanakanoknad, and Michael J Cassidy. "Relation between traffic density and capacity drop at three freeway bottlenecks". In: *Transportation Research Part B: Methodological* 41.1 (2007), pp. 82–95.

[33] Joan Claybrook and Shaun Kildare. "Autonomous vehicles: No driver... no regulation?" In: *Science* 361.6397 (2018), pp. 36–37.

[34] Felipe Codevilla et al. "End-to-end driving via conditional imitation learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 4693–4700.

[35] Graham Cookson and Bob Pishue. "Inrix global traffic scorecard–appendices". In: *INRIX research* (2017).

[36] Shumo Cui et al. "Stabilizing traffic flow via a single autonomous vehicle: Possibilities and limitations". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1336–1341.

[37] Jifeng Dai et al. "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems* 29 (2016).

[38] Peter Dayan and Geoffrey E Hinton. "Feudal reinforcement learning". In: *Advances in neural information processing systems*. 1993, pp. 271–278.

[39] Marc Deisenroth and Carl E Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer. 2011, pp. 465–472.

[40] Charles Desjardins and Brahim Chaib-Draa. "Cooperative adaptive cruise control: A reinforcement learning approach". In: *IEEE Transactions on intelligent transportation systems* 12.4 (2011), pp. 1248–1260.

[41] Ernst Dieter Dickmanns and Volker Graefe. "Dynamic monocular machine vision". In: *Machine vision and applications* 1.4 (1988), pp. 223–240.

[42] F Dion et al. "Connected corridors: I-210 pilot integrated corridor management system concept of operations". In: *California PATH, Berkeley, CA* (2015).

[43] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

[44] Yan Duan et al. "Benchmarking deep reinforcement learning for continuous control". In: *International Conference on Machine Learning*. 2016, pp. 1329–1338.

[45] Tom Eccles et al. "Biases for Emergent Communication in Multi-agent Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13111–13121.

[46] Martin Fellendorf and Peter Vortisch. "Microscopic traffic flow simulator VISSIM". In: *Fundamentals of traffic simulation*. Springer, 2010, pp. 63–93.

[47] Carlos Florensa, Yan Duan, and Pieter Abbeel. "Stochastic Neural Networks for Hierarchical Reinforcement Learning". In: *arxiv preprint arxiv:1704.03012* (2017).

[48] Jakob Foerster et al. "Bayesian action decoder for deep multi-agent reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1942–1951.

[49] Jakob Foerster et al. "Counterfactual multi-agent policy gradients". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 2018.

[50] Jakob Foerster et al. "Stabilising experience replay for deep multi-agent reinforcement learning". In: *arXiv preprint arXiv:1702.08887* (2017).

[51] Jakob N. Foerster et al. "Learning to Communicate with Deep Multi-Agent Reinforcement Learning". In: *CoRR* abs/1605.06676 (2016). arXiv: 1605.06676. URL: http://arxiv.org/abs/1605.06676.

[52] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *arXiv preprint arXiv:1802.09477* (2018).

[53] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[54] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[55] Gabriel Gomes, Adolf May, and Roberto Horowitz. "Congested freeway microsimulation model using VISSIM". In: *Transportation Research Record* 1876.1 (2004), pp. 71–81.

[56] Sorin Grigorescu et al. "A survey of deep learning techniques for autonomous driving". In: *Journal of Field Robotics* 37.3 (2020), pp. 362–386.

[57] Jacopo Guanetti, Yeojun Kim, and Francesco Borrelli. "Control of connected and automated vehicles: State of the art and future challenges". In: *Annual reviews in control* 45 (2018), pp. 18–40.

[58] George Gunter et al. "Are commercially implemented adaptive cruise control systems string stable?" In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[59] Nico Gürtler, Dieter Büchler, and Georg Martius. "Hierarchical Reinforcement Learning with Timed Subgoals". In: *Advances in Neural Information Processing Systems* 34 (2021).

[60] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *arXiv preprint arXiv:1801.01290* (2018).

