Jointly Forecasting and Controlling Behavior by Learning from High-Dimensional Data

Nicholas Rhinehart
The Robotics Institute
Carnegie Mellon University
CMU-RI-TR-19-76

Doctoral committee:

Dr. Kris M. Kitani, Chair	*Carnegie Mellon University*
Dr. Martial Hebert	*Carnegie Mellon University*
Dr. Ruslan Salakhutdinov	*Carnegie Mellon University*
Dr. Sergey Levine	*University of California, Berkeley*
Dr. Paul Vernaza	*Aurora Innovation*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

Abstract

Achieving a precise predictive understanding of the future is difficult, yet widely studied in the natural sciences. Significant research activity has been dedicated to building testable models of cause and effect. From a certain view, the ability to forecast the universe is the "*holy grail*"; the ultimate goal of science. If we had it, we could anticipate, and therefore (at least implicitly) understand all observable phenomena. The human capability to forecast offers complementary motivation. Critical to our intelligence is our ability to *plan* behaviors by considering how our actions are likely to result in future payoff, especially in the presence of other collaborative and competitive agents. In this work, we seek to *computationally model the future in the presence of agent behavior given rich observations of the environment*. The brunt of our focus is to reason about what *agents* could do, instead of other sources of stochasticity. This focus on future agent behavior allows us to *tightly couple and jointly perform forecasting and control*.

The field of Computer Vision (CV) is focused on designing algorithms to automatically understand images, videos, and other perceptual data. However, the field's effort to-date focuses on *non-interactive*, *present-focused* tasks [79, 81, 158, 184]. Most CV contributions are algorithms to answer questions like "what is that", and "what happened", rather than "what *could* happen", or "how *could* I achieve X". Computer Vision has under-explored reasoning about the interactive and decision-based nature of the world. In contrast, Reinforcement Learning (RL) prioritizes modeling interactions and decisions by focusing on how to design algorithms to evoke behavior that maximizes a scalar reward signal. The resulting learning agents, in order to perform well, must have an understanding of how their current behaviors will affect their prospects of future reward. However, in the dominant paradigm of model-free RL [218], agents reason *implicitly* about the future. In contrast, model-based RL learns one-step dynamics to estimate "what *could* happen in the near future". Yet model-based RL primarily focuses on control, rather than explicitly forecasting a single agent (let alone multiple agents).

In this thesis, we consider the problem of designing algorithms to enable computational systems to (1) *forecast* future behavior of intelligent agents given rich observations of their environments, as well as to (2) use this reasoning for *control*. We believe these two problems should be tightly integrated and jointly considered, and use them to structure this thesis. We define *forecasting* to be the problem of *estimating the set of possible outcomes of a system*, whereas *control* is the problem of *producing actions that generate a single outcome of a system*. We often use Imitation Learning and Reinforcement Learning to formulate and situate our work.

We contribute forecasting and control approaches to excel in diverse, realistic, single-agent, and multi-agent domains. The first part of the thesis focuses on progressively designing more capable forecasting models. We proceed through approaches to (1) forecast *single actions* of daily behavior by developing matrix factorization models [169], (2) forecast *goal-driven action trajectories* of daily behavior by developing Online Inverse Reinforcement Learning models [168, 170], (3) forecast *motion trajectories* of vehicles by developing a deep reversible generative models [171, 174]. The second part of the thesis focuses on progressively designing more capable models that tightly couple forecasting and control. We discuss (4) forecasting as *auxiliary supervision for implicitly-planned control* [228], (5) *forecasting and explicitly planning with the same model* [176], and (6) forecasting and planning *future interactions of multiple agents* [175].

With all of my love, I dedicate this dissertation to my parents and siblings.
You bring me to life, and
you make the world glow.

Acknowledgements

I am deeply grateful and forever indebted to a large cast of advisers, friends, collaborators, peers, and family. I give thanks to their mentoring, friendship, dedication, camaraderie, and love that has shaped me and my work. Without them, this thesis would not have been possible. Science is a human endeavor – our work owes no small debt to the relationships that sustain and enrich us.

The dominant factor in my career success and growth has been my Ph.D. adviser, Kris Kitani. It is difficult to imagine a better adviser than Kris. Kris has always done the right things for me, and, plainly put, has never let me down – something that I wish all students could claim of their advisers. Early in my studies, he would challenge me to improve my research justifications and expand my knowledge. After those early years, he began to give me more latitude to explore bigger ideas, while still remaining present to help me hone them. Kris has a remarkable sense of his students's needs. Kris is unshakably patient, kind, and wise. Because of Kris, I have come to be fearless in the face of the unknowns in research and bold in my goals, among countless other improvements. I am inexpressibly grateful for Kris.

I owe a large debt to my M.S. adviser, Drew Bagnell. I met Drew before my senior year of college. I had not settled upon what I would pursue after I graduated. It was a research experience with Drew and his student, Debadeepta Dey, that convinced me that I *had* to pursue research, ideally at CMU. Despite my next-to-nothing knowledge at the time, Drew hired me as an M.S. researcher at CMU. I am forever grateful for the faith he had in me. As Drew's student, I was inspired to pursue technical clarity, precision, and creativity in my work. Drew is another key factor in my career, one without which I may not have pursued a research career at all.

I am very lucky and grateful to have worked with several fantastic collaborators and mentors I had at internships, including Paul Vernaza, Sergey Levine, and Rowan McAllister. Paul's love for mathematics is infectious, and I was not spared. Sergey has stewarded my recent research through his deep insights and unbounded knowledge. Rowan and I have collaborated closely for over a year; it is an almost daily pleasure to hone, improve, and create with Rowan.

This research would also not have been possible without the many friendships and collaborations I have had at CMU, including Bhav Ashok, Michelle Cedeño, Achal Dave, Allie Del Giorno, Katie Lagree, Kumar Shaurya Shankar, Tanmay Shankar, Arjun Sharma, Mohit Sharma, Gunnar Sigurdsson, Wen Sun, and Arun Venkataraman, among many, many others.

I dedicate this thesis to my family: Mom, Dad, Zach, Tessa, Erin, and Elliott. Your love and support drives my heart, and through it, fuels the engine of science.

Contents

Chapter 1

Introduction

1.1 Science Seeks To Forecast; Intelligence Requires Us To Forecast

> Prediction is difficult, especially when it involves the future.
>
> Niels Bohr, Yogi Berra

Achieving a precise predictive understanding of the future is difficult, yet widely studied in the natural sciences. Significant research activity has been dedicated to building testable models of cause and effect. From a certain view, a perfect predictive model of the universe is the "*holy grail*"; the ultimate goal of science. If we had it, we could anticipate, and therefore (at least implicitly) understand all observable phenomena. We approach the difficulty of modeling the future by deferring as much of the modeling as possible to be *computationally learned*. In this work, we seek to *computationally model the future in the presence of agent behavior given rich observations of the environment*. The brunt of our focus is to reason about what *agents* will do, instead of other dynamic aspects of the environment. Whereas many natural science theories offer human-crafted predictive models of physical phenomena, we instead offer a paradigm of learned correlation-based predictive models of behavior-based phenomena.

The human capability to forecast offers complementary motivation. Humans use rich environment observations to inform their understanding, and ultimately, their future behavior. Critical to our intelligence is our ability to *plan* behaviors by considering how our actions are likely to result in future payoff, especially in the presence of other collaborative and competitive agents. We argue that a system *cannot* be intelligent if it cannot explicitly reason about the future of itself and other meaningful entities. By this logic, explicitly reasoning about the future is a necessary component of intelligence. *Therefore, as scientists, we must design systems to explicitly forecast in order to have any hope of building intelligent systems.*

The field of Computer Vision (CV) is focused on designing algorithms to automatically understand images, videos, and other perceptual data. However, the field's effort to-date focuses on *non-interactive*, *present-focused* tasks, like object detection [81], scene classification [184], geometric understanding [79], and activity classification [158]. Most CV contributions are algorithms to answer questions like "what is that", and "what happened", rather than "what *will* happen", "what *could* happen", or "how *could* I achieve X". Computer Vision has under-explored reasoning about the interactive and decision-based nature of the world.

In contrast, Reinforcement Learning (RL) prioritizes modeling interactions and decisions by focusing on how to design algorithms to evoke behavior that maximizes a scalar reward signal. The resulting learning agents, in order to perform well, must have an understanding of how their

current behaviors will affect their prospects of future reward. However, in the dominant paradigm of model-free RL [218], agents reason *implicitly* about the future. In contrast, model-based RL learns one-step dynamics as $P(s'|s, a; \theta)$ or $s' = f(s, a; \theta)$. One-step dynamics provide an *explicit* estimate of "what *could* happen in the near future". Combined with knowledge of how the agent will react to any given situation as a policy $\pi(s'|s, a)$ or $s' = \pi(s, a)$, these objects enable us to forecast the distribution of future outcomes at arbitrarily-long time horizons. Unfortunately, in multi-agent systems, these objects are *insufficient* to forecast the future. We must also estimate how *all* agents will behave, in combination with the one-step world dynamics, in order to achieve a distribution of future outcomes over multiple time-steps.

In this thesis, we consider the problem of designing algorithms to enable computational systems to (1) *forecast* future behavior of intelligent agents given rich observations of their environments, as well as to (2) use this reasoning for *control*. We believe these two problems should be tightly integrated and jointly considered, and use them to structure this thesis. We define *forecasting* to be the problem of *estimating the set of possible outcomes of a system*, whereas *control* is the problem of *producing actions that generate a single outcome of a system*. We often use Imitation Learning and Reinforcement Learning to formulate and situate our work.

We contribute forecasting and control approaches to excel in diverse, realistic, single-agent, and multi-agent domains. The first part of the thesis focuses on progressively designing more capable forecasting models. We proceed through approaches to (1) forecast *single actions* of daily behavior by developing matrix factorization models [169], the(2) forecast *goal-driven action trajectories* of daily behavior by developing Online Inverse Reinforcement Learning models [168, 170], (3) forecast *motion trajectories* of vehicles by developing a deep reversible generative models [171, 174]. The second part of the thesis focuses on progressively designing more capable models that tightly couple forecasting and control. We discuss (4) forecasting as *auxiliary supervision for implicitly-planned control* [228], (5) generating and executing *forecasting and explicitly planning with the same model* [176], and (6) forecasting and planning *future interactions of multiple agents* [175].

1.2 Main Contributions and Organization

Our main contributions, and the remainder of the thesis, are presented in two parts as follows. We will use Fig. 1.1 to visually situate each component of this work. We can also situate our work relative to the learned computational model-outputs of the diagram shown in Fig. 1.2. We can then view the goal of this thesis as towards designing the most general and flexible learned computational model. We seek to 1) generate continuous-coordinate sequences of multi-agent sequences and 2) generate interpetable plans and controls, steerable with high-level directions/goals, that are sequentially fed into an environment that generates high-dimensional features that the computational model uses to inform its predictions and decisions.

1.2.1 Part I: Activity and Motion Forecasting from High-Dimensional Observations

Part I consists of five chapters that present our work on forecasting activities of a first-person camera wearer. Chapter 2 focuses on learning to forecast functionality of environments by observing behaviors from a first-person camera. Chapter 3 focuses on learning to forecast future goals of a first-person camera wearer and doing so online. The latter three chapters present our work on forecasting motions of expert drivers in single- and multi-agent settings with advanced density estimation techniques. Chapter 4 focuses on learning to forecast a vehicle's distribution of future trajectories, then turning its focus to improving the learning procedure of likelihood-based generative models.

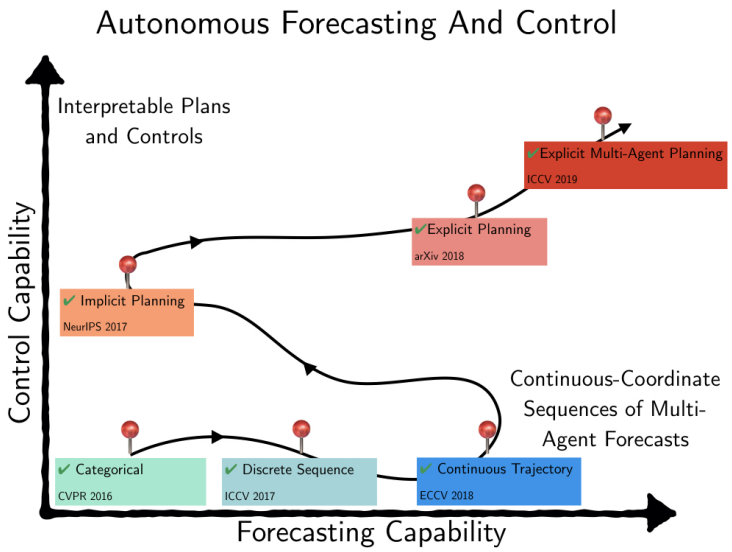


Figure 1.1: Decomposition of this thesis into six primary chapters. Each is summarized by its forecasting and control capabilities.

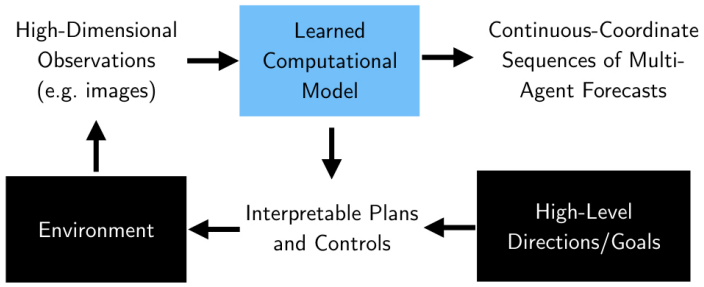


Figure 1.2: Visualization of the system we seek to build. This thesis can be views as progressively improving the 1) forecasting and 2) control output capabilities of a learned computation model.

Chapter 7 focuses on learning to forecast a joint distribution over multi-agent motion of vehicles, and using this distribution for intent-based reasoning about subsets of other vehicles.

1.2.2 Part II: Forecasting for Control from Rich Observations

Part II consists of two chapters that present our work on two techniques to integrate forecasting with controlled behavior. Chapter 5 focuses on improving the learning procedure of sequential decision-making models by encouraging their representations to be predictive of the future. Chapter 6 focuses on using a distribution over future expert vehicle trajectories as a prior for control that is

flexible to new tasks, and robust to noise in task specification.

We conclude by summarizing our contributions and offering promising directions of future work.

1.3 Bibliographical Remarks

This thesis only contains works for which the author was a primary contributor. Chapters 2 and 3 are based on joint work with Kris Kitani [168–170]. Chapters 4 is based on joint work with Paul Vernaza, Kris Kitani, Kihyuk Sohn, and Anqi Liu [174]. Chapter 5 is based on joint work with Arun Venkatraman, Wen Sun, Drew Bagnell, Byron Boots, Kris Kitani, Martial Hebert, and Lerrel Pinto [228]. Chapters 7 and 6 are based on joint work with Rowan McAllister, Sergey Levine, and Kris Kitani [175, 176].

1.4 Excluded Research

I excluded a significant portion of research undertaken during my Ph.D. in order to reduce the length of this document. Below are topics of work I have excluded:

1. Activity Forecasting: Generative Hybrid Activity Forecasting [73].
2. Imitation Learning: Directed Info-GAIL for Learning Hierarchical Policies [200], Learning Neural Parsers [199], Human-Interactive Subgoal Supervision [146], and Learning Sequential Object Detection [177].
3. Reinforcement Learning: Network-to-Network Compression [13].

1.5 Related Work

We now briefly summarize threads of related work, classified into two subsections: Forecasting from Rich Observations (Section 1.5.1) and Decision-Theoretic Modeling (Section 1.5.2). We defer extended related work discussion to each Chapter.

1.5.1 Forecasting from Rich Observations

We categorize related work in this section into four components: Activity Forecasting, Motion Forecasting, Multi-agent Forecasting, and Pixel Forecasting.

1.5.1.1 Activity Forecasting

Data-driven: Activity forecasting methods typically treat the problem of predicting future behaviors as a classification task. In [84, 186], the tasks are to recognize an unfinished event or activity. In [84], the model predicts the onset for a single facial action primitive, *e.g.* the completion of a smile, which may take less than a second. Similarly, [186] predicts the completion of short human to human interactions. In [107], a hierarchical structured SVM is employed to forecast actions about a second in the future, and [232] demonstrates a semi-supervised approach for forecasting human actions a second into the future. Other works predict actions several seconds into the future [34, 104, 118]. In [84, 107], future activities are predicted from a third-person view via classification-based approaches In [185, 186], the future behavior of a person is predicted via activity classification of first-person

video.

Functional understanding: Human actions are deeply connected to the environment in which they are performed. Different environments *afford* different functionalities. Fouhey et al.[59] used detection of sitting, standing, and walking actions to obtain better estimates of 3D geometry for a single densely explored room. Gupta et al.[76] addressed the inverse problem of inferring actions from estimated 3D scene geometry using a single image of a room. Delaitre et al.[45] also used time lapse video of human actions to learn the functional attributes of objects in a single scene. The work of Savva et al.[192] obtains a dense 3D representation of small workspace (e.g.desk and chair space) and learns the functional attributes of the scene by observing human interactions. Another flavor of approaches reason in the joint space of activities and objects. In Moore et al.[134], human actions are recognized by using information about objects in the scene. Gall et al.[63] uses human interaction information to perform unsupervised categorization of objects. Other approaches have capitalized on the interplay between actions and objects: Gupta et al.[75] demonstrate an approach to use object information for pose detection, Yao et al.[247] jointly model objects and poses to perform recognition of both objects and actions, and [153] performs object recognition by observing human activities.

1.5.1.2 Motion Forecasting

Motion Forecasting applications span two primary input domains: third-person observations and first-person observations. See [183] for a survey of methods that span both input domains applied to the task of human motion forecasting.

Third-person observations: The method of [114] predicts future trajectories of wide-receivers from surveillance video. In [16, 101, 128, 246] future pedestrian trajectories are predicted from surveillance video. Deterministic vehicle predictions are produced in [93], and deterministic pedestrian trajectories are produced in [8, 147, 178]. However, non-determinism is a key aspect of forecasting: the future is generally uncertain, with many plausible outcomes. While several approaches forecast distributions over trajectories [62, 113], they do not produce probability density functions.