[61] Patrick Hart, Leonard Rychly, and Alois Knoll. "Lane-merging using policy-based reinforcement learning and post-optimization". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 3176–3181.

[62] Jeffrey Hawke et al. "Urban driving with conditional imitation learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 251–257.

[63] Simon Haykin. "A comprehensive foundation". In: ().

[64] David Held et al. "Automatic goal generation for reinforcement learning agents". In: *arXiv preprint arXiv:1705.06366* (2017).

[65] Dwight A Hennessy and David L Wiesenthal. "Traffic congestion, driver stress, and driver aggression". In: *Aggressive Behavior: Official Journal of the International Society for Research on Aggression* 25.6 (1999), pp. 409–423.

[66] Robert Herman et al. "Traffic dynamics: analysis of stability in car following". In: *Operations research* 7.1 (1959), pp. 86–106.

[67] Julia Hirschberg and Christopher D Manning. "Advances in natural language processing". In: *Science* 349.6245 (2015), pp. 261–266.

[68] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[69] Carl-Johan Hoel, Krister Wolff, and Leo Laine. "Automated speed and lane change decision making using deep reinforcement learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2148–2155.

[70] Zhenhua Huang et al. "Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.4 (2017), pp. 730–741.

[71] Kathy Jang et al. "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles". In: *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. 2019, pp. 291–300.

[72] Michael Janner et al. "When to trust your model: Model-based policy optimization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[73] Natasha Jaques et al. "Social influence as intrinsic motivation for multi-agent deep reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3040–3049.

[74] Rui Jiang, Qingsong Wu, and Zuojin Zhu. "Full velocity difference model for a car-following theory". In: *Physical Review E* 64.1 (2001), p. 017101.

[75] Hossein Jula, Elias B Kosmatopoulos, and Petros A Ioannou. "Collision avoidance analysis for lane changing and merging". In: *IEEE Transactions on vehicular technology* 49.6 (2000), pp. 2295–2308.

[76] Alex Kendall et al. "Learning to drive in a day". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8248–8254.

[77] Arne Kesting, Martin Treiber, and Dirk Helbing. "General lane-changing model MOBIL for car-following models". In: *Transportation Research Record* 1999.1 (2007), pp. 86–94.

[78] Woojun Kim, Jongeui Park, and Youngchul Sung. "Communication in multi-agent reinforcement learning: Intention sharing". In: *International Conference on Learning Representations*. 2020.

[79] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[80] Daniel Krajzewicz et al. "Recent development and applications of SUMO-Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3&4 (2012).

[81] Abdul Rahman Kreidieh, Zhe Fu, and Alexandre M Bayen. "Learning energy-efficient driving behaviors by imitating experts". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, pp. 2689–2695.

[82] Abdul Rahman Kreidieh, Cathy Wu, and Alexandre M Bayen. "Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1475–1480.

[83] Abdul Rahman Kreidieh et al. "Inter-level cooperation in hierarchical reinforcement learning". In: *arXiv preprint arXiv:1912.02368* (2019).

[84] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[85] Fabian Kröger. "Automated driving in its social, historical and cultural contexts". In: *Autonomous Driving*. Springer, 2016, pp. 41–68.

[86] Tejas D Kulkarni et al. "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation". In: *Advances in neural information processing systems*. 2016, pp. 3675–3683.

[87] Siddharth Krishna Kumar. "On weight initialization in deep neural networks". In: *arXiv preprint arXiv:1704.08863* (2017).

[88] Jorge A Laval and Carlos F Daganzo. "Lane-changing in traffic streams". In: *Transportation Research Part B: Methodological* 40.3 (2006), pp. 251–264.

[89] Ludovic Leclercq, Jorge A Laval, and Nicolas Chiabaut. "Capacity drops at merges: An endogenous model". In: *Procedia-Social and Behavioral Sciences* 17 (2011), pp. 12–26.

[90] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[91] Donghan Lee et al. "Convolution neural network-based lane change intention prediction of surrounding vehicles for ACC". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–6.