First-person observations: Other trajectory forecasting approaches use demonstrations observed from a bird's-eye view; [246] infers latent goal locations and [9] employ LSTMs to jointly reason about trajectories of multiple humans. In [206], the model forecasted short-term future trajectories of a first-person camera wearer by retrieving the nearest neighbors from a dataset of first-person trajectories under an obstacle-avoidance cost function, with each trajectory representing predictions of where the user will move in view of the frame; in [209], a similar model with learned cost function is extended to multiple users.

1.5.1.3 Multi-agent Forecasting

Game-theoretic: Traditionally, multi-agent planning and game theory approaches explicitly model multiple agents' policies or internal states, usually by generalizing Markov decision processes (MDPs) to multiple decisions makers [39, 221]. These frameworks facilitate reasoning about collaboration strategies, but suffer from "state space explosion" intractability except when interactions are known to be sparse [131] or hierarchically decomposable [58].

Data-driven: Data-driven approaches have been applied to forecast complex interactions between multiple pedestrians [8, 20, 55, 77, 128], vehicles [46, 113, 148], and athletes [54, 110, 114, 210,

248, 250]. These methods attempt to generalize from previously observed interactions to predict multi-agent behavior in new situations. Generative models for multi-agent forecasting and control have been proposed. In terms of multi-agent forecasting, our work is related to [193] which uses a conditional VAE [99] encoding of the joint states of multiple agents together with recurrent cells to predict future human actions.

1.5.1.4 Pixel Forecasting

Pixel Forecasting methods generate full image or video representations of predictions, endowing their samples with interpretability. In [232–234], unsupervised model are learned to generate sequences and representations of future images. In [235], surveillance image predictions of vehicles are formed by smoothing a patch across the image. [236] and [231] also predict future video frames with an intermediate pose prediction. In [57], video predictions are used to inform a robot's behavior. In [23], image boundaries are predicted. One drawback to image-based forecasting methods is the difficulty in measuring the model's quality, a drawback shared by many popular generative models.

1.5.2 Decision-Theoretic Modeling

We categorize related work in this section into two components: Imitation Learning and Model-based Planning. Imitation Learning (IL) learns a model to mimic demonstrated behavior [3, 144]. Model-based Planning is the approach of using a learned dynamics model of how the world or a system operates in order to control an agent.

1.5.2.1 Imitation Learning and Reinforcement Learning

Behavior Cloning: Behavior cloning [144, 157] is an IL approach that learns to mimic one-step behaviors of an expert by applying straightforward supervised learning without interacting with the underlying environment until test-time. A body of previous work has explored BC for autonomous driving in the CARLA simulator [41, 42, 122, 123, 191]. These approaches condition on scene images and a discrete set of directives by a high-level planner. [123] is a BC+RL approach that collects data online to bootstrap a policy learned from observations. In contrast to some Imitation Learning methods, including BC [157], behavior forecasting models are not executed in the environment of the observed agent – they are instead predictive models of the agent. In this sense, forecasting can be considered non-interactive Imitation Learning without execution. One key difference is that in forecasting, we are not required to actually execute our plans in the real world.

Inverse Reinforcement Learning: Inverse Reinforcement Learning (IRL) is a form of imitation learning in which a reward function is learned to model demonstrated behavior. Other Inverse Reinforcement Learning approaches have been used to predict pedestrian behavior and high-level taxi planning [101, 256]. In the IRL method of [245], a cost map representation is used to plan vehicle trajectories. However, no time-profile is represented in the predictions, preventing use of time-profiled metrics and modeling. GAIL [83, 121] is also a form of IRL, yet its adversarial framework and policy optimization are difficult to tune and lead to slow convergence.

1.5.2.2 Model-based planning

Imitative Planning: Related to Imitation Learning, several prior works [11, 208, 220] used imitation learning to train policies that contain planning-like modules as part of the model architecture. Englert et al.[52] propose a one-step model-based Imitation Learning approach.

Model-based Reinforcement Learning (MBRL) plans through a dynamics model in order to optimize a reward function [15, 44]. Because the dynamics of MBRL captures only what is possible, rather than what is expert-preferred, the task of evoking expert-like behavior is offloaded to the reward function, which can be difficult and time-consuming to craft properly. Recent work has shown that there exists settings in which MBRL requires exponentially less samples than model-free RL [213]

System Identification with Learned Models System identification focuses on designing models of dynamical systems, primarily for control purposes, and is related to MBRL. Across ML and CV, Recurrent Neural Networks (RNNs) are often employed to perform indirect system identification [111, 216]. In lieu of ground truth access to latent states, RNNs employ internal states to summarize previous data, serving as a learner's memory. Internal states are modified towards minimizing the target application's loss, e.g., minimizing observation loss in filtering or cumulative reward in reinforcement learning. The target application's loss is not directly defined over the internal states: they are updated via the chain rule (backpropagation) through the global loss. Although this modeling is indirect, RNNs nonetheless can achieve state-of-the-art results on many robotics [51, 80], vision [140, 143], and natural language tasks [38, 68, 162] when training succeeds. However, recurrent model optimization is hampered by two main difficulties: 1) non-convexity, and 2) the loss does not directly encourage the internal state to model the latent state. A poor internal state representation can yield poor task performance, but rarely does the task objective directly measure the quality of the internal state. Predictive-State Representations (PSRs) [26, 82, 86] offer an alternative internal state representation to that of RNNs in terms of the available observations.

Part I

Activity and Motion Forecasting from High-Dimensional Observations

Chapter 2

Forecasting Singular Actions with Action Maps

2.1 Introduction

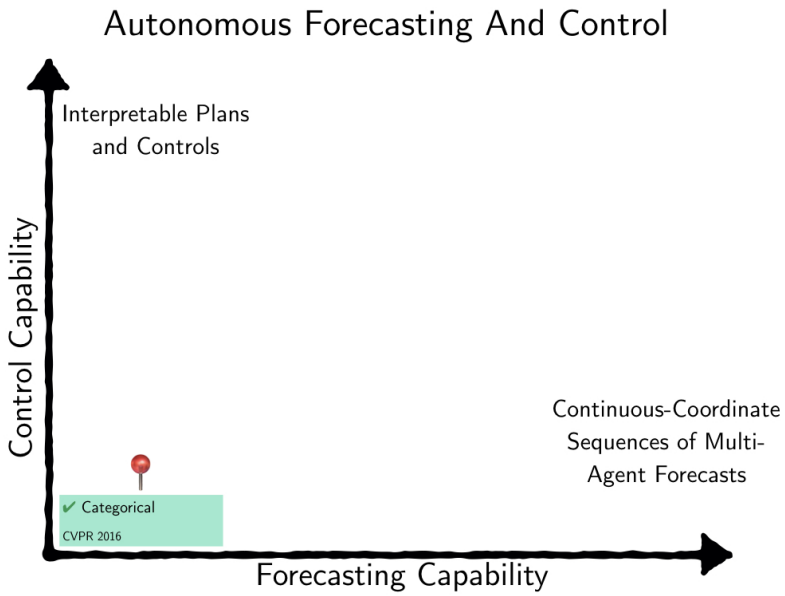


Figure 2.1: This chapter focuses on building a one-step categorical forecasting approach.

The goal of this work is to endow intelligent systems with the ability to understand the functional attributes of their environment. Such functional understanding of spaces is a crucial component of holistic understanding and decision making by any agent, human or robotic. Functional understanding of a scene can range from the immediate environment to the distant. For example, at the scale of a single room, a person can perceive the arrangement of tables, chairs, and computers in an office environment, and reason that they could sit down and type at the computer. People can also reason about the functionality about nearby rooms, for example, the presence of a kitchen down the hall from the office is useful functional and spatial information for when the person decides to prepare a meal. The goal of this work is to learn a computational model of the functionality of large environments, called *Action Maps* (AMs), by observing human interactions and the visual context of those action within a large environment. These can be thought of as forecasting a *one-step* future as

a discrete action, as shown in Fig. 2.1.

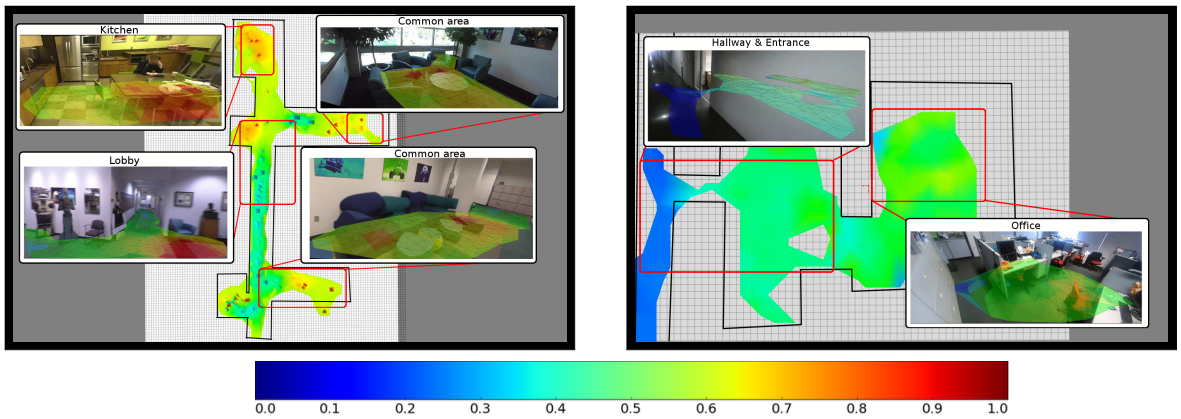


Figure 2.2: Action Map prediction for the `sit` activity by using our method to combine appearance data and activity observations. Activity and appearance information from the top scene in combination with *only* appearance information (no activity observations) from the bottom scene is used to model the relationship between activities, scene information, and object information to make predictions for both scenes. Areas in the scenes where a person can sit are estimated by our method, such as the chairs and couches in both views.

There has been significant work in the area of automating the functional understanding of an environment, though much has focused on single scenes [45, 59, 63, 76, 94, 105]. In this work, we aim to extend automated functional understanding to very large spaces (e.g., an entire office building or home). This presents two key technical challenges:

- How can we capture observations of activity across large environments?
- How can we generalize functional understanding to handle the inevitable data sparsity of less explored or new areas?

In order to address the first challenge of observing activity across large environments, we take a departure from the fixed surveillance camera paradigm, and propose an approach that uses a first-person point-of-view camera. By virtue of its placement, its view of the wearer's interactions with the environment is usually unobstructed by the wearer's body and other elements in the scene. An egocentric camera is portable across multiple rooms, whereas fixed cameras are not. An egocentric camera allows for the observation of hand-based activities, such typing or opening doors, as well as the observation of some ego-motion based activities, such as sitting down or standing. The first-person paradigm is well suited for large-scale sensing and allows observation of interactions with many environments.

Although we can capture a large number of observations of activity across large environments with wearable cameras, it is still not practical to wait to observe all possible actions in all possible locations. This leads to the second technical challenge of generalizing functional understanding from a *sparse* set of action observations, which requires generalization to new locations. Our method generalizes by using another source of visual observation – which we call *side-information* – that encodes per-location cues relevant to activities. In particular, we propose to extract visual side-information using scene classification [253] and object detection [66] techniques. With this information, our method learns to model the relationship between actions, scenes, and objects. In a scene with no actions, we use scene and object information, coupled with actions in a separate scene,

to infer possible actions. We propose to solve the problem of generalizing functional understanding (i.e., generating dense AMs) by formulating the problem as matrix completion. Our method constructs a matrix where each row represents a location and each column represents an action type (e.g., read, sit, type, write, open, wash). The goal of matrix completion is to use the observed entries to fill the missing entries. In this work, we make use of Regularized Weighted Non-Negative Matrix Factorization (RWNMF) [72], allowing us to elegantly leverage *side-information* to model the relationship between activities, scenes, and objects, and predict missing activity affordances.

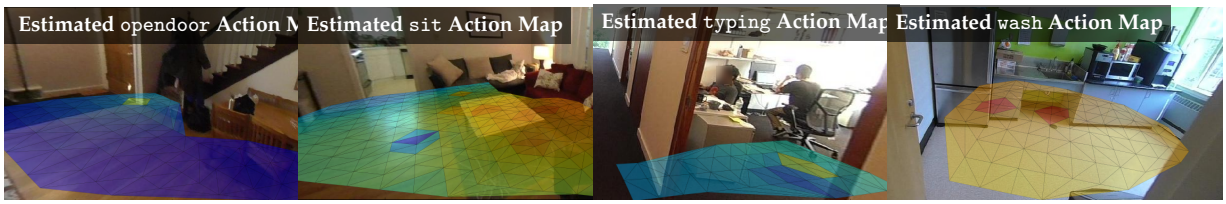


Figure 2.3: Projected Action Map examples learned by our method. With global estimates of large Action Maps produced by our method, we use localized images within the scene to show visualizations of the Action Maps by projecting them to the images.

2.1.1 Contributions

To the best of our knowledge, this is the first work to generate *Action Maps*, such as those in Figures 2.2 and 2.3, over large spaces using a wearable camera. The first-person vision paradigm is an essential tool for this problem, as it can capture a wide range of visual information across a large environment. Our approach unifies scene functionality information via a regularized matrix completion framework that appropriately addresses the issue of sparse observations and provides a vehicle to leverage visual side information.

We demonstrate the efficacy of our proposed approach on five different multi-room scenes: one home and four office environments. Our experiments in real large-scale environments show how first-person sensing can be used to efficiently observe human activity along with visual side-information across large spaces. **1)** We show that our method can be used to model visual information from both single and multiple scenes simultaneously, and makes efficient use of all available activity information. **2)** We show that our method's power increases as the set of performed activity increases. **3)** Furthermore, we demonstrate how our proposed matrix factorization framework can be used to leverage sparse observations of human actions along with visual side-information to perform functionality estimation of large *novel* scenes in which *no activities* have been demonstrated. We compare our proposed method against natural baselines such as object-detection-based Action Maps and scene classification, and show that our approach outperforms them in nearly all of our experiments. **4)** Additionally, as a proof-of-concept application of the rich information in an Action Map, we present an application of our Action Maps as priors for localization.

2.1.2 Background

Human actions are deeply connected to the scene. Scene context (e.g., a chair or common room) can be a strong indicator of actions (e.g., sitting). Likewise, observing an action like sitting, is a strong indicator that there must be a sittable surface in the scene. In the context of time lapse video, Fouhey et al. [59] used detection of sitting, standing, and walking actions to obtain better estimates of 3D geometry for a single densely explored room. Gupta et al. [76] addressed the inverse problem of inferring actions from estimated 3D scene geometry using a single image of a room. Their approach synthetically inserted skeleton models into the 3D scene to reason about possible

functional attributes of the scene. Delaitre et al. [45] also used time lapse video of human actions to learn the functional attributes of objects in a single scene. The work of Savva et al.[192] obtains a dense 3D representation of small workspace (e.g.desk and chair space) and learns the functional attributes of the scene by observing human interactions. Similar to previous work, this work seeks to understand the functionality of scenes. However, limitations of previous work include the reduced size of the physical space and the presumed density of interactions. In contrast, our approach attempts to infers the dense functionality over an entire building (e.g., office floor or house), and reasons about multiple large scenes simultaneously by modeling the relationship between scene information, object information, and sparse activities.

Another flavor of approaches reason in the joint space of activities and objects. In Moore et al. [134], human actions are recognized by using information about objects in the scene. Gall et al. [63] uses human interaction information to perform unsupervised categorization of objects. Other approaches have capitalized on the interplay between actions and objects: Gupta et al. [75] demonstrate an approach to use object information for pose detection, and Yao et al. [247] jointly model objects and poses to perform recognition of both objects and actions. The approach of [153] performs object recognition by observing human activities, and notes an important idea that our approach also uses: whereas object information may sometimes be too small in detail, human activities usually are not. We capitalize on this observation close-up observation capability of an egocentric camera.

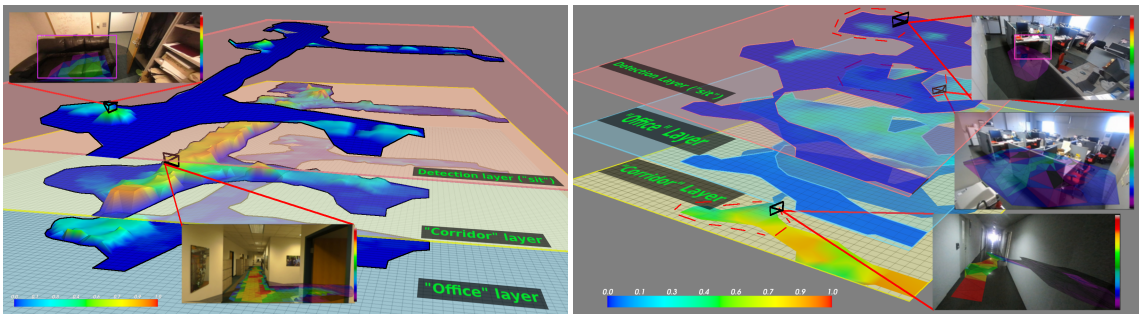
The egocentric paradigm is an excellent method for understanding human activities at close range [53, 120, 155, 207]. Our work builds on such egocentric action recognition techniques by associating actions with physical locations in a single holistic framework. By bringing together ideas from single image functional scene understanding, object functionality understanding and egocentric action analysis, we propose a computational model that enables cross-building level functional understanding of scenes.

2.2 Constructing Action Maps

Our goal is to build *Action Map*s that associate possible actions for every spatial location on a map over a large environment. We decompose the process into three steps. We first build a physical map of the environment by using egocentric videos to obtain a 3D reconstruction of the scene using structure from motion. Second, we use a collection of recorded human activity videos recorded with an egocentric camera to detect and spatially localize actions. This collection of videos is also used to learn the visual context of actions (i.e., scene appearance and object detections) which is later used as a source of side information for modeling and inference. Third, we aggregate the localized action detection and visual context data using a matrix completion framework to generate the final Action Map. The focus of our method is the third step, which we describe next. We mention how we obtain the visual context in Section 2.2.1.1, and describe the first two steps in detail in Section 2.3.2.

2.2.1 Action Map Prediction as Matrix Factorization

We now describe our method for integrating the sparse set of localized actions and visual side-information to generate a dense Action Map (AM) using regularized matrix completion. Our goal is to recover an AM in matrix form $\mathbf{R} \in \mathbb{R}_{+}^{M \times A}$, where M is the number of locations on the discretized ground plane and A is the number of possible actions. Each row of the AM matrix $\mathbf{R}$ contains the action scores $\mathbf{r}_m$, where m is a location index, and each entry r_{ma} describes the extent to which an activity a can be performed at location m. To complete the missing entries of $\mathbf{R}$, we design a



(a) Office Flr. A Features (b) Office Flr. B Features

Figure 2.4: Several Office Flr. A and Office Flr. B Features. The “office" and “corridor" layers correspond to the features from the scene classification CNN, and the “sit" layer corresponds to the object detection CNN features aggregated across all `sit`-able objects, which is also one of the baselines as described in Section 2.3.2. This figure demonstrates our idea that object information and scene information can be used to relate scenes to each other. This relationship is the basis for transferring and sharing activity functionality between scenes. Heatmaps from several layers are shown projected into localized images from the scene. Note that the “office" portion of Office Flr. A also contains sittable regions, and that the much larger “office" area in Office Flr. B contains a select few sittable regions. The corridors in both scenes are described well by the features, and these areas strongly correlate with an *an absence* of functionality, as scene in Figure 2.2.

similarity metric for our side-information, enabling the method to model the relationship between activities, scenes, and objects.

We impose structure on the rows and columns of the AM matrix by computing similarity scores with the side-information. Examples of this side information are shown in Figure 2.4, where two features from scene classification, plus one feature from object detection are shown in the same physical space as the AM. Figure 2.4 serves to further motivate the idea of exploiting scene and object information between two different scenes to relate the functionality of the scenes. We define three kernel functions based on scene appearance, object detections and spatial continuity. This structure is integrated as regularization in the RWNMF objective function (Equation (2.2)).

2.2.1.1 Integrating Side-Information

To integrate side-information into our formulation, we build two weighted graphs that describe the cross-location (row) similarities, and cross-action (column) similarities. We are primarily interested in the cross-location similarities, and discuss how we handle the cross-action similarities in Section 2.2.2. To build the cross-location graph, we aggregate the spatial proximity, scene-classification, and object detection information as a linear combination of kernel-based similarities, as shown in Equation 2.1.

For every location a in the AM, we compute the scene classification score $\mathbf{p}_a = [p_{1a} \dots p_{Ca}]$ for each image as the average of the C-dimensional outputs from the Places-CNN of images within a small radius.

We use Structure-from-Motion (SFM) keypoints inside each detection to estimate the back-projected 3D location of the detected object in the environment by taking the mean of their 3D locations, which are then projected to the ground plane to form a set $\mathcal{D}_f$ for each object category $f \in [1 \dots F]$. The SFM reconstruction is also used to localize images and described further in Section 2.3.2. We calculate the object detection scores $\mathbf{o}_a = [o_{1a} \dots o_{Fa}]$ for each location a as the max score of object detection of the nearby back-projected object detections $d \in \mathcal{D}_f$ within a

$r = \sqrt{2}$ grid-cell radius, exponentially weighted by its distance along the floor from the object z_d: $o_{fa} = \max_{d \in \mathcal{D}_f} {}^{1}/_{\sqrt{2r^2\pi}} \exp(-z_d^2/2r^2)$.

We wish to enforce similarity of activities between nearby locations, as well as between locations that have similar object detections and scene classification description. Between any two locations a, b, and given associated scene classification scores $\mathbf{p}_a, \mathbf{p}_b$, object detection scores $\mathbf{o}_a, \mathbf{o}_b$, and 2D grid locations $\mathbf{x}_a, \mathbf{x}_b$ the kernel is of the form:

$$k(a,b) = (1-\alpha)k_{\mathrm{s}}(\mathbf{x}_a, \mathbf{x}_b) + \frac{\alpha}{2}k_p(\mathbf{p}_a, \mathbf{p}_b) + \frac{\alpha}{2}k_o(\mathbf{o}_a, \mathbf{o}'_b), \tag{2.1}$$

where k_s is an RBF kernel between the spatial coordinates of each location, k_p and k_o as χ^2 kernels on scene classification scores and object detection scores, and k_o has 0 similarity between locations with no object score.

Thus, there is a tradeoff between the k_s, k_p and k_o kernels, controlled by α. When $\alpha = 0$, only spatial smoothness is considered, and when $\alpha = 1$, only scene classification and object detection terms are considered, ignoring spatial smoothness. When a location in one scene is compared to a location in a new scene or the same scene, $k(\cdot,\cdot)$ returns higher scores for locations with similar objects and places, and as shown Section 2.2.2, places more regularization constraint on the objective function, rewarding solutions that predict similar functionalities for *both* locations.

2.2.2 Completing the Action Map Matrix

To build our model, we seek to minimize the RWNMF objective function in Equation 2.2:

$$J(\mathbf{U},\mathbf{V}) = \left\| \mathbf{W} \circ (\mathbf{R} - \mathbf{U}\mathbf{V}^T) \right\|_F^2 + \frac{\lambda}{2}\sum_{i,j}^{M} \left\|\mathbf{u}_i - \mathbf{u}_j\right\| \mathbf{K}^U_{ij} + \frac{\mu}{2}\sum_{i,j}^{A} \left\|\mathbf{v}_i - \mathbf{v}_j\right\| \mathbf{K}^V_{ij} \tag{2.2}$$

where $\mathbf{U} \in \mathbb{R}_+^{M \times D}$, $\mathbf{V} \in \mathbb{R}_+^{A \times D}$, together form the decomposition, $\mathbf{W} \in \mathbb{R}_+^{M \times A}$ is the weight matrix with 0s for unexplored locations, and $\mathbf{K}^U$ the kernel Gram matrix of the side information defined by its elements: $\mathbf{K}^U_{ij} = k(i,j)$. The squared-loss term penalizes decompositions with values different from the observed values in $\mathbf{R}$. The term involving $\mathbf{K}^U$ penalizes decompositions in which highly similar locations have different decompositions in the rows ($\mathbf{u}_i^T$) of $\mathbf{U}$. Roughly, locations with high similarity in scene appearance, object presence, or position impose penalty on the resulting decomposition for predicting different affordance values in the AM. The term involving $\mathbf{K}_V$ corresponds to the cross-action smoothing, which we take as the identity matrix, enforcing no penalty for differences across per-location action labels.

To minimize the objective function, we use the regularized multiplicative update rules following [72]. Multiplicative update schemes for NMF are generally constructed such that their iterative application yields a non-increasing update to the objective function; [72] showed that these update rules yield non-increasing updates to the objective function. Thus, after enough iterations, a local minima in the objective function is found, and the resulting decomposition and its predictions are returned.

Values in $\mathbf{W}$ are set to counteract class imbalance. The number of observed values for each activity is computed as n_c, and assigned to each nonempty location i's corresponding entry as $w_{ic} = 1/n_c$, and the zeros from observed cameras associated with no activities as $w = 1/n_z$.

Scene	# GT locs.	# Actions	Length	r_e	r_a
Office Flr. A	40	90	53.3 min.	0.59	0.03
Office Flr. D	15	44	32.8 min.	0.23	0.03
Office Flr. C	44	14	12.2 min.	0.16	0.01
Office Flr. B	50	13	3.3 min.	0.67	0.04
Home A	15	17	13.4 min.	0.75	0.04

Table 2.1: Scene stats. The number of GT locations is the number of distinct places a specific activity can be performed. The number of activity demonstrations is the total number of demonstrations collected in each environment. $r_e = \frac{\#\text{cells explored}}{\#\text{total cells}}$, $r_a = \frac{\#\text{cells with non-empty actions}}{\#\text{total cells}}$.

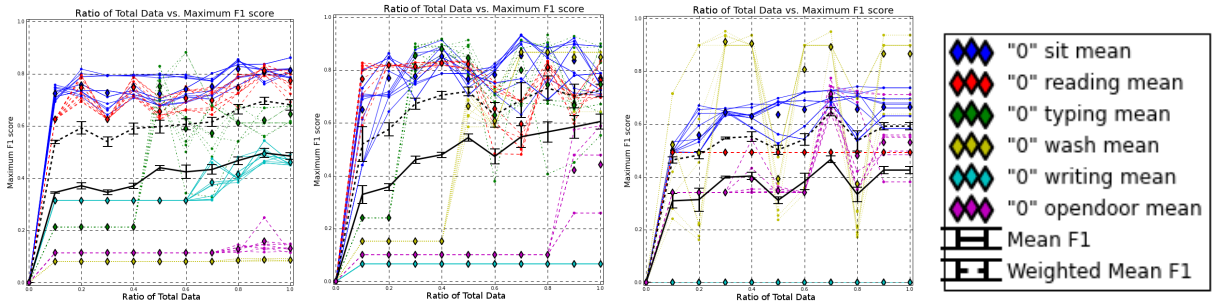
Approach	W. Max F1	W. Mean F1	Max F1	Mean F1	W. Max F1	W. Mean F1	Max F1	Mean F1
	Office Flr. A				Office Flr. C			
S sng	0.73	0.72 ± 0.01	0.44	0.43 ± 0.02	0.74	**0.66** ± 0.1	0.48	0.42 ± 0.06
SOP_D sng	0.63	0.61 ± 0.01	0.34	0.32 ± 0.01	0.67	0.46 ± 0.08	0.41	0.29 ± 0.05
SOP sng	0.74	0.69 ± 0.04	**0.56**	0.5 ± 0.04	0.68	0.53 ± 0.1	0.53	0.44 ± 0.06
SOP_D all	0.75	0.71 ± 0.02	0.44	0.43 ± 0.01	0.67	0.55 ± 0.06	0.45	0.38 ± 0.03
SOP all	**0.76**	**0.73** ± 0.02	0.54	**0.51** ± 0.02	**0.77**	0.58 ± 0.07	**0.56**	**0.46** ± 0.04
	Home A				Office Flr. D			
S sng	0.57	0.53 ± 0.04	0.35	0.34 ± 0.02	0.68	0.57 ± 0.11	0.57	0.45 ± 0.12
SOP_D sng	0.5	0.48 ± 0.01	0.26	0.24 ± 0.02	0.56	0.49 ± 0.05	0.37	0.32 ± 0.04
SOP sng	**0.62**	**0.6** ± 0.01	0.43	**0.4** ± 0.02	0.69	0.55 ± 0.08	0.68	0.54 ± 0.07
SOP_D all	0.52	0.49 ± 0.03	0.27	0.25 ± 0.02	0.81	0.68 ± 0.07	0.59	0.46 ± 0.08
SOP all	**0.62**	0.55 ± 0.03	**0.45**	**0.4**± 0.02	**0.82**	**0.73** ± 0.08	**0.77**	**0.61** ± 0.09
	Office Flr. B							
S sng	0.56	0.55 ± 0.01	0.38	0.38 ± 0.01				
SOP sng	0.56	0.55 ± 0.01	0.44	0.38 ± 0.03				
SOP_D all	**0.58**	**0.56** ± 0.01	0.39	0.37 ± 0.03				
SOP all	**0.58**	**0.56** ± 0.01	**0.53**	**0.44** ± 0.04				

Table 2.2: Prediction results by using the activity observations for each scene ("sng"), and, as separate results, by simultaneously fitting data from all scenes ("all"). By using observations from all scenes, the performance of our method on each scene improves over using each scene's observation data alone. Additionally, our method is able to integrate activity detections without much performance loss: a $_D$ suffix indicates activity detection predictions were used, otherwise, labelled activities were used. "S" stands for spatial kernel only, and "SOP" stands for "Spatial+Object Detection+Scene Classification" kernels. The spatial kernel only is useful yet outperformed by the full model. Side information from multiple scenes generally improves the performance.

2.3 Experiments

Our dataset consists of 5 large, multi-room scenes from various locations. Three scenes, Office Flr. A, Office Flr. D, and Office Flr. C, are taken from three *distinct* office buildings in the United States, and another scene, Office Flr. B, comes from an office building in Japan. Each office scene has standard office rooms, common rooms, and a small kitchen area. A final scene, Home A, consists a kitchen, a living room, and a dining room. See Table 2.1 for scene activity and sparsity statistics. Our goal is to predict dense Action Maps from sparse activity demonstrations.

The first experiments (Section 2.3.3) measure our method's performance when supplied with all observed action data that covers on average about half of all locations and some actions (See



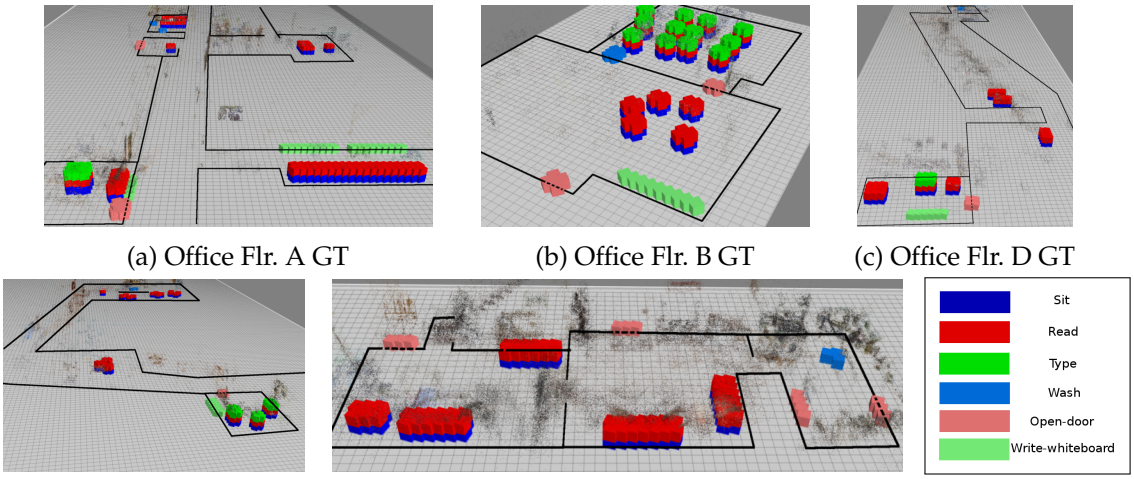
(a) Office Flr. A Elapse (b) Office Flr. D Elapse (c) Home A Elapse (d) Legend

Figure 2.5: Performance improves a function of available data. For each parameter setting, we show the $F1$ scores for each activity label, as well as the mean and weighted mean of the $F1$ scores across all parameter settings and activity labels. Some variations in performance are observed as new activities are introduced, as the correlations between an established activities and newly introduced activities are initially sparse. As more data is collected, erroneous correlations are unlearnt, and correct ones are reinforced.

Approach	W. Max F1	W. Mean F1	Max F1	Mean F1	W. Max F1	W. Mean F1	Max F1	Mean F1
	Office Flr. B				Office Flr. D			
RFC	0.38	0.38	**0.62**	**0.62**	0.27	0.27	0.41	**0.41**
Det.	0.59	0.59	0.33	0.33	0.44	0.44	0.28	0.28
NMF	0.35	0.35	0.24	0.24	0.65	0.65	0.40	0.40
SO	0.69	0.67 $\pm$ 0.02	0.44	0.42 $\pm$ 0.01	0.65	0.51 $\pm$ 0.12	0.46	0.36 $\pm$ 0.09
SP	**0.74**	**0.69** $\pm$ 0.02	0.46	0.43 $\pm$ 0.02	**0.68**	**0.55** $\pm$ 0.12	**0.51**	0.38 $\pm$ 0.09
SOP	0.57	0.54 $\pm$ 0.03	0.28	0.26 $\pm$ 0.02	0.42	0.36 $\pm$ 0.02	0.28	0.25 $\pm$ 0.01
	Office Flr. C				Home A			
RFC	0.24	0.24	0.37	0.37	0.28	0.28	0.35	0.35
Det.	0.54	0.54	0.31	0.31	0.53	0.53	0.25	0.25
NMF	0.39	0.39	0.27	0.27	0.43	0.43	0.25	0.25
SO	0.67	0.55 $\pm$ 0.1	0.47	0.39 $\pm$ 0.07	0.59	0.51 $\pm$ 0.07	0.41	0.33
SP	0.61	0.56 $\pm$ 0.08	0.47	0.39 $\pm$ 0.06	**0.61**	**0.58** $\pm$ 0.01	**0.45**	**0.42** $\pm$ 0.03
SOP	**0.74**	**0.63** $\pm$ 0.05	**0.64**	**0.54** $\pm$ 0.05	0.54	0.45 $\pm$ 0.03	0.3	0.26 $\pm$ 0.01

Table 2.3: Performance of our algorithm by using activity observations from Office Flr. A to make predictions in novel scenes. Each baseline method is run with a single parameter setting, and thus their maxes and means are equivalent. The baseline methods “RFC", “Det.", and “NMF" correspond to the Random Forest Classification, Object Detection AMs, and non-regularized NMF augmented matrix approaches, respectively. Variants of our approach, **SO**, **SP**, and **SOP** correspond to using “Spatial+Object Detection" kernels, “Spatial+Scene Classification" kernels, and “Spatial+Object Detection+Scene Classification" kernels. Multiple metrics are considered to observe the effects of ground-truth class imbalance, and means are used to quantify performance across a variety of parameter settings.

Table 2.1 for the coverage statistics). Additionally, this experiments compares against performance of the spatial kernel-only approach, which serves to illustrate the utility of including side-information. However, as it takes some time to collect the observations of each scene, we demonstrate a second set of experiments (Section 2.3.4), to showcase our method handling fractions of the already sparse observation data while still maintaining reasonable performance. In Section 2.3.5, our third set of experiments shows that if our method is presented with novel scenes for which there is *zero* activity



(a) Office Flr. A GT (b) Office Flr. B GT (c) Office Flr. D GT

(d) Office Flr. C GT (e) Home A GT (f) Legend

Figure 2.6: Ground truth labels and SFM points in each scene. Dotted lines indicate a doorway, solid lines indicate walls.

demonstrations, our method can still make predictions in these new environments. This final set of experiments also investigates which side-information is most helpful for our task.

2.3.1 Performance scoring

To evaluate an AM, we perform binary classification across all activities and compute mean $F1$ scores. We collect the ground truth activity classes for every image in the scene by retrieving them from labelled grid cells, as shown in Figure 2.6, in a small triangle in front of each camera, which represents the viewable space. We collect the predicted AM scores from the same grid cells and average the scores to produce per-image AM scores. We used 100 evenly-spaced thresholds to evaluate binary classification performance by averaging $F1$ scores across the thresholds. We report $F1$ scores as opposed to the overall accuracy, as the overall accuracy of our method is very high due to the large amount of space in each scene with no labelled functionality (a large amount of "true negatives"). The activity classes we use are `sit`, `type`, `open-door`, `read`, `write-whiteboard` and `wash`. This set of activities provides good coverage of common activities that a person can do in an office or home setting. To summarize results, we compute the unweighted and weighted averages of per-class $F1$ scores, where the weighted average is computed by using the normalized counts of the GT classes in the images.