[92] Jonathan W Lee et al. "Integrated Framework of Vehicle Dynamics, Instabilities, Energy Models, and Sparse Flow Smoothing Controllers". In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. 2021, pp. 41–47.

[93] Sergey Levine and Vladlen Koltun. "Guided policy search". In: *International conference on machine learning*. PMLR. 2013, pp. 1–9.

[94] Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[95] Andrew Levy, Robert Platt, and Kate Saenko. "Hierarchical actor-critic". In: *arXiv preprint arXiv:1712.00948* (2017).

[96] Li Li, Yisheng Lv, and Fei-Yue Wang. "Traffic signal timing via deep reinforcement learning". In: *IEEE/CAA Journal of Automatica Sinica* 3.3 (2016), pp. 247–254.

[97] Shengbo Eben Li et al. "Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities". In: *IEEE Intelligent Transportation Systems Magazine* 9.3 (2017), pp. 46–58.

[98] Zhibin Li et al. "Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks". In: *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), pp. 3204–3217.

[99] Eric Liang et al. "Ray RLLib: A Composable and Scalable Reinforcement Learning Library". In: *arXiv preprint arXiv:1712.09381* (2017).

[100] Nathan Lichtlé et al. "Deploying Traffic Smoothing Cruise Controllers Learned from Trajectory Data". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2884–2890.

[101] Nathan Lichtlé et al. "Fuel Consumption Reduction of Multi-Lane Road Networks using Decentralized Mixed-Autonomy Control". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 2068–2073.

[102] Michael James Lighthill and Gerald Beresford Whitham. "On kinematic waves II. A theory of traffic flow on long crowded roads". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178 (1955), pp. 317–345.

[103] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[104] Minghuan Liu, Menghui Zhu, and Weinan Zhang. "Goal-Conditioned Reinforcement Learning: Problems and Solutions". In: *arXiv preprint arXiv:2201.08299* (2022).

[105] Ryan Lowe et al. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6379–6390.

[106] Xiao-Yun Lu, J Karl Hedrick, and Michael Drew. "ACC/CACC-control design, stability and robust performance". In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*. Vol. 6. IEEE. 2002, pp. 4327–4332.

[107] Horia Mania, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning". In: *arXiv preprint arXiv:1803.07055* (2018).

[108] Roland Marchand. "The Designers Go to the Fair II: Norman Bel Geddes, The General Motors" Futurama," and the Visit to the Factory Transformed". In: *Design Issues* 8.2 (1992), pp. 23–40.

[109] Paula Marchesini and Wilhelmina Adriana Maria Weijermars. *The relationship between road safety and congestion on motorways*. SWOV Institute for Road Safety Research Leidschendam, Netherlands, 2010.

[110] Vicente Milanés and Steven E Shladover. "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data". In: *Transportation Research Part C: Emerging Technologies* 48 (2014), pp. 285–300.

[111] Vicente Milanés et al. "Cooperative adaptive cruise control in real traffic situations". In: *IEEE Transactions on intelligent transportation systems* 15.1 (2013), pp. 296–305.

[112] Branka Mirchevska et al. "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. 2156–2162.

[113] Namiko Mitarai and Hiizu Nakanishi. "Convective instability and structure formation in traffic flow". In: *Journal of the Physical Society of Japan* 69.11 (2000), pp. 3752–3761.

[114] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), p. 529.

[115] Igor Mordatch and Pieter Abbeel. "Emergence of grounded compositional language in multi-agent populations". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[116] Nassim Motamedidehkordi, Martin Margreiter, and Thomas Benz. "Shockwave suppression by vehicle-to-vehicle communication". In: *Transportation research procedia* 15 (2016), pp. 471–482.

[117] Ofir Nachum et al. "Data-efficient hierarchical reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3303–3313.

[118] Ofir Nachum et al. "Near-Optimal Representation Learning for Hierarchical Reinforcement Learning". In: *CoRR* abs/1810.01257 (2018). arXiv: 1810.01257. URL: http://arxiv.org/abs/1810.01257.

[119] Ofir Nachum et al. "Near-optimal representation learning for hierarchical reinforcement learning". In: *arXiv preprint arXiv:1810.01257* (2018).