2.3.2 Preprocessing and parameters

The first step to build the AM is to build a physical map of the environment. We use Structure-From-Motion (SFM) [242] with egocentric videos of a walk through of the environment to obtain a 3D reconstruction of the scene. Next, we consider two important categories of detectable actions: (1) those that involve the user's hands (gesture-based activities), and (2) those that involve significant motion of the user's head, or egomotion-based activities. We used the deep network architecture inspired by [201] to perform activity detection, as the two stream network takes into account both appearance (e.g., hands and objects) as well as motion (e.g., optical flow induced by ego-motion and local hand-object manipulations). When actions are detected by our action recognition module, we need a method for estimating the location of this action. We use the SFM model to compute the 3D camera pose of new images.

As we define an AM over a 2D ground plane (floor layout), we project the 3D camera pose associated to an action to the ground plane. To obtain a ground plane estimate, we fit a plane to a collection of localized cameras using SFM. We assume that the egocentric camera lies approximately at eye level, thus this *height plane* is tangent to the top of the camera wearer's head. We then translate this plane along its normal, while iteratively refitting planes with RANSAC to points in the SFM model. Once we have an estimate of the 2D ground plane in 3D space, we can use it to project the localized actions onto the ground plane. When dealing with multiple scenes, distances must be calibrated between them. We use prior knowledge of the user's height to form estimates of the absolute scale of each scene. Specifically, we use the distance between the ground plane and the user height plane, along with a known user height, to convert distances in the reconstruction to meters. Finally, we grid each scene with cells of size 0.25 meters. (we use a radius of 2 grid cells, which is ~ 0.5 meters after metric estimation).

Since actions are often strongly correlated with the surrounding area and objects, as shown in Figure 2.4, we also extract the visual context of each action as a source of side-information. For every image obtained with the wearable camera, we run scene classification and object detection with [253] and [66]. We use the pre-trained "Places205-GoogLeNet" model for scene-classification, which yields 205 features per image, one per each scene type, and a radius of 2 grid cells inside which to average the classification scores. For object detection, we use the pretrained "Bvlc_reference_rcnn_ilsvrc13" model, which performs object detection for 205 different object categories, and use NMS with overlap ratio 0.3, and min detection score 0.5.

We use a small grid of parameters for our method ($\alpha \in [0, .1, .3, .5, .7, .9, 1]$, $\lambda \in [10^{-3}, 10^{-2}]$, $\gamma \in [100, 1000]$), where each γ is used for the χ^2 kernels, and evaluate performance of multiple runs as the cross-run maximum and cross-run average of each of the various scores. In a scenario with many additional test scenes, a single choice of parameters could be selected via cross-validation. We also consider variations of our kernel that use different combinations of side-information: Spatial+object detection (SO), Spatial+scene classification (SP), and Spatial+object detection+scene classification (SOP). In the first two cases, the $\frac{\alpha}{2}$ weight of Equation 2.1 becomes α for the object detection or scene classification kernel that is on, and 0 for the other.

2.3.3 Full observation experiments

When all activity observations are available, our method is able to perform quite well. The dominant source of error is that of camera localization, which reduces the spatial precision of the AM. In Table 2.2, we evaluate the performance of our method run on each scene separately, as well as running once with all of the scenes in a single matrix. When multiple scenes are used, side-information is crucial: without it, there is no similarity enforced across scenes. In single scene case, we find that using a spatial kernel only can perform well, yet is generally outperformed by using all side information, especially when side information and activity demonstrations are present from other scenes. By using the data from all scenes simultaneously in a global factorization, performance increases globally over using each single scene's data alone. This is expected and desirable: simultaneous understanding of multiple scenes can improve as the set of available scenes with observation data grows.

2.3.4 Partial observation experiments

We expose our algorithm to various fractions of the total activity demonstrations to simulate an increasing amount of observed actions. We find that performance is high even with only a few demonstrations and steadily increases as the amount of activity demonstrations increases. The Office Flr. A, Office Flr. D, and Home A scenes have enough activity demonstration data to illustrate

the performance gains of our method as a function of the available data. We show quantitative per-class results for these in Figure 2.5. Sharp increases can be observed in the per-class trends, which correspond to the increase of coverage of each activity class. In Figure 2.7, we show the overhead view of the AM for the `sit` and `type` labels for the Office Flr. A as a function of the available data, where it can be seen how the AM qualitatively improves over time as observations are collected.

2.3.5 Novel scene experiments

Another scenario is the task of predicting AMs for novel scenes containing *zero* activity observation data. Our method leverages the appearance and activity observation data in one scene, and only appearance data in the novel scene to make predictions. We now introduce three baselines we consider. The first baseline is to perform per-image classification with the object detection and scene classification features, which serves to estimate image-wise performance of using the object detection and scene classification information. This baseline requires observations in a labelled scene for training. We use Random Forests [30] as the classification method, trained on images from the source scene. The second baseline we consider is non-regularized Weighted Nonnegative Matrix Factorization by augmenting the target matrix $\mathbf{R}$ with the object detection and scene classification features for each location. This baseline does not explicitly enforce the similarity that the regularized framework does, thus, we expect it to not perform as well as our framework. The third baseline we consider is to build AMs from the back-projected object detections by directly associating each detection category with an activity category.

We use the Office Flr. A demonstration and appearance data as input and evaluate the performance by applying the learned model to each of the other scenes. These results (Table 2.3) illustrate that our method's AM predictions outperform the baselines in $\frac{13}{16}$ cases, and that the appearance information is capitalized upon the most by our method. We find that scene classification is particularly beneficial to performance, a phenomenon for which we present two hypothesized factors: 1) as shown in [252] "object detectors emerge in deep scene CNNs", suggesting that the Scene Classification features subsume the cues present in the object detector features, and 2) due to localization noise, correlations between localized activities and localized objects are not as strong, and can serve to introduce noise to the Spatial+Scene Classification kernel combination when this object information is integrated.

Overall, we find that our model harnesses the power of activity observations in concert with the availability of rich scene classification and object detection information to estimate the functionality of environments both with and without activity observations. See the Supplementary Material for additional novel scene experiments.

2.4 Action Maps for Localization

We demonstrate a proof-of-concept application of Action Maps to the task of localization. Intuitively, by leveraging the "where an activity can be done" functional-spatial information from Action Maps, along with "what activity has been done" functional information from activity detection, the user's spatial location is constrained to be in one of several areas. We localize activity sequences in each 2D map based on the combination of predicted action locations from the Action Map, and observed actions in each frame. In Figure 2.8, we show the spatial discrepancy in grid cells between the K-best AM location guesses decreases. Thus, an Action Map can be used to localize a person with observations of their activity.

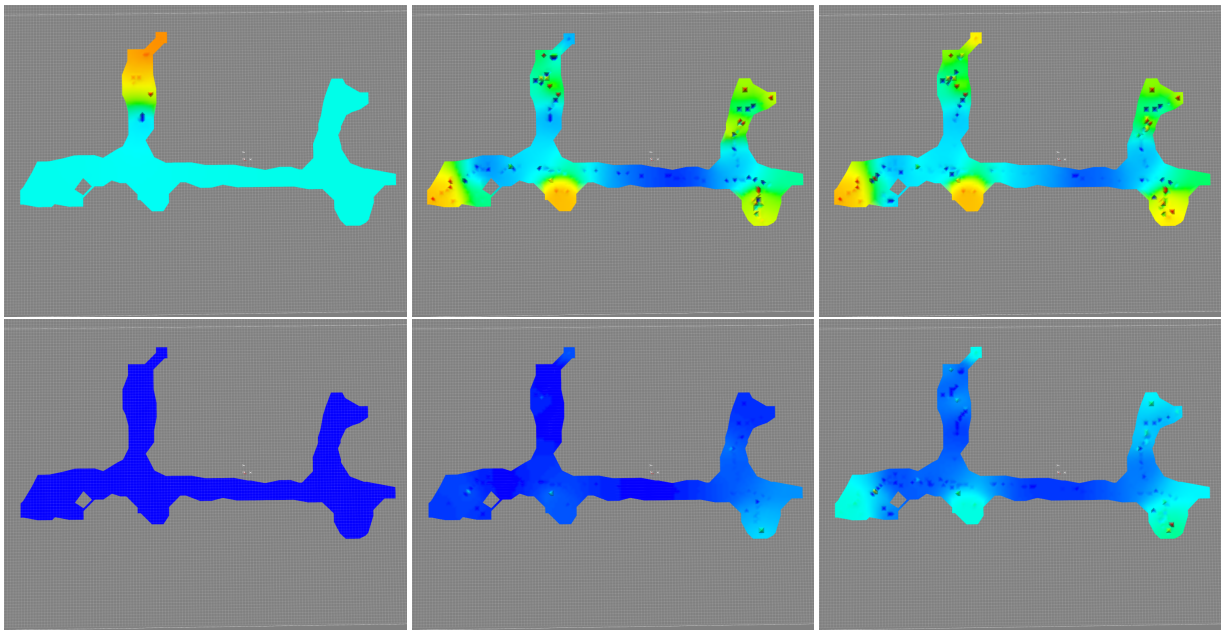


Figure 2.7: 'Sit' (top row) and 'Type' (bottom row) AMs as the amount of observed data increases on Office Flr. A. The columns stand for 10%, 80%, and 100% of the data.

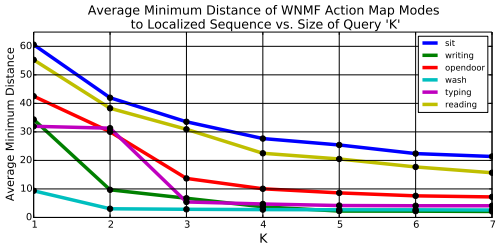


Figure 2.8: Localizing with an Action Map and observed activities. Activities that are more specialized are localized with less guesses.

2.5 Conclusion

We have demonstrated a novel method for generating functional maps of uninstrumented common environments. Our model jointly considers scene appearance and functionality while consolidating evidence from the natural vantage point of the user, and is able to learn from a user's demonstrations to make predictions of functionality of less explored and completely novel areas. Finally, our proof-of-concept application hints at the breadth of future work that can exploit the rich spatial and functional information present in Action Maps. These can be thought of as forecasting a *one-step* future as a discrete action, as was shown in Fig. 2.1. Action maps are a fairly limited way to predict behavior. They only modelled behavior as a one-off occurrence. The next step is to model sequences of behavior.

Chapter 3

Forecasting Action Trajectories with Online Inverse Reinforcement Learning

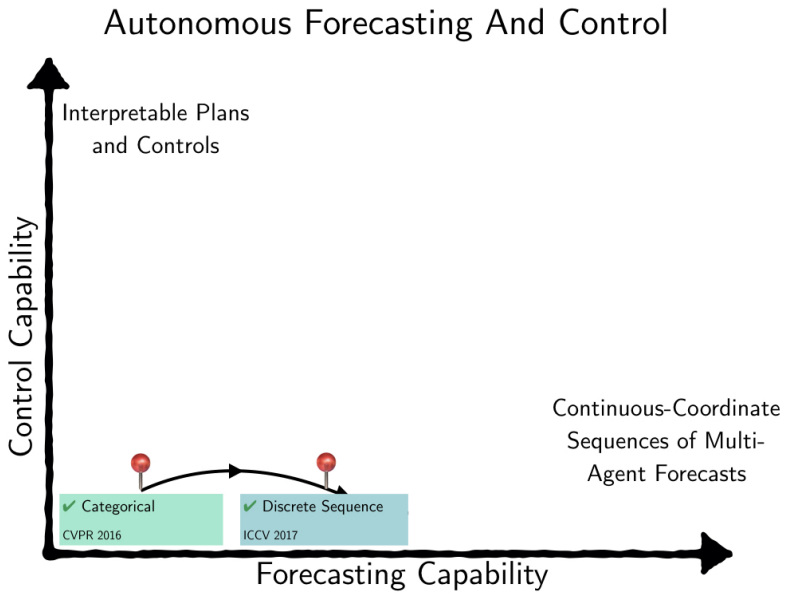


Figure 3.1: This chapter focuses on building a multi-step categorical forecasting approach.

3.1 Introduction

Our long-term aim is to develop an AI system that can learn about a person's intent and goals by continuously observing their behavior. Towards this goal, we propose an online Inverse Reinforcement Learning (IRL) technique to learn a decision-theoretic human activity model from video captured by a wearable camera. The approach we will describe explicitly models *discrete sequences* of behavior. Its situation within our framework is depicted in Fig. 3.1.

The use of a wearable camera is critical to our task, as human activities must be observed up close and across large environments. Imagine a person's daily activities—perhaps they are at home today, moving about, completing tasks. Perhaps they are a scientist that conducts a long series of experiments across various stations in a laboratory, or they work in an office building where they

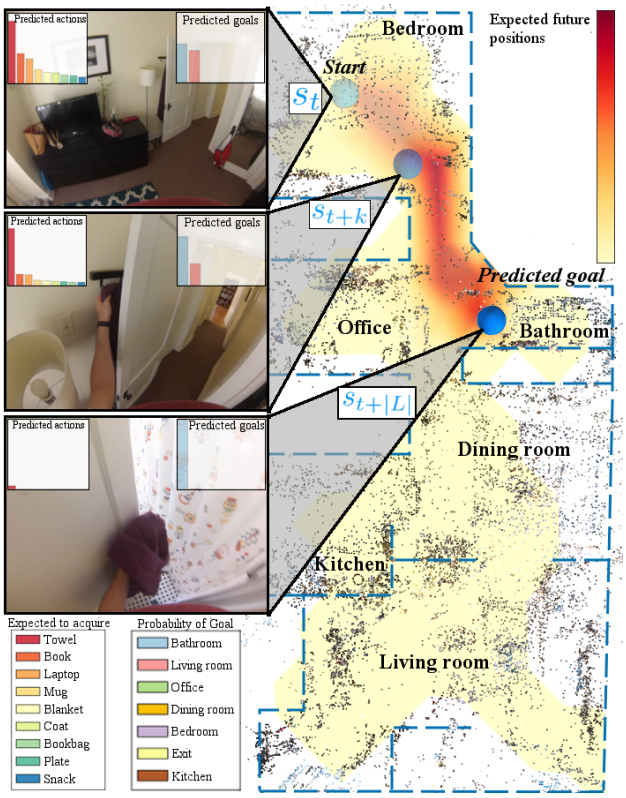


Figure 3.2: **Forecasting future behavior from first-person video.** Overhead map shows likely future goal states. s_i is user *state* at time i. Histogram insets display predictions of user's long-term semantic goal (inner right) and acquired objects (inner left).

(a) Lab environment

(b) Home 1 environment

Figure 3.3: Sparse SLAM points (3.3a) and offline dense reconstruction (3.3b) using [60] for two of our dataset environments.

walk about their floor, get coffee, *etc*. As people tend to be very mobile, a wearable camera is ideal for observing a person's behavior.

Since our task is to continuously learn human behavior models from observed behavior captured with a wearable camera, our task is best described as an online IRL problem. The problem is an *inverse* Reinforcment Learning problem because the underlying reward or cost function of the person is unknown. We must infer it along with the policy from the demonstrated behaviors. Our task is also an *online learning problem*, because our algorithm must continuously learn as a part of a life-long process. From this perspective, an online learning approach is required to learn effectively over time.

We present an algorithm that *incrementally learns spatial and semantic intentions* (where you will go and what you will do) of a first-person camera wearer. By tracking the goals a person achieves, the algorithm builds a set of possible futures. At any time, the user's future is predicted among this set of goals. We term our algorithm "Discovering Agent Rewards for K-futures Online" (DARKO), as it learns to associate rewards with semantic states and actions from demonstrations to predict among K possible goals.

To the best of our knowledge, we present the first application of ideas from online learning theory and inverse reinforcement learning to the task of continuously learning human behavior models with a wearable camera. Our proposed algorithm is distinct from traditional IRL problems as we jointly discover transitions, goals, and the reward function of the underlying Markov Decision Process model. Our proposed human behavior model also goes beyond first-person trajectory forecasting by predicting future human activities that can happen outside the immediate field of view and far into the future.

3.2 Related Work

We extend a portfolio of visual sensing algorithms (SLAM, stop detection, scene classification, action and object recognition) in concert with a decision-theoretic approach to model and forecast behavior online. Several topics of related work cover components of our approach. We will summarize each in the following and relate them to our approach.

3.2.1 First-person vision (FPV)

Wearable cameras have been used for various human behavior understanding tasks [53, 115, 119, 155, 187] because they give direct access to detailed visual information about a person's actions. Leveraging this feature of FPV, recent work has shown that it is possible to predict where people will look during actions [119] and how people will use the environment [169]. Previous first-person vision approaches are narrower in their scope of modeling relative to our approach: they are batch models (learned once), and generally do not attempt to model predictions with respect to information outside of the camera's frame of view.

3.2.2 Decision-Theoretic Modeling

Given agent demonstrations, the task of *inverse reinforcement learning* (IRL) is to recover a *reward function* of an underlying Markov Decision Process (MDP) [3]. IRL has been used to model taxi driver behavior [257] and forecast pedestrian trajectories [101, 258]. In contrast to these approaches, we go beyond physical trajectory forecasting by reasoning over future object interactions and both uncovering and forecasting future goals in terms of scene types.

3.2.3 Online Learning Theory

The theory of learning to making optimal predictions from streaming data is well-studied [198], but its fruits are seldom applied to computer vision (*e.g.* [33], [177]), compared to the more prevalent application of supervised learning. However, we believe online learning theory and practice will gain traction to confront the challenges of ever-increasing visual data. In the context of IRL, online learning theory was used in [163], an imitation learning framework similar to IRL in its recovery of costs, to analyze performance of a mobile robot path planner.

3.2.4 Forecasting

Our task fits in the broad category of *forecasting* with visual information. There are two primary research thrusts in the forecasting category: *trajectory* and *behavior* forecasting. The former attempts to predict an agent's future behavior in 2D or 3D coordinates. The latter attempts to predict the behavior of agents in terms of categories of low-level activities. However, the majority of the work along these veins does not model or capitalize upon two key aspects of the problem: *uncertainty* in the future behaviors (which requires predicting either a distribution or multiple samples), and the *goal-driven nature* of agents. Agents generally take low-level motions and actions in order to achieve goals. Our work explicitly forecasts and models goals with uncertainty.

3.2.4.1 Trajectory Forecasting

Physical trajectory forecasting has received much attention from the vision community. The task is to predict the future spatial coordinates of an agent. Multiple human trajectory forecasting from a surveillance camera was investigated by [127]. Other trajectory forecasting approaches use demonstrations observed from a bird's-eye view; [246] infers latent goal locations and [9] employ LSTMs to jointly reason about trajectories of multiple humans. In [206], the model forecasted short-term future trajectories of a first-person camera wearer by retrieving the nearest neighbors from a dataset of first-person trajectories under an obstacle-avoidance cost function, with each trajectory representing predictions of where the user will move in view of the frame; in [209], a similar model with learned cost function is extended to multiple users.

A drawback of predicting future spatial coordinates of an agent is an interpetability gap. If a person is tasked with predicting and communicating some agent's future, they will not produce a list of high-fidelity spatial coordinates — instead they will frame their prediction in the *semantics* of activity. Methods that forecast interpretable futures grounded in categories of behavior are known as activity forecasting.

3.2.4.2 Activity Forecasting

Activity forecasting methods typically treat the problem of predicting future behaviors as a classification task. In [84, 186], the tasks are to recognize an unfinished event or activity. In [84], the model predicts the onset for a single facial action primitive, *e.g.* the completion of a smile, which may take less than a second. Similarly, [186] predicts the completion of short human to human interactions. In [107], a hierarchical structured SVM is employed to forecast actions about a second in the future, and [232] demonstrates a semi-supervised approach for forecasting human actions a second into the future. Other works predict actions several seconds into the future [34, 104, 118, 235]. In contrast, we focus on high-level transitions over a sequence of future actions that may occur outside the frame of view, and take a longer time to complete (in our dataset, the mean time to completion is 21.4 seconds).

3.3 Online IRL with DARKO

Our goal is to forecast the future behaviors of a person from a continuous stream of video captured by a wearable camera. Given a continuous stream of FPV video, our approach extracts a sequence of state variables $\{s_1, s_2, \dots\}$ using a portfolio of visual sensing algorithms (*e.g.*., SLAM, stop detection, scene classification, action and object recognition). In an online fashion, we segment this state sequence into episodes (short trajectories) by discovering terminal goal states (*e.g.*., when a person stops). Using the most recent episode, we adaptively solve the inverse reinforcement

learning problem using online updates. Solving the IRL problem in an online fashion means that we incrementally learn the underlying decision process model.

3.3.1 First-Person Behavior Model

A Markov Decision Process (MDP) is commonly used to model the sequential decision process of a rational agent. In our case, we use it to describe the activity of a person with a wearable camera. In a typical reinforcement learning problem, all elements of the MDP are assumed to be known and the task is to estimate an optimal policy $\pi(a|s)$, that maps a state s to an action a, by observing rewards. In our novel online inverse formulation, the transition function, reward function, policy, and goal states are unknown and must be inferred as new video data arrives. Formally, our MDP is defined as:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R_\theta).$$

3.3.2 States

$\mathcal{S}$ is the state space: the set of states an agent can visit. In our online formulation, $\mathcal{S}$ is initially empty, and must be expanded as new states are discovered. We define a state s as a vector that includes the location of the person (3D position), the last place the person stopped (a previous goal state), and information about any object that the person might be holding. Formally, a state $s \in \mathcal{S}$ is denoted as:

$$s = [x, y, z, o_1 \ldots, o_{|\mathcal{O}|}, h_1, \ldots h_{|\mathcal{K}|}].$$

The triplet $[x, y, z]$ is a discrete 3D position. To obtain the position, we use a monocular visual SLAM algorithm [135] to localize the agent in a continuously built map.

The vector $o_1 \ldots, o_{|\mathcal{O}|}$ encodes any objects that the person is currently holding. We include this information in the state vector because the objects a user acquires are strongly correlated to the intended activity [53]. $o_j = 1$ if the user has object j in their possession and zero otherwise. $\mathcal{O}$ is a set of pre-defined objects available to the user. $\mathcal{K}$ is a set of pre-defined scene types available to the user, which can be larger than the true number of scene types. The vector $h_1, \ldots h_K$ encodes the last scene type the person stopped. Example scene types are `kitchen` and `office`. $h_i = 1$ if the user last arrived at scene type i and is zero otherwise.

3.3.3 Goals

We also define a special type of state called a *goal state* $s \subset \mathcal{S}_g$, to denote states where the person has achieved a goal. One of our methods assumes that when a person stops for a certain period of time, their location in the environment is a goal. This method detect goal states by using a velocity-based stop detector. Whenever a goal state is encountered, the sequence of states since the last goal state to the current goal state is considered a completed episode ξ. The set of goals states $\mathcal{S}_g \subset \mathcal{S}$ expands with each detection. We explain later how $\mathcal{S}_g$ is used to perform goal forecasting.

3.3.4 Actions

$\mathcal{A}$ is the set of actions. $\mathcal{A}$ can be decomposed into two parts: $\mathcal{A} = \mathcal{A}_m \cup \mathcal{A}_c$. The act of moving from one location in the environment to another location is denoted as $a_m \in \mathcal{A}_m$. Like $\mathcal{S}$, $\mathcal{A}_m$ must be built incrementally. The set $\mathcal{A}_c$ is the set of possible `acquire` and `release` actions of each object: $\mathcal{A}_c = \{\texttt{acquire}, \texttt{release}\} \times \mathcal{O}$. The act of releasing or picking up an object is denoted as $a_c \in \mathcal{A}_c$. Each action a_c must be detected. We do so with an image-based first-person action classifier. More complex approaches could improve performance [126].

3.3.5 Transition Function

The transition function $T : (s, a) \mapsto s'$ represents how actions move a person from one state to the next state. T is constructed incrementally as new states are observed and new actions are performed. In our work, T is built by keeping a table of observed (s, a, s') triplets, which describes the connectivity graph over the state space. More advanced methods could also be used to infer more complex transition dynamics [214, 238].

3.3.6 Reward Function

$R(s, a; \theta)$ is an instantaneous reward function of action a at state s. The standard and simplest parametric model of R is the inner product between a vector of features $f(s, a)$ and a vector of weights θ. We adopt this standard model, however, different representations could be employed [244]. The reward function is essential in value-based reinforcement learning methods (in contrast to policy search methods) as it is used to compute the policy $\pi(a|s)$. In the maximum entropy setting, the policy is given by $\pi(a|s) \propto e^{Q(s,a)-V(s)}$, where the value functions $V(s)$ and $Q(s, a)$ are computed from the reward function by solving the Bellman equations [257]. In our context, we learn the reward function online.

Intuitively, we would like the features f of the reward function to incorporate information such as the position in an environment or objects in possession, since it is reasonable to believe that the goal of many activities is to reach a certain room or to retrieve a certain object. To this end, we define the features of the reward to mirror the information already contained in the state s_t: the position, previous scene type, and objects held. To be concrete, the feature vector $f(s, a)$ is the concatenation of the 3-d position coordinates $[x, y, z]$, a K-dimensional indicator vector over previous goal state type and a $|\mathcal{O}|$-dimensional indicator vector over held objects. We also concatenate a $|\mathcal{A}_c|$-dimensional indicator vector over actions $a_c \in \mathcal{A}_c$.

3.3.7 The DARKO Algorithm

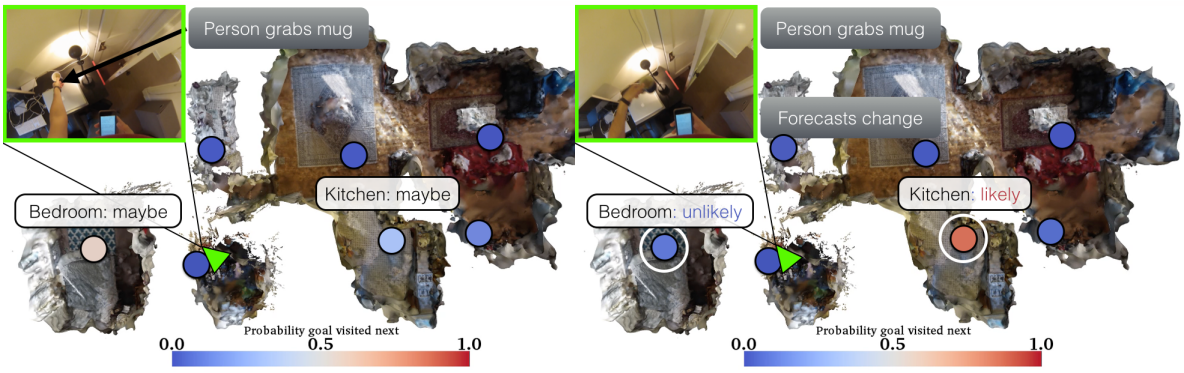
Algorithm 1 DARKO(SLAM, ACTDET, GOALDET)

$s \leftarrow \mathbf{0}, \theta = \mathbf{0}, \mathcal{S} = \{\}, \mathcal{S}_g = \{\}$, T.INIT, $\xi = []$
while True **do**
  frame $\leftarrow$ NEWFRAME
  $[x, y, z] \leftarrow$ SLAM.TRACKframe
  $a \leftarrow$ ACTDET$[x, y, z]$, frame
  $\xi \leftarrow \xi \oplus (s, a), \mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
  T.EXPANDs, a, $s \leftarrow T(s, a)$
  ▶ Goal forecasting, trajectory forecasting, ...
  is_goal $\leftarrow$ GOALDET(s, frame, $\mathcal{S}_g$)
  if is_goal **then**
    $\mathcal{S}_g \leftarrow \mathcal{S}_g \cup \{s\}$
    $\pi, \theta \leftarrow$ ONLINEIRLθ, $\mathcal{S}$, T, ξ, $\mathcal{S}_g$
    $s \leftarrow T(s, a = \text{at\_goal}), \xi = []$
  end if
end while

We now describe our proposed algorithm for incrementally learning all MDP parameters, most importantly the reward function, given a continuous stream of first-person video (see DARKO in

Algorithm 2 ONLINEIRL($\theta, \mathcal{S}, T, \xi, \mathcal{S}_g; \lambda, B$)

1: $\overline{f}_i = \sum_{(s,a)\in\xi} f(s,a)$
2: ▶ Compute $R(s,a;\theta)\ \forall s \in \mathcal{S}, a \in \mathcal{A}$
3: $\pi \leftarrow$ SOFTVALUEITERATION$R, \mathcal{S}, \mathcal{S}_g, T$
4: $\hat{f}_i \leftarrow E_\pi\left[f(s,a)\right]$
5: $\theta \leftarrow \text{proj}_{\|\theta\|_2 \le B}(\theta - \lambda(\overline{f}_i - \hat{f}_i))$
6: **return** π, θ



(a) Goal Posterior Before Mug Acquired

(b) Goal Posterior After Mug Acquired

Figure 3.4: **Goal Posterior Change Visualization:** Goal posteriors for two frames are visualized in the Home 1 environment. The person's location is in green, images from the camera are inset at top left, and goal posteriors are colored according to the above colormaps. Before grabbing the mug (Figure 3.4a), DARKO forecasts roughly equivalent probability to bedroom and kitchen. After the user grabs the mug (Figure 3.4b), DARKO correctly predicts the user is likeliest to go to the kitchen.

Algorithm 1). The procedure begins by initializing s, reward parameters θ, empty state space $\mathcal{S}$, goal space $\mathcal{S}_g$, transition function T, and current episode ξ.

3.3.8 State Space Update

Image frames are obtained from a first-person camera (the NEWFRAME function), and SLAM is used to track the user's location (lines 3 and 4). An image-based action detection algorithm, ACTDET, detects hand-object interactions a_c and decides movements a_m as a function of current and previous position. While we provide an effective method for ACTDET, our focus is to integrate (rather than optimize) its outputs. Lines 6 and 7 show how the trajectory is updated and MDP parameters of state space and transition function are expanded. Line 8 represents a collection of generalized forecasting tasks (see Section 3.4.4), such as the computation of future goal posterior and trajectory forecasting. An example of our primary forecasting task, goal forecasting, is illustrated in Figure 3.4. Our method uses the current environment and policy models to forecast a distribution over the person's goals, described in more detail later. In the example, the distribution is initially uncertain about the person's goal, and then becomes confident the person will go to the goal in the kitchen, given the evidence that they are in the office and recently acquired a mug.

3.3.9 Goal Detection

The GOALDET procedure denotes detecting new goals or recognizing previous goals. One of our GOALDET implementations is a stop-detection algorithm. This uses the camera velocity computed from SLAM (Line 9). If a goal state has been detected, that terminal state is added to the set of goal states $\mathcal{S}_g$. The detection of a terminal state also marks the end of an episode ξ. The previous goal state type is also updated for the next episode. Again, while we provide two effective method for GOALDET, our focus is to integrate (rather than optimize) its outputs.

3.3.10 Online Inverse Reinforcement Learning

With the termination of each episode ξ, the reward function R and corresponding policy π are updated via the reward parameters θ (Line 12). The parameter update uses a sequence of demonstrated behavior via the episode ξ, and the current parameters of the MDP. More specifically, ONLINEIRL (Algorithm 2) performs online gradient descent on the likelihood under the maximum entropy distribution by updating current parameters of the reward function. The gradient of the loss can be shown to be the difference between the feature counts of the expert, $\bar{f}$, and feature counts of the policy $\hat{f}$. Computing the gradient requires solving the soft value iteration algorithm of [255]. We include a projection step to ensure $\|\theta\|_2 \leq B$.

To the best of our knowledge, this is the first work to propose an online algorithm for maximum entropy IRL in the streaming data setting. Following the standard procedure for ensuring good performance of an online algorithm, we analyze our algorithm in terms of the regret bound. The regret $\mathcal{R}_t$ of any online algorithm is defined as:

$$\mathcal{R}_t(\{\theta_i\}_{i=0}^t) = \sum_{i=0}^{t} l_i(\theta_i) - \min_{\theta^*} \sum_{i=0}^{t} l(\theta^*).$$

The regret is the cumulative difference between the performance of the online model using current parameter θ versus the best hindsight model using the best parameters θ^*. The loss l_t is a function of the t'th demonstrated trajectory, and measures how well the model explains the trajectory. In our setup, the loss function is defined as $l_i(\theta_i; \xi_i) = -\frac{1}{|\xi_i|}\sum_{j=0}^{|\xi_i|} \log \pi_\theta(a_j|s_j)$. This loss function must be convex wrt. θ for the following proof to hold. When the MDP dynamics are deterministic, $-\log P(\xi|\theta)$ is convex in θ [257] and $-\log P(\xi|\theta) = -\log P(s_0) - \sum_{j=0}^{|\xi_i|} \log \pi_\theta(a_j|s_j)$. Thus l_i is convex in θ.

Theorem 1 (ONLINEIRL is no-regret)**.** *Let $\hat{f}, \bar{f} \in [0,1]^d$, $||\theta||_2 \leq B$. The regret of Algorithm 2 satisfies $\mathcal{R}_t \leq 2B\sqrt{2td}$.*

Proof. By Equation 2.5 of [198], the regret of online gradient descent on convex losses l_t is bounded:

$$\mathcal{R}_t \leq \frac{1}{2\lambda}\|\theta\|_2^2 + \lambda \sum_{i=1}^{t} \|\nabla_{\theta_t}\|_2^2, \tag{3.1}$$

where λ is the learning rate and $\nabla_{\theta_t} = \partial_\theta l_t$. We will employ bounds on $\|\theta\|_2^2, \|\nabla_{\theta_t}\|_2^2$, and a minimizing choice of λ to prove the result. Writing the general gradient in terms of the expected features (and omitting the subscript t):

$$\|\nabla_\theta\|_2^2 = \|\bar{f} - \hat{f}\|_2^2 = \bar{f}^T\bar{f} + \hat{f}^T\hat{f} - 2\bar{f}^T\hat{f} \tag{3.2}$$

Using:

$$\begin{aligned} 0 \leq \|x-y\|_2^2 &= x^T x + y^T y - 2x^T y \\ 2x^T y &\leq x^T x + y^T y \\ 2(-x)^T y &\leq (-x)^T(-x) + y^T y \\ -2x^T y &\leq x^T x + y^T y, \\ \therefore -2\bar{f}^T \hat{f} &\leq \bar{f}^T \bar{f} + \hat{f}^T \hat{f}, \text{ (Setting } x = \bar{f}, y = \hat{f}) \end{aligned}$$

then Equation 3.2 becomes:

$$\begin{aligned} \|\nabla_\theta\|_2^2 &\leq \bar{f}^T \bar{f} + \hat{f}^T \hat{f} + \bar{f}^T \bar{f} + \hat{f}^T \hat{f} = 2\bar{f}^T \bar{f} + 2\hat{f}^T \hat{f} \\ &\leq 4d. \text{ (Since } \bar{f}, \hat{f} \in [0,1]^d) \end{aligned} \tag{3.3}$$

Thus, using Equation 3.3 in Equation 3.1, and that the projection step of θ (constraining the set of θ to be the convex ball with radius B) ensures $\|\theta\|_2 \leq B$:

$$\mathcal{R}_t \leq \frac{B^2}{2\lambda} + \lambda \sum_{i=1}^{t} 4d = \frac{B^2}{2\lambda} + 4\lambda t d.$$

With the minimizing choice of $\lambda = \frac{B}{2\sqrt{2td}}$,

$$\mathcal{R}_t \leq B\sqrt{2td} + \frac{2Btd}{\sqrt{2td}} = 2B\sqrt{2td}$$

□

Therefore, our algorithm is no-regret ($\lim_{t\to\infty} \frac{\mathcal{R}_t}{t} = 0$), which guarantees the quality of our continuous forecasting model with respect to one learned in batch. Our experiments also confirm this property.

3.4 Generalized Activity Forecasting

Without Line 8, Algorithm 1 only describes our online IRL process to infer the reward function. In order to make incremental predictions about the person's future behaviors online, we can leverage the current MDP and reward function. An important function which lays the basis for predicting future behaviors is the state visitation function, denoted D. We now show how D can be modified to perform generalized queries about future behavior.