[120] Soroush Nasiriany et al. "Planning with goal-conditioned policies". In: *arXiv preprint arXiv:1911.08453* (2019).

[121] Gordon Frank Newell. "A simplified car-following theory: a lower order model". In: *Transportation Research Part B: Methodological* 36.3 (2002), pp. 195–205.

[122] Gábor Orosz and Gábor Stépán. "Subcritical Hopf bifurcations in a car-following model with reaction-time delay". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 462.2073 (2006), pp. 2643–2670.

[123] Gábor Orosz, R Eddie Wilson, and Gábor Stépán. *Traffic jams: dynamics and control*. 2010.

[124] Gábor Orosz et al. "Exciting traffic jams: Nonlinear phenomena behind traffic jam formation on highways". In: *Physical review E* 80.4 (2009), p. 046205.

[125] Brian Paden et al. "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE Transactions on intelligent vehicles* 1.1 (2016), pp. 33–55.

[126] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[127] Markos Papageorgiou, Habib Hadj-Salem, Jean-Marc Blosseville, et al. "ALINEA: A local feedback control law for on-ramp metering". In: *Transportation Research Record* 1320.1 (1991), pp. 58–67.

[128] Markos Papageorgiou and Apostolos Kotsialos. "Freeway ramp metering: An overview". In: *IEEE transactions on intelligent transportation systems* 3.4 (2002), pp. 271–281.

[129] Harold J Payne. "Model of freeway traffic and control". In: *Mathematical Model of Public System* (1971), pp. 51–61.

[130] Xue Bin Peng et al. "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning". In: *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* 36.4 (2017).

[131] Lerrel Pinto et al. "Robust adversarial reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2817–2826.

[132] Dean A Pomerleau. *Alvinn: An autonomous land vehicle in a neural network*. Tech. rep. Carnegie-Mellon University, 1989.

[133] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[134] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[135] Paul I Richards. "Shock waves on the highway". In: *Operations research* 4.1 (1956), pp. 42–51.

[136] Stéphane Ross and Drew Bagnell. "Efficient reductions for imitation learning". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 661–668.

[137] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[138] Meead Saberi and Hani S Mahmassani. "Empirical characterization and interpretation of hysteresis and capacity drop phenomena in freeway networks". In: *Transportation Research Record: Journal of the Transportation Research Board, Transportation Research Board of the National Academies, Washington, DC* (2013).

[139] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[140] Tim Salzmann et al. "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data". In: *European Conference on Computer Vision*. Springer. 2020, pp. 683–700.

[141] Wouter J Schakel, Victor L Knoop, and Bart van Arem. "Integrated lane change model with relaxation and synchronization". In: *Transportation Research Record* 2316.1 (2012), pp. 47–57.

[142] Tom Schaul et al. "Universal value function approximators". In: *International conference on machine learning*. 2015, pp. 1312–1320.

[143] Oliver Scheel et al. "Attention-based lane change prediction". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8655–8661.

[144] Thorsten Schmidt-Dumont and Jan H van Vuuren. "Decentralised reinforcement learning for ramp metering and variable speed limits on highways". In: *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS* 14.8 (2015), p. 1.

[145] Julian Schrittwieser et al. "Mastering atari, go, chess and shogi by planning with a learned model". In: *Nature* 588.7839 (2020), pp. 604–609.

[146] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[147] John Schulman et al. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[148] Toru Seo et al. "Traffic state estimation on highway: A comprehensive survey". In: *Annual reviews in control* 43 (2017), pp. 128–151.

[149] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. "Safe, multi-agent, reinforcement learning for autonomous driving". In: *arXiv preprint arXiv:1610.03295* (2016).

[150] David Silver et al. "Deterministic policy gradient algorithms". In: *ICML*. 2014.

[151] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[152] David Silver et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[153] Silvia Siri et al. "Freeway traffic control: A survey". In: *Automatica* 130 (2021), p. 109655.