3.4.1 State Visitation Function D

Using the current estimate of the MDP and the reward function, we can compute the policy of the agent. Using the policy, we can forward simulate a distribution of all possible futures. This distribution is called the state visitation distribution [255]. More formally, the posterior expected count of future visitation to a state s_x can be defined as

$$D_{s_x|\xi_{0\to t}} \triangleq \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\left[\sum_{\tau=t+1}^{T} I(s_\tau = s_x)\right]. \tag{3.4}$$

This quantity represents the agent's expectation to visit each state in the future given the partial trajectory. $\xi_{0\to t}$ indicates a partial trajectory starting at time 0 and ending at time t. The expectation is taken under the maximum causal entropy distribution, $P(\xi_{t+1\to T}|\xi_{0\to t})$, which gives the probability of a future trajectory given the current trajectory. I is the indicator function, which counts agent visits to s_x. Equation 3.4 is estimated by sampling trajectories from $\pi_\theta(a|s)$, and also is employed in Algorithm 2, line 4 to compute $\hat{f}$.

3.4.2 Activity Forecasting with State Subsets

In this work, we extend the idea of state visitations to a single state s_x to a more general *subset of states* $\mathcal{S}_p$. While a generalized prediction task was not particularly meaningful in the context of trajectory prediction [101, 257], predictions over a subset of states now represents semantically meaningful concepts in our proposed MDP. By using the state space representation of our first-person behavior model, we can construct subsets of the state space that have interesting semantic meaning, such as "having an object o_i" or "all states closest to goal k with $\mathcal{O}_j$ set of objects."

Formally, we define the expected count of visitation to a subset of states $\mathcal{S}_p$ satisfying some property p:

$$D_{\mathcal{S}_p|\xi_{0\to t}} \triangleq \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\left[\sum_{\tau=t+1}^{T} I(s_\tau \in \mathcal{S}_p)\right] \tag{3.5}$$

$$= \sum_{s_x\in\mathcal{S}_p} \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\left[\sum_{\tau=t+1}^{T} I(s_\tau = s_x)\right]$$

$$= \sum_{s_x\in\mathcal{S}_p} D_{s_x|\xi_{0\to t}}. \tag{3.6}$$

Equation 3.6 is essentially marginalizing over the state subspace of Equation 3.4.

3.4.3 Forecasting Trajectory Length

Leveraging Equation 3.6, we present a method to predict the length of the future trajectory. Formally, we can denote the expected trajectory length:

$$\hat{\tau}_{\xi_{t+1\to T}|\xi_{0\to t}} \triangleq \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\,|\xi_{t+1\to T}| \tag{3.7}$$

Consider evaluating $D_{\mathcal{S}_p|\xi_{0\to t}}$ from Equation 3.6 by setting $\mathcal{S}_p = \mathcal{S}$, that is, by considering the expected future visitation count to the entire state space. Then,

$$D_{\mathcal{S}|\xi_{0\to t}} = \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\left[\sum_{\tau=t+1}^{T} I(s_\tau \in \mathcal{S})\right]$$

$$= \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\left[\sum_{\tau=t+1}^{T} 1\right]$$

$$= \mathbb{E}\,|\xi_{t+1\to T}| = \hat{\tau}_{\xi_{t+1\to T}|\xi_{0\to t}} \tag{3.8}$$

where $|\xi|$ indicates the number of states in trajectory ξ.

3.4.4 Future Goal Forecasting

As previously described, we wish to predict the final goal of a person's action sequence. For example, if I went to the study to pick up a cup, how likely am I to go to the kitchen versus the living room? This problem can be posed as solving for the MAP estimate of $P(g|\xi)\forall g \in \mathcal{S}_g$, the posterior over goals. It describes *what goal the user seeks given their current trajectory*, defined as:

$$P(g|\xi_{0\to t}) \propto P(g)e^{V_{s_t}(g)-V_{s_0}(g)}, \tag{3.9}$$

where $V_{s_i}(g)$ is the *value* of g with respect to a partial trajectory that ends in s_i. The value function is computed from the learned reward function, see [255] for details. Notice that the likelihood term is exponentially proportional to the value difference between the start state s_0 and the current state s_t. In this way, the likelihood encodes the progress made towards a goal g in terms of the value function. This progress is a function of the current spatial and activity trajectory, and encodes a representation of how the person tends to behave. We illustrate that our model learns intuitive values in Figure 3.4: the model becomes more certain in the person's future goal after receiving additional evidence (a detection that they acquired a mug).

3.4.5 Derivation of Other Forecasting Tasks

Our use of the Maximum Entropy Inverse Reinforcement Learning framework enables us to construct additional meaningful inference tasks. While we do not evaluate these tasks, we provide them to illustrate how our approach can be efficiently extended.

3.4.5.1 Action-Subspace Visitation

To derive the action-subspace visitation, we first use the posterior expected visitation count of performing an action a_y immediately after arriving at a state s_x is given in Equation 3.10, from [255].

$$D_{a_y,s_x|\xi_{0\to t}} \triangleq \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}\left[\sum_{\tau=t+1}^{T} I(s_\tau = s_x, a_\tau = a_y)\right] \tag{3.10}$$

$$D_{a_y,s_x|\xi_{0\to t}} = \pi(a_y|s_x)D_{s_x|\xi_{0\to t}} \tag{3.11}$$

Our definition of the posterior expected action subspace visitation count is given in Equation 3.12. This expresses the future expectation the user will perform an action a_y while in a subspace S_p, given their current trajectory $\xi_{0\to t}$. Hereafter, we denote $\mathbb{E} \triangleq \mathbb{E}_{P(\xi_{t+1\to T}|\xi_{0\to t})}$ for brevity.

$$\begin{aligned} D_{a_y,\mathcal{S}_p|\xi_{0\to t}} &\triangleq \mathbb{E}\left[\sum_{\tau=t+1}^{T} I(s_\tau \in \mathcal{S}_p)I(a_\tau = a_y)\right] \\ &= \mathbb{E}\left[\sum_{s_x\in\mathcal{S}_p}\sum_{\tau=t+1}^{T} I(s_\tau = s_x, a_\tau = a_y)\right] \\ &= \sum_{s_x\in\mathcal{S}_p}\mathbb{E}\left[\sum_{\tau=t+1}^{T} I(s_\tau = s_x, a_\tau = a_y)\right] \\ &= \sum_{s_x\in\mathcal{S}_p} D_{a_y,s_x|\xi_{0\to t}} = \sum_{s_x\in\mathcal{S}_p}\pi(a_y|s_x)D_{s_x|\xi_{0\to t}}. \end{aligned} \tag{3.12}$$

Thus, the posterior expected action subspace visitation is straightforward to compute with $D_{s_x|\xi_{0\to t}}$. Various inference tasks can be constructed by choosing a_y and $\mathcal{S}_p$ appropriately.

3.4.5.2 Joint Action-State Subspace Visitation

We additionally derive the expected transition count from a subspace of states to a subspace of actions. This expresses the expectation that the user will perform an $a_y \in \mathcal{A}_y$ from a $s_x \in \mathcal{S}_p$. It is defined as:

$$
\begin{aligned}
D_{\mathcal{A}_y,\mathcal{S}_p|\xi_{0\to t}} &\triangleq \mathbb{E}\left[\sum_{\tau=t+1}^{T} I(s_\tau \in \mathcal{S}_p) I(a_\tau \in \mathcal{A}_y)\right] \\
&= \mathbb{E}\left[\sum_{\substack{a_y\in\mathcal{A}_y \\ s_x\in\mathcal{S}_p}} \sum_{\tau=t+1}^{T} I(s_\tau = s_x, a_\tau = a_y)\right] \\
&= \sum_{\substack{a_y\in\mathcal{A}_y \\ s_x\in\mathcal{S}_p}} \mathbb{E}\left[\sum_{\tau=t+1}^{T} I(s_\tau = s_x, a_\tau = a_y)\right] \\
&= \sum_{\substack{a_y\in\mathcal{A}_y \\ s_x\in\mathcal{S}_p}} D_{a_y,s_x|\xi_{0\to t}} = \sum_{\substack{a_y\in\mathcal{A}_y \\ s_x\in\mathcal{S}_p}} \pi(a_y|s_x) D_{s_x|\xi_{0\to t}}.
\end{aligned}
\tag{3.13}
$$

Again, computing this quantity is straightforward with $D_{s_x|\xi_{0\to t}}$. By marginalizing $D_{s_x|\xi_{0\to}}$ over various action and state subspaces that have semantic meaning, different inference quantities can be expressed and computed. The construction and evaluation of new and richer inference quantities is a promising direction for future work.

3.5 Experiments

We first present the dataset we collected. Then, we discuss our methods for goal discovery and action recognition. To reiterate, our focus is not to engineer these methods, but instead to make intelligent use of their outputs in DARKO for the purpose of behavior modeling. We compare DARKO's performance versus several baselines on the task of goal forecasting, and show DARKO's performance is superior. Next, we analyze DARKO's performance under less noisy conditions, to illustrate how it improves when provided with more robust goal discovery and action detection algorithms. Then, we illustrate DARKO 's empirical no-regret performance, which further shows it is an efficient online learning algorithm. Finally, we present trajectory length forecasting results, and find that our length forecasts exhibit low median error.

3.5.1 First-Person Continuous Activity Dataset

We collected a dataset of sequential goal-seeking behavior in several different environments such as home, office and laboratory. The users recorded a series of activities that naturally occur in each scenario. Each user helped design the script they followed, which involved their prior assumptions about what objects they will use and what goal they will seek. An example direction from a script is "obtain a snack and plate in kitchen, eat at dining room table."

Table 3.1: **Scene types available in each environment.**

Environment	Scene Type Set
Home 1	{bathroom, bedroom, exit, dining room, kitchen, living room, office}
Home 2	{bathroom, bedroom, exit, dining room, kitchen, living room, office, television stand}
Office 1	bathroom, exit, kitchen, lounge, office, printer station, water fountain}
Office 2	{bathroom, exit, kitchen, lounge, office, printer room, water fountain}
Lab 1	{cabinet stand, exit, gel room, lab bench 1, lab bench 2, refrigeration room}

Table 3.2: **Objects available in each environment.**

Environment	Object Set
Home 1	{bookbag, book, blanket, coat, laptop, mug, plate, snack, towel}
Home 2	{bookbag, book, blanket, coat, guitar, laptop, mug, plate, remote, snack, towel}
Office 1	{bookbag, textbook, bottle, coat, laptop, mug, paper, plate, snack}
Office 2	{bookbag, textbook, bottle, coat, laptop, mug, paper, plate, snack}
Lab 1	{beaker, coat, plate, pipette, tube}

Table 3.3: **Labels example:** A small snippet of ground truth labels for Home 1.

Frame Index	6750	6900	7200
Action/Arrival	Release Coat	Acquire Bookbag	Arrive Office
Frame Index	7400	7630	7700
Action/Arrival	Acquire Mug	Arrive Kitchen	Release Mug

Users wore a hat-mounted Go-Pro Hero camera with $94°$ vertical, $123°$ horizontal FOVs. Our dataset is comprised of 5 user environments, and includes over 250 actions with 19 objects, 17 different scene types, at least 6 activity goals per environment, and about 200 high-level activities (trajectories). In each environment, the user recorded 3–4 long sequences of high-level activities, where each sequence represents a full day of behavior. Our dataset represents over 15 days of recording. The scenes present in each environment are shown in Table 3.1, and the objects available in each environment are shown in Table 3.2.

For evaluation, all ground truth labels of objects (*e.g.* cup, backpack), actions (*i.e.* acquire, release) and goals (*e.g.* kitchen, bedroom) were first manually annotated. A goal label correspond to *when* a high-level direction was completed, and *in which scene* it was completed, *e.g.* (dining room, time=$65s$). An action label indicates when an activity was performed, *e.g.* (acquire, cup, time=$25s$). A small example of labels is shown in Table 3.2.

Figure 3.5: **Goal forecasting examples:** A temporal sequence of goal forecasting results is shown in each row from left to right, with the forecasted goal icons and sorted goal probabilities inset (green: $P(g^*|\xi)$, red: $P(g_i \neq g^*|\xi)$). *Top*: the scientist acquires a sample to boil in the gel electrophoresis room. *Middle*: the user gets a textbook and goes to the lounge. *Bottom*: the user leaves their apartment.

3.5.2 Goal Discovery and Action Recognition

We describe two goal discovery methods and an action recognition method that we implemented to serve as input to DARKO. With respect to Algorithm 1, these are GOALDET and ACTDET.

3.5.3 Scene-based Goal Discovery

This model assumes that if a scene classifier is very confident in the scene type for several images frames, the camera wearer must be in a meaningful place in the environment (*i.e.*., kitchen, bedroom,

Table 3.4: **Goal Discovery and Action Recognition.** The per-scene goal discovery and action recognition accuracies are shown for our methods. A 3-second window is used around every goal discovery to compute accuracy.

Method	Home 1	Home 2	Office 1	Office 2	Lab 1
Scene Discovery	0.93	0.24	0.62	0.49	0.32
Stop Discovery	0.62	0.68	0.67	0.69	0.73
Act. Recognition	0.64	0.63	0.66	0.56	0.71

office). We use the output of a scene classifier from [253] (GoogLeNet model) on every frame from the wearable camera. If the mean scene classifier probability for a scene type is above a threshold for 20 consecutive image frames, then we add the current state s_t to the set of goals $\mathcal{S}_g$.

3.5.4 Stop-based Goal Discovery

This model assumes that when a person stops, they are at an important location. Using SLAM's 3D camera positions, we apply a threshold on velocity to detect stops in motion. When a stop is detected, we add the current state s_t to the set of goals $\mathcal{S}_g$. In Table 3.4, temporal accuracies are computed by counting detections within 3-second windows of ground truth labels as true positives; for the scene-based method, a true positive also requires the scene type to match the ground truth scene type. Stop-based discovery is more reliable across all environments, thus, we use it as our primary goal discovery method.

3.5.5 Image-based Object Recognition

We designed an object recognition approach that classifies the object the user interacts with at every temporally-labeled window. It overwrites the ground-truth object label with its detection. The approach first detects regions of `person` in each frame with [165] to focus on objects near the visible hands, which are cropped with context and fed into an image-classifier trained on ImageNet [202]. The outputs are remapped to our object set, and a final classification is produced by taking the maximum across objects. The per-action classification accuracies in Table 3.4 demonstrate the method can produce reasonable action classifications across all scenes. While imperfect, these detections serve as useful input to DARKO.

3.5.6 Goal Forecasting Performance

At every time step, our method predicts the user's goal or final destination (*e.g.*., bedroom, exit) as described in Section 3.4.4 and shown in Figure 3.5.

3.5.6.1 Baseline goal forecasting models

To understand the goal prediction reliability, we compare our approach to several baseline methods for estimating the goal posterior $P(g|\xi_{0\to t})$, where g is a goal and $\xi_{0\to t}$ is the observed state sequence up to the current time step. Each baseline requires the state tracking and goal discovery components of DARKO.

- **Uniform Model (Uniform):** This model returns a uniform posterior over possible goals $P_n(g) = 1/K_n$ known at the current episode n, defining worst case performance.

Table 3.5: **Goal Forecasting Results (Visual Detections):** Proposed goal posterior (Sec.3.4.4) achieves best $\overline{P}_{g^*}$ (mean probability of true goal).

Method	Home 1	Home 2	Office 1	Office 2	Lab 1
DARKO	**0.524**	**0.378**	**0.667**	**0.392**	**0.473**
MMED [84]	0.403	0.299	0.600	0.382	0.297
RNN	0.291	0.274	0.397	0.313	0.455
Logistic	0.458	0.297	0.569	0.323	0.348
Uniform	0.181	0.098	0.233	0.111	0.113

- **Logistic Regression Model (Logistic):** A logistic regression model $P_n(g|s_t)$ is fit to map states s_t to goals g. We used the implementation available in scikit-learn [151].
- **Max-Margin Event Detection (MMED) [84]:** A set of max-margin models $P_n(g|\phi(s_{t:t-w}))$ are trained to map features ϕ of a w-step history of state vectors $s_{t:t-w}$ to a goal score. We found the `sumL1norm` features provided with the publicly available code to perform the best, and report those best results.
- **RNN Classifier (RNN):** An RNN is trained to predict $P_n(g|\xi_{0\to t})$. We experimented with a variety of parameters and report the best results. After each goal is detected, the RNN is refit. The settings we experimented with are cell $\in$ {GRU, Basic}, learning rate $\in \{0.1, 0.01, 0.001, 0.0001\}$, hidden dimension $\in \{8, 16, 32, 64\}$, epochs after each goal $\in \{5, 10, 50, 100\}$. We use the implementations available in [1].

Since all methods above are online algorithms, each of the models P_n is updated after every episode n. In order quantify performance with a single score, we use the mean probability assigned to the ground truth goal type g^* over all episodes $\{\xi_n\}_{n=1}^N$:

$$\overline{P}(g^*|\{\xi_n\}_{n=1}^N) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} P_n(g^*|\xi_{nt}) \tag{3.14}$$

We denote Equation 3.14 as $\overline{P}_{g^*}$ for brevity. The goal forecasting performance results are summarized in Table 3.5 using $\overline{P}_{g^*}$.

3.5.7 Goal Forecasting with Perfect Visual Detectors

The experimental results up to this point have exclusively used visual detectors as input (*e.g.*., SLAM, scene classification, object recognition). While we have shown that our approach learns meaningful human activity models from real computer vision input, we would also like to understand how our online IRL method performs when decoupled from the noise of the vision-based input. We perform the same experiments described in Section 3.5.6 but with idealized (ground truth) inputs for goal discovery and action recognition. We still use SLAM for localization.

Table 3.6 summarizes the mean true goal probability for each of the dataset environments. We observe a mean absolute performance improvement of 0.27 by using idealized inputs. Our proposed model continues to outperform the baselines methods. This performance indicates that as vision-based component technologies improve, we can expect significant improvements in the ability to predict the goals of human activities.

We also measure performance when the action detection is built from ground truth and the goal discovery is built from our described methods. Our expectation is that DARKO with stop-based

Table 3.6: **Goal Forecasting Results (Labelled Detections):** Proposed goal posterior achieves best $\overline{P}_{g^*}$ (mean probability of true goal). Methods benefit from better detections.

Method	Home 1	Home 2	Office 1	Office 2	Lab 1
DARKO	**0.851**	**0.683**	**0.700**	**0.666**	**0.880**
MMED [84]	0.648	0.563	0.589	0.624	0.683
RNN	0.441	0.322	0.504	0.454	0.651
Logistic	0.517	0.519	0.650	0.657	0.774
Uniform	0.153	0.128	0.154	0.151	0.167

Table 3.7: **Visual goal discovery:** Better goal discovery (cf. Table 3.4) yields better $\overline{P}_{g^*}$. Here, action detection labels are used to isolate performance differences.