[154] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[155] Raphael E Stern et al. "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments". In: *Transportation Research Part C: Emerging Technologies* 89 (2018), pp. 205–221.

[156] Yuki Sugiyama et al. "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam". In: *New journal of physics* 10.3 (2008), p. 033001.

[157] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. "Learning Multiagent Communication with Backpropagation". In: *CoRR* abs/1605.07736 (2016). arXiv: `1605.07736`. URL: `http://arxiv.org/abs/1605.07736`.

[158] Sainbayar Sukhbaatar et al. "Learning goal embeddings via self-play for hierarchical reinforcement learning". In: *arXiv preprint arXiv:1811.09083* (2018).

[159] Liting Sun et al. "A fast integrated planning and control framework for autonomous driving via imitation learning". In: *Dynamic Systems and Control Conference*. Vol. 51913. American Society of Mechanical Engineers. 2018, V003T37A012.

[160] Richard S Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM Sigart Bulletin* 2.4 (1991), pp. 160–163.

[161] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[162] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[163] Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems*. 2000, pp. 1057–1063.

[164] Darbha Swaroop and J Karl Hedrick. "String stability of interconnected systems". In: *IEEE transactions on automatic control* 41.3 (1996), pp. 349–357.

[165] Eckhard Szimba and Martin Hartmann. "Assessing travel time savings and user benefits of automated driving–A case study for a commuting relation". In: *Transport Policy* 98 (2020), pp. 229–237.

[166] Alireza Talebpour and Hani S Mahmassani. "Influence of connected and autonomous vehicles on traffic flow stability and throughput". In: *Transportation Research Part C: Emerging Technologies* 71 (2016), pp. 143–163.

[167] Alireza Talebpour, Hani S Mahmassani, and Samer H Hamdar. "Modeling lane-changing behavior in a connected environment: A game theory approach". In: *Transportation Research Procedia* 7 (2015), pp. 420–440.

[168] Ardi Tampuu et al. "Multiagent cooperation and competition with deep reinforcement learning". In: *PloS one* 12.4 (2017), e0172395.

[169] Haoran Tang et al. "# exploration: A study of count-based exploration for deep reinforcement learning". In: *Advances in neural information processing systems* 30 (2017).

[170] Matthew E Taylor and Peter Stone. "Transfer learning for reinforcement learning domains: A survey". In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[171] Charles Thorpe et al. "Vision and navigation for the Carnegie-Mellon Navlab". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.3 (1988), pp. 362–373.

[172] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[173] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[174] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations". In: *Physical review E* 62.2 (2000), p. 1805.

[175] Martin Treiber and Arne Kesting. "Evidence of convective instability in congested traffic flow: A systematic empirical and theoretical investigation". In: *Transportation Research Part B: Methodological* 45.9 (2011), pp. 1362–1377.

[176] Martin Treiber and Arne Kesting. "The Intelligent Driver Model with stochasticity–New insights into traffic flow oscillations". In: *Transportation Research Part B: Methodological* (2017).

[177] Martin Treiber and Arne Kesting. "Traffic flow dynamics". In: *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013).

[178] Martin Treiber and Arne Kesting. "Trajectory and Floating-Car Data". In: *Traffic Flow Dynamics*. Springer, 2013, pp. 7–12.

[179] Martin Treiber, Arne Kesting, and Christian Thiemann. "How much does traffic congestion increase fuel consumption and emissions? Applying a fuel consumption model to the NGSIM trajectory data". In: *87th Annual Meeting of the Transportation Research Board, Washington, DC*. 2008, p. 71.

[180] Kathryn Tunyasuvunakool et al. "Highly accurate protein structure prediction for the human proteome". In: *Nature* 596.7873 (2021), pp. 590–596.

[181] Ardalan Vahidi and Antonio Sciarretta. "Energy saving potentials of connected and automated vehicles". In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 822–843.

[182] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[183] Alexander Sasha Vezhnevets et al. "Feudal networks for hierarchical reinforcement learning". In: *arXiv preprint arXiv:1703.01161* (2017).