Method	Home 1	Home 2	Office 1	Office 2	Lab 1
Scene-based	0.438	0.346	0.560	0.238	0.426
Stop-based	0.614	0.395	0.644	0.625	0.709

Table 3.8: **Feature Ablation Results:** Full state and action features (Sec. 3.3.1) yield best goal prediction results.

Feature Type	Home 1	Home 2	Office 1	Office 2	Lab 1
State+Action	**0.851**	**0.683**	**0.700**	**0.666**	0.880
State only	0.735	0.574	0.581	0.549	**0.892**
Position only	0.674	0.597	0.605	0.622	0.886

discovery should outperform DARKO with scene-based based discovery, given the stop-detector's more reliable goal detection performance (Table 3.4). The results over the dataset are given in Table 3.7, confirming our expectation.

3.5.8 Goal Forecasting Performance over Time

In additional to understanding the performance of goal prediction with a single score, we also plot the performance of goal prediction over time. We evaluate the goal forecasting performance as a function of the fraction of time until reaching the goal. In Figure 3.6, we plot the *mean* probability of the true goal at each fractional time step $\widehat{P}(g^*|\xi_t) = \frac{1}{N}\sum_{n=1}^{N} P_n(g^*|\xi_{nt})$. Using fractional trajectory length allows for a performance comparison across trajectories of different lengths.

As shown in Figure 3.6, DARKO exhibits the property of maintaining uncertainty early in the trajectory and converging to the correct prediction as time elapses in most cases. In contrast, the logistic regression, RNN, and MMED perform worse at most time steps. As it approaches the goal, our method always produces a higher confidence in the correct goal with lower variance. We tried `argmax` and Platt scaling [156] to perform multi-class prediction with MMED; `argmax` yielded higher $\overline{P}_{g^*}$, in addition to making $\widehat{P}_{g^*}$ noisier. While the RNN sees many states, its trajectory-centric hidden-state representation may not have enough data to generalize as well as the state-centric baselines.

3.5.9 Reward Function Feature Ablation Analysis

In Table 3.8, we show the mean true goal probability when labels are used as detectors (to isolate performance in the ideal case). While the purely positional representation of state performs well, it

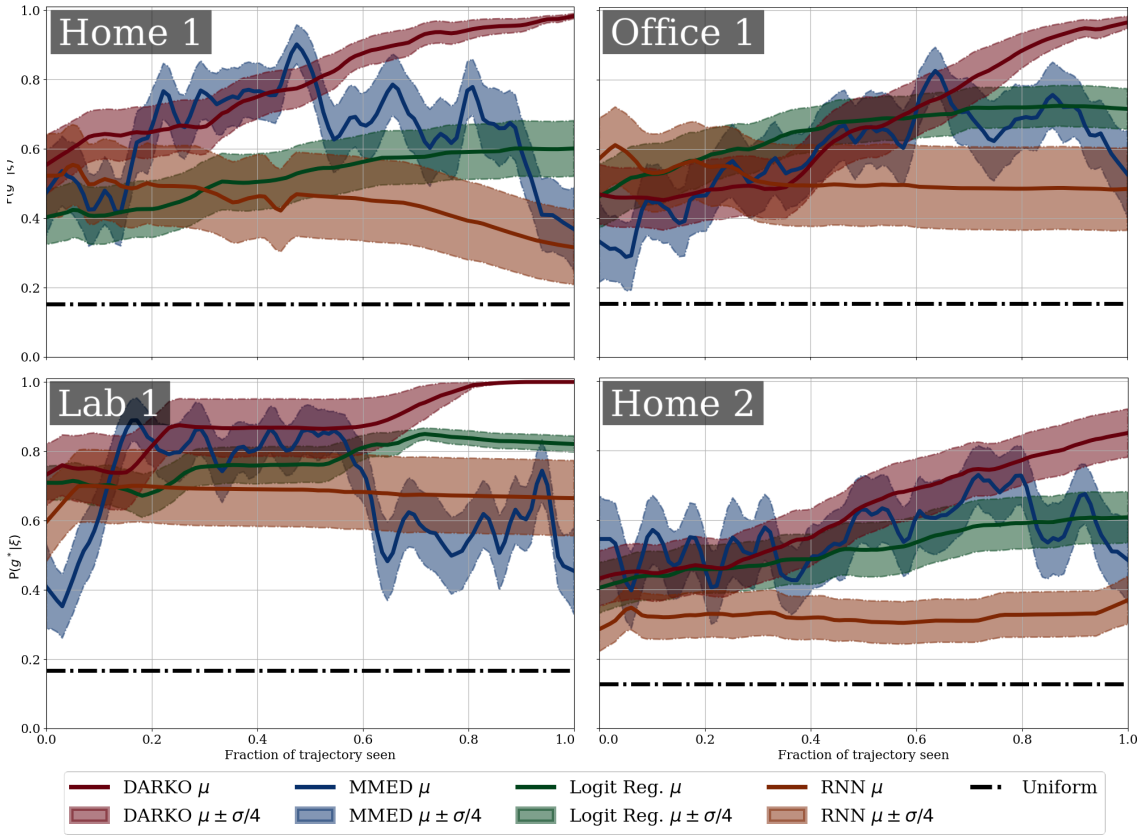


Figure 3.6: **Goal posterior forecasting over time:** $\widehat{P}_{g^*}$ vs. fraction of trajectory length, across all trajectories. DARKO outperforms other methods and becomes more confident in the correct goal as the trajectories elapse.

is almost always outperformed by the full representation of rewards that include features of both the full state and action. In Lab 1, the simpler representations slightly outperform the full, due to the relative simplicity of the high-level activities in Lab 1. Here, knowledge of the state and previous goal alone is highly predictive of future goal. These results indicate that our method was able to (i) make use of simple representations, and (ii) capitalize on more information present in the richer representations without sacrificing performance where the representation is unnecessary.

3.5.10 Incorporating Detection Noise

Current paradigms in vision often yield noise in the action and goal detectors necessary for DARKO. We hypothesize that judicious incorporation of these uncertainties can improve our method's performance. We first describe our method for incorporating uncertainty in each goal detection, then conduct a performance analysis with synthetic noise. Then, we analyze the performance with real, noisy goal detection. We find DARKO can still perform well with forms of noisy goal and action detection. We find incorporating goal uncertainty significantly improves performance with synthetic noise, and shows improvements in the real goal detector setting. These results show that DARKO can tolerate the effects of noise, and further support the claim that it can enjoy the benefits of better scene and activity detection algorithms.

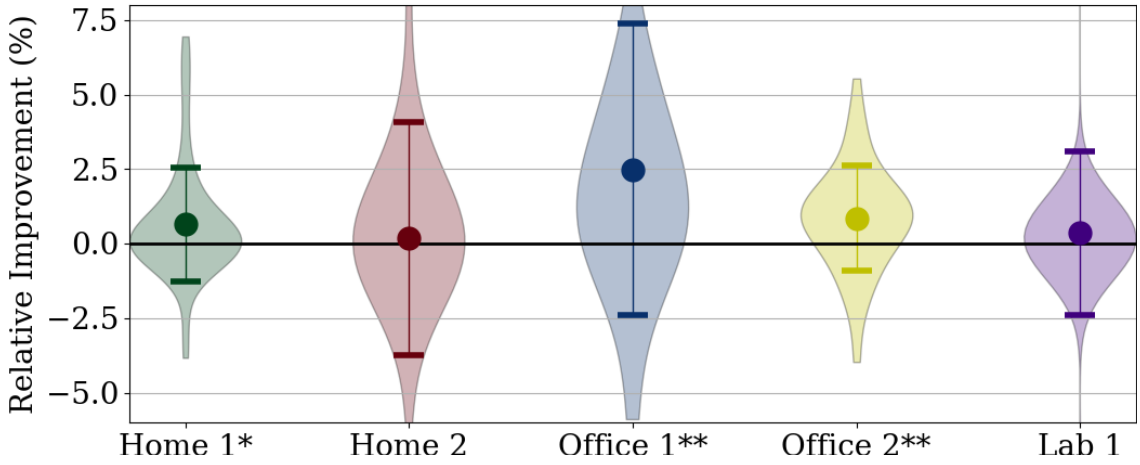


Figure 3.7: **Relative improvement from incorporating goal uncertainty.** Per-scene violin plots, means, and standard deviations are shown. Per-scene one-sided paired t-tests are performed, testing the hypotheses that incorporating goal uncertainty improves goal prediction performance. A * indicates $p < 0.05$, and ** indicates $p < 0.005$.

3.5.10.1 Harnessing goal detection confidence

In many scenarios, probability $\rho_g \in [0, 1]$ may be associated with each goal detection. We designed an effective method for handling real-world uncertainty. For known perfect goals, SOFTVALUEITERATION uses $V(g) = 0, \forall g \in \mathcal{S}_g$. Each goal is a maxima of the value function $V(s) \in (-\infty, 0], \forall s \in \mathcal{S}$ and represents a reward of 1 in log space. *We replace each goal value with its log-probability:* $V(g) = \ln \rho_g$*, which has the effect of biasing the policy towards goals with greater certainty*. This results from the value iteration assigning higher value to states and actions closer to more certain goals, which makes the policy likelier to visit them. For example, if the goal detector yields a false positive of `bathroom` in the same area as a true positive detection of `kitchen`, the goal prediction posteriors for both goals will suffer, unless the false positive has an associated low ρ_g (high uncertainty), in which case the policy is biased towards the correct goal of kitchen.

3.5.10.2 Noise analysis

To test the efficacy of the goal detection confidence weighting approach, we first analyze DARKO under the effect of adding noise to the GT. We add incorrect goal detections with probabilities $\rho_g \sim \mathcal{N}(0.1, 0.05)$, under various amounts of noise inserted uniformly at random across time: from $10\%, 20\%, \ldots, 90\%$ of the number of original goal detections. For every scene, at each noise amount, we sample noise 5 times, and run DARKO with and without goal uncertainty for each corrupted sample, resulting in 225 paired experiments that evaluate the average goal forecasting probability. Per-scene results are shown in Figures 3.7. *A one-sided Wilcoxon signed-rank test supports the hypothesis that incorporating high goal uncertainty yields better goal posterior prediction performance than not incorporating the uncertainty with* $p < 10^{-7}$.

3.5.11 Empirical Regret Analysis

We empirically show that our model has no-regret with respect to the best model computed in hindsight under the MaxEntIRL loss function (negative log-loss). In particular, we compute the regret (cumulative loss difference) between our online algorithm and the best hindsight model

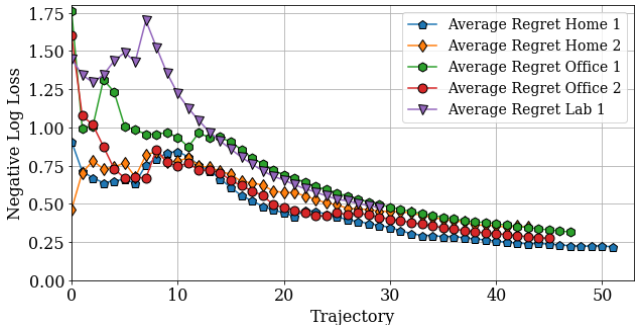


Figure 3.8: **Empirical regret.** DARKO exhibits sublinear convergence in average regret. Initial noise is overcome after DARKO adjusts to the user's early behaviors.

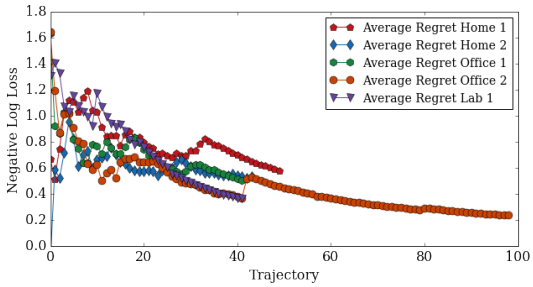


Figure 3.9: **Noisy Empirical Regret.** DARKO's online behavior model exhibits sublinear convergence in average regret. Initial noise is overcome after DARKO adjusts to learning about the user's early behaviors.

using the batch MaxEntIOC algorithm [257] at the end of all episodes. We plot the average regret $\frac{\mathcal{R}_t}{t}$ for each environment in the dataset in Figure 3.8. The average regret of our algorithm approaches zero over time, matching our analysis.

We additionally show the empirical regret when using our goal discovery and action detection methods in Figure 3.9. We observe somewhat noisier regret behavior than in the original case, as the underlying demonstrations are noisier. The number of trajectories in Office 2 is higher here due to errors in the goal forecasting method, resulting in more goals being detected, which segments the demonstrations into more trajectories.

3.5.12 Evaluation of Trajectory Length Estimates

Our model can also be used to estimate how long it will take a person to reach a predicted goal state. We predict the expected trajectory length as derived in Section 3.4.1. For the n-th episode, we use the normalized trajectory length prediction error defined as $\epsilon_n = \sum_{t=1}^{T_n} \frac{|\tau_{nt} - \hat{\tau}_{nt}|}{\tau_{nt}}$, where τ_{nt} is the true trajectory length and $\hat{\tau}_{nt}$ (Eq. 3.8) is the predicted trajectory length. Proper evaluation of trajectory length towards a goal is challenging because our approach must learn valid goals in an online fashion. When a person approaches a new goal, our approach cannot accurately predict the goal because it has yet to learn that it is a valid goal state. As a result, our algorithm makes wrong goal predictions during episodes that terminate in new goal states. If we simply evaluate the mean performance, it will be dominated by the errors of the first episode terminating in a new goal state.

We evaluate median ϵ_n over all N episodes. The median is not dominated by the errors of the first episode toward a new goal. We find most trajectory length forecasts are accurate, evidenced by

Table 3.9: **Trajectory length forecasting results.** Error is relative to the true length of each trajectory. Most trajectory forecasts are fairly accurate.

Statistic	Home 1	Home 2	Office 1	Office 2	Lab 1
Med. % Err.	30.0	34.8	17.3	18.4	6.3
Med. % Err. NN	29.0	33.5	42.9	36.0	35.4
Mean $\|\xi\|$	20.5	31.0	27.1	13.7	23.5

the median of the normalized prediction error in Table 3.9. We include a partial-trajectory nearest neighbors baseline (NN). In Lab 1, the median trajectory length estimate is within 6.3% of the true trajectory length.

3.6 Visualizations

We provide example 3D visualizations of (i) future state visitation and (ii) the value function. These visualizations were produced in Mayavi [160], and include the SLAM points.

3.6.1 Future state visitation visualizations

See Figure 3.10 for example visualizations of the expected future visitation counts. In order to visualize in 3 dimensions, we take the max visitation count across all states at each position. In rows 1, 2, and 3, a single demonstration is shown, which adapts to the agent's trajectory (history). In row 4, the future state distribution drastically changes after each time the agent reaches a new goal.

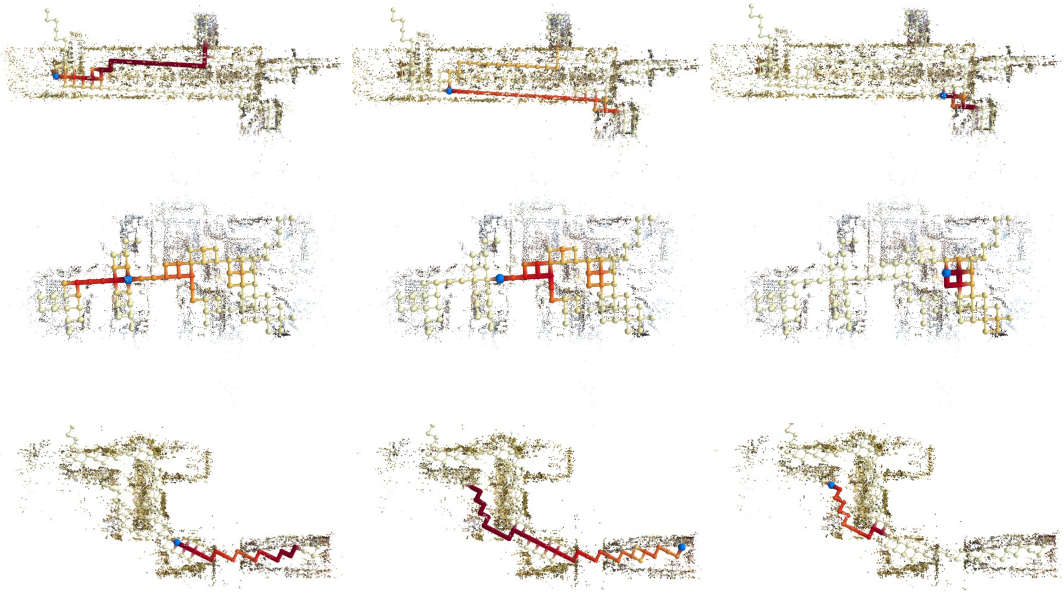


Figure 3.10: Future state visitation predictions changing as the agent (blue sphere) follows their trajectory. The state visitations are projected to 3D by taking the max over all states at each location. The visualizations are, by row: Office 1, Home 1, Lab 1

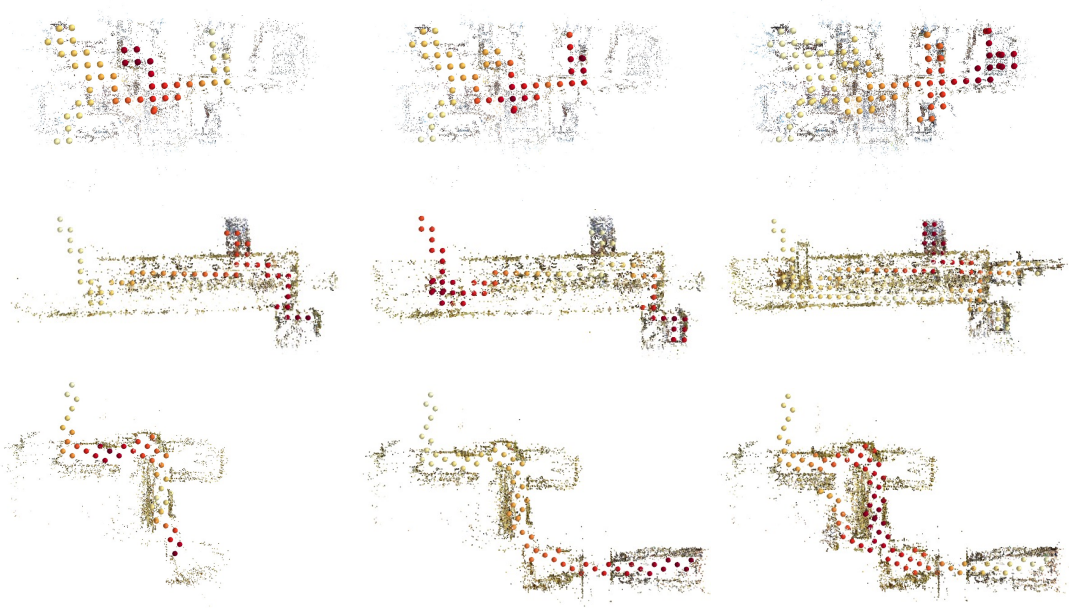


Figure 3.11: Projections of the value function ($V(s)$) for environments as time elapses (left to right). The state space expands as the user visits more locations. For each position, the maximum value (across all states at that position) is displayed: $\max_{s \in \mathcal{S}_x} V(s)$. From top to bottom, the environments are Home 1, Office 1, Lab 1.