[184] Eugene Vinitsky et al. "Benchmarks for reinforcement learning in mixed-autonomy traffic". In: *Conference on robot learning*. PMLR. 2018, pp. 399–409.

[185] Eugene Vinitsky et al. "Optimizing Mixed Autonomy Traffic Flow With Decentralized Autonomous Vehicles and Multi-Agent RL". In: *arXiv preprint arXiv:2011.00120* (2020).

[186] Kay W Axhausen, Andreas Horni, and Kai Nagel. *The multi-agent transport simulation MATSim*. Ubiquity Press, 2016.

[187] Zia Wadud, Don MacKenzie, and Paul Leiby. "Help or hindrance? The travel, energy and carbon impacts of highly automated vehicles". In: *Transportation Research Part A: Policy and Practice* 86 (2016), pp. 1–18.

[188] Jane X Wang et al. "Evolving Intrinsic Motivations for Altruistic Behavior". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2019, pp. 683–692.

[189] Jiawei Wang et al. "Leading cruise control in mixed traffic flow: System modeling, controllability, and string stability". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[190] Jun Wang et al. "Safety of autonomous vehicles". In: *Journal of advanced transportation* 2020 (2020).

[191] Meng Wang et al. "Game theoretic approach for predictive lane-changing and car-following control". In: *Transportation Research Part C: Emerging Technologies* 58 (2015), pp. 73–92.

[192] Pin Wang and Ching-Yao Chan. "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–6.

[193] Pin Wang, Ching-Yao Chan, and Arnaud de La Fortelle. "A reinforcement learning based approach for automated lane change maneuvers". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1379–1384.

[194] Jonathan A Ward and R Eddie Wilson. "Criteria for convective versus absolute string instability in car-following models". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 467.2132 (2011), pp. 2185–2208.

[195] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[196] Junqing Wei et al. "A behavioral planning framework for autonomous driving". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE. 2014, pp. 458–464.

[197] Michael Weinberg and Jeffrey S Rosenschein. "Best-response multiagent learning in non-stationary environments". In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. IEEE Computer Society. 2004, pp. 506–513.

[198] MA Wiering. "Multi-agent reinforcement learning for traffic light control". In: *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*. 2000, pp. 1151–1158.

[199] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[200] Bohan Wu, Jayesh K Gupta, and Mykel Kochenderfer. "Model primitives for hierarchical lifelong reinforcement learning". In: *Autonomous Agents and Multi-Agent Systems* 34.1 (2020), pp. 1–38.

[201] Cathy Wu et al. "Emergent behaviors in mixed-autonomy traffic". In: *Conference on Robot Learning*. PMLR. 2017, pp. 398–407.

[202] Cathy Wu et al. "Flow: A Modular Learning Framework for Mixed Autonomy Traffic". In: *IEEE Transactions on Robotics* (2021).

[203] Cathy Wu et al. "Variance reduction for policy gradient with action-dependent factorized baselines". In: *arXiv preprint arXiv:1803.07246* (2018).

[204] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[205] Lingyun Xiao and Feng Gao. "A comprehensive review of the development of adaptive cruise control systems". In: *Vehicle system dynamics* 48.10 (2010), pp. 1167–1192.

[206] Yang Xing et al. "An ensemble deep learning approach for driver lane change intention inference". In: *Transportation Research Part C: Emerging Technologies* 115 (2020), p. 102615.

[207] H Michael Zhang. "A non-equilibrium traffic model devoid of gas-like behavior". In: *Transportation Research Part B: Methodological* 36.3 (2002), pp. 275–290.

[208] Dongbin Zhao, Bin Wang, and Derong Liu. "A supervised actor–critic approach for adaptive cruise control". In: *Soft Computing* 17.11 (2013), pp. 2089–2099.

[209] Hao Zhou and Jorge Laval. "Longitudinal motion planning for autonomous vehicles and its impact on congestion: A survey". In: *arXiv preprint arXiv:1910.06070* (2019).

[210] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.

[211] Zeyu Zhu and Huijing Zhao. "A survey of deep rl and il for autonomous driving policy learning". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).