3.6.2 Value function visualizations

See Figure 3.11 for example visualizations of the value function over time. Note 1) the state space size changes, and 2) that the value function changes over time, as the component of state that indicates the previous goal affects the value function.

3.7 Conclusion

We proposed the first method for continuously modeling and forecasting a first-person camera wearer's future semantic behaviors at far-reaching spatial and temporal horizons. Our method goes beyond predicting the physical trajectory of the user to predict their future semantic goals, and models the user's relationship to objects and their environment. We have proposed several efficient and extensible methods for forecasting other semantic quantities of interest. Exciting avenues for future work include building upon the semantic state representation to model more aspects of the environment (which enables forecasting of more detailed futures), validation against human forecasting performance, and further generalizing the notion of a "goal" and how goals are discovered.

This model of sequential behavior extends the discrete single-behavior capability of Chapter 2 to discrete sequential behaviors. However, it cannot represent the continuous underlying motions of high-level activities. We seek this modeling capability in order to recover a finer-grained understanding of the future.

Chapter 4

Forecasting Motion Trajectories with Deep Reversible Generative Models

4.1 Introduction

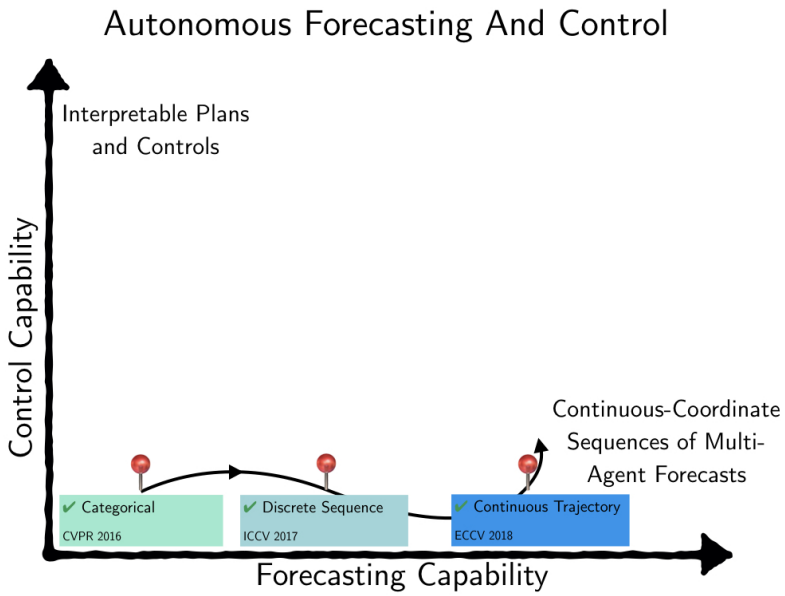


Figure 4.1: This chapter focuses on building a multi-step continuous motion forecasting approach.

We consider forecasting a vehicle's trajectory (*i.e.*, predicting future paths). Forecasts can be used to foresee and avoid dangerous scenarios, plan safe paths, and model driver behavior. Context from the environment informs prediction, *e.g.* a map populated with features from imagery and LIDAR. We would like to learn a context-conditioned *distribution* over spatiotemporal trajectories to represent the many possible outcomes of the vehicle's future. With this distribution, we can perform inference tasks such as *sampling* a set of plausible paths, or *assigning a likelihood* to a particular observed path. Sampling suggests routes and visualizes the model; assigning likelihood helps measure the model's quality. This approach's capability to forecast motion as a sequence of continuous points makes it more powerful than discrete sequence modeling. Its situation within our contributions is depicted in Fig. 4.1.

Our key motivation is to learn a trajectory forecasting model that is simultaneously "diverse"—

Figure 4.2: *Left:* Natural image input. *Middle:* generated trajectories (red circles) and true, expert future (blue squares) overlaid on LIDAR map. *Right:* Generated trajectories respect approximate prior $\tilde{p}$, here a "cost function," overlaid as a heatmap. Making the expert paths likely corresponds to $\min_\pi H(p, q_\pi)$. Only producing likely paths corresponds to steering the trajectories away from unlikely territory via $\min_\pi H(q_\pi, \tilde{p})$. Doing both, *i.e.* producing most of the likely paths while mostly producing likely paths corresponds to $\min_\pi H(p, q_\pi) + \beta H(q_\pi, \tilde{p})$.

covering all the modes of the data distribution—and "precise" in the sense that it rarely generates bad trajectories, such as trajectories that intersect obstacles. Fig. 4.2 contrasts a model trained to cover modes, versus a model trained to cover modes *and* generate good samples, which generates fewer samples hitting perceived obstacles.

To achieve these ends, we propose learning a distribution over trajectories q_π that minimizes a symmetrized cross-entropy between q_π and the training data distribution, p. We represent q_π as a trajectory distribution induced by *rolling out* (simulating) a stochastic one-step *policy* π for T steps to produce a trajectory sample x. Denoting the scene context by ϕ, our objective can be written as

$$\min_\pi \underbrace{\mathbb{E}_{x\sim p} - \log q_\pi(x|\phi)}_{H(p,q_\pi)} + \beta \underbrace{\mathbb{E}_{x\sim q_\pi} - \log \tilde{p}(x|\phi)}_{H(q_\pi,\tilde{p})} \tag{4.1}$$

The two cross-entropy terms serve complementary purposes, as illustrated in Fig. 4.3: $H(p, q_\pi)$ encourages q_π to cover the modes of p, but fails to adequately penalize generating "low-quality" samples; $H(q_\pi, \tilde{p})$ encourages q_π to produce "high-quality" samples likely under an approximate data density $\tilde{p}$, but is insensitive to mode loss.

We advocate using the symmetrized cross-entropy metrics *for both training and evaluation* of trajectory forecasting methods. This is made feasible by viewing the distribution q_π as the *push-forward* of a base distribution under the function g_π that *rolls-out* (simulates) a stochastic policy π (see Fig. 4.4b). This idea (also known as the *reparameterization trick*, [47, 99]) enables optimization of model-sample quality metrics such as $H(q_\pi, \tilde{p})$ with SGD. Our representation also admits efficient accurate computation of $H(p, q_\pi)$, even when the policy is a very complex function of context and past state, such as a CNN.

Consider, for instance, the middle subfigure of Fig. 4.3: $H(q_\pi, p) \approx \frac{1}{2}(M_0' - \log \epsilon)$. We can derive this by analyzing the relative supports of the *good* and *bad* versions of q_π. Suppose q_π is the good reference model and q_π' is the bad model illustrated in the middle figure. We assume q_π' is a mixture of q_π and a "rotated" version of q_π^R that rotates the support of q_π into the pictured obstacles, such that $q_\pi'(x) = 0.5q_\pi(x) + 0.5q_\pi^R(x)$. Let $A = \text{support}(q_\pi)$ and $B = \text{support}(q_\pi^R)$. Assume A and B are

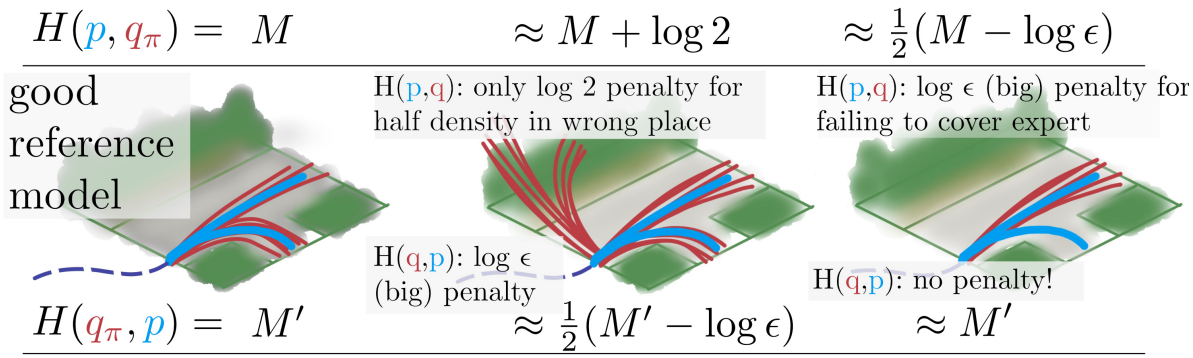


Figure 4.3: Illustration of the complementarity of cross-entropies $H(p, q_\pi)$ (top) and $H(q_\pi, p)$ (bottom). Dashed lines show past vehicle path. Light blue lines delineate samples from the data (expert) distribution p. Samples from the model q_π are depicted as red lines. Green areas represent obstacles (areas with low p). The left figure shows cross-entropy values for a reference model. Other figures show poor models and their effects on each metric. ϵ is a very small nonnegative number. This Figure is based on graphics created by Paul Vernaza.

approximately disjoint. We then have

$$H(q'_\pi, p) = -\int q'_\pi(x) \log p(x) dx \tag{4.2}$$

$$= -\int_{A \cup B} (\frac{1}{2} q_\pi(x) + \frac{1}{2} q^R_\pi(x)) \log p(x) \tag{4.3}$$

$$\approx -\frac{1}{2} \int_A q_\pi(x) \log p(x) + -\frac{1}{2} \int_B q^R_\pi(x) \log p(x) \tag{4.4}$$

Assuming $\log p(x) \approx \epsilon, \ \forall x \in B$, we then have $H(q'_\pi, p) \approx \frac{1}{2}(M'_0 - \log \epsilon)$. The other terms can be derived via similar analyses.

We present the following novel contributions: 1) recognize and address the diversity-precision trade-off of generative forecasting models and formulating a symmetrized cross-entropy training objective to address it; 2) propose to train a policy to induce a roll-out distribution minimizing this objective; 3) use the pushforward parameterization to render inference and learning in this model efficient; 4) refine an existing deep imitation learning method (GAIL) based on our parameterization; 5) illuminate deficiencies of previously-used trajectory forecasting metrics; 6) outperform state-of-the-art forecasting and imitation learning methods, including our improvements to GAIL; 7) present CALIFORECASTING, a novel large scale dataset designed specifically for vehicle ego-motion forecasting.

4.2 Related Work

Trajectory Forecasting prior work spans two primary domains: trajectories of vehicles, and trajectories of people. The method of [114] predicts future trajectories of wide-receivers from surveillance video. In [16, 101, 128, 246] future pedestrian trajectories are predicted from surveillance video. Deterministic vehicle predictions are produced in [93], and deterministic pedestrian trajectories are produced in [8, 147, 178]. However, non-determinism is a key aspect of forecasting: the future is generally uncertain, with many plausible outcomes. While several approaches forecast distributions over trajectories [62, 113], global sample quality and likelihood have not been considered or measured, hindering performance evaluation.

Activity Forecasting is distinct from trajectory forecasting, as it predicts categorical activities. In [84, 107, 185, 186], future activities are predicted via classification-based approaches. In [170], a first-person camera wearer's future goals are forecasted with Inverse Reinforcement Learning (IRL). IRL has been applied to predict and control robot, taxi, and pedestrian behavior [101, 163, 256].

Imitation Learning can be used to frame our problem: learn a model to mimic an agent's behavior from a set of demonstrations [3]. One subtle difference is that in forecasting, we are not required to actually execute our plans in the real world. IRL is a form of imitation learning in which a reward function is learned to model demonstrated behavior. In the IRL method of [245], a cost map representation is used to plan vehicle trajectories. However, no time-profile is represented in the predictions, preventing use of time-profiled metrics and modeling. GAIL [83, 121] is also a form of IRL, yet its adversarial framework and policy optimization are difficult to tune and lead to slow convergence. By adding the assumption of model dynamics, we derive a new differentiable GAIL training approach, supplanting the noisy, inefficient policy gradient search procedure. We show this easier-to-train approach achieves better performance in our domain.

Image Forecasting methods generate full image or video representations of predictions, endowing their samples with interpretability. In [232–234], unsupervised model are learned to generate sequences and representations of future images. In [235], surveillance image predictions of vehicles are formed by smoothing a patch across the image. [236] and [231] also predict future video frames with an intermediate pose prediction. In [57], predictions inform a robot's behavior, and in [228], policy representations for imitation and reinforcement learning are guided by a future observation forecasting objective. In [23], image boundaries are predicted. One drawback to image-based forecasting methods is difficulty in measurement, a drawback shared by many popular generative models.

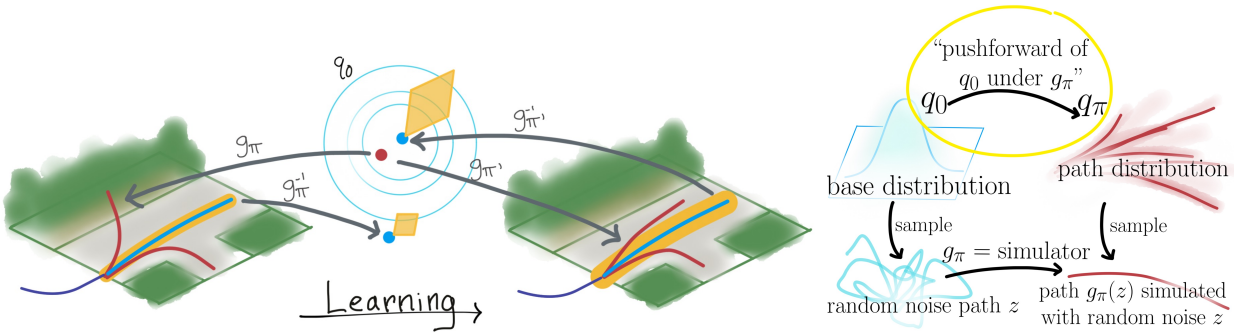
Generative models have surged in popularity [47, 70, 74, 83, 113, 233, 254]. However, one major difficulty is *performance evaluation*. Most popular models are quantified through heuristics that attempt to measures the "quality" of model samples [113]. In image generation, the Inception score is a popular heuristic [190]. These fail to measure the learned distribution's likelihood, the gold standard of evaluating probabilistic models. Notable exceptions include [47, 100], which also leverage invertible pushforward models to perform exact likelihood inference.

4.3 Approach

We approach the forecasting problem from an *imitation learning* perspective, learning a *policy* (state-to-action mapping) π that mimics the actions of an expert in varying contexts. We are given a set of training episodes (a short car path trajectory) $\{(x, \phi)_n\}_{n=1}^N$. Each episode $(x, \phi)_n$ has $x \in \mathbb{R}^{T\times 2}$ as a sequence of T two-dimensional future vehicle locations and ϕ as an associated set of side information. In our implementation, ϕ contains the past path of the car and a feature grid derived from LIDAR and semantic segmentation class scores. The grid is centered on the vehicle's position at $t = 0$ and is aligned with its heading.

Repeatedly applying the policy π from a start state with the context ϕ results in a distribution $q_\pi(x|\phi)$ over trajectories x, since our policy is stochastic. Similarly, the training set is drawn from a *data distribution* $p(x|\phi)$. We therefore train π so as to minimize a divergence between q_π and p. This divergence consists of a weighted combination of the cross-entropies $H(p, q_\pi)$ and $H(q_\pi, \tilde{p})$. The distribution $\tilde{p}$ is an approximation to p, which we assume cannot be evaluated. As discussed in Sec. 4.3.1.3, we might choose $\tilde{p}$ to be approximately uniformly distributed over non-obstacle regions. In the following, Φ denotes the distribution of ground-truth features:

$$\min_\pi \mathbb{E}_{\phi\sim\Phi}\left[-\mathbb{E}_{x\sim p(\cdot|\phi)} \log q_\pi(x|\phi) - \beta\mathbb{E}_{x\sim q_\pi(\cdot|\phi)} \log \tilde{p}(x|\phi)\right]. \tag{4.5}$$



(a) (b)

Figure 4.4: (a) Consider making trajectories inside the yellow region on the road likelier by increasing $\log q_\pi(x)$ for the demonstration $x \sim p$ inside the region. This is achieved by making an infinitesimal region around $g_\pi^{-1}(x)$ more likely under q_0 by moving the region (yellow parallelogram, size proportional to $|\det J_{g_\pi}|^{-1}$) towards a mode of q_0 (here, the center of a Gaussian), and making the region bigger. Increasing $\log \tilde{p}(x)$ for some sample $x \sim q_\pi$ is equivalent to sampling a (red) point z from q_0 and adjusting π so as to increase $\log \tilde{p}(q_0(z))$. (b) Pushing forward a base distribution to a trajectory distribution. Both Figures are based on graphics created by Paul Vernaza.

The motivation for this objective is illustrated in Fig. 4.3. The two factors are complementary. $H(p, q_\pi)$ is intuitively similar to *recall* in binary classification, in that it is very sensitive to the model's ability to produce all of the examples in the dataset, but is relatively insensitive to whether the model produces examples that are unlikely under the data. $H(q_\pi, \tilde{p})$ is intuitively similar to *precision* in that it is very sensitive to whether the model produces samples likely under $\tilde{p}$, but is insensitive to q_π's likelihood to produce *all* samples in the dataset.

4.3.1 Pushforward distribution modeling

Optimizing Eq equation 4.5 presents at least two challenges: we must be able to evaluate $q_\pi(x|\phi)$ at arbitrary x in order to compute $H(p, q_\pi)$, and we must be able to differentiate the expression $\mathbb{E}_{x\sim q_\pi(\cdot|\phi)} \log \tilde{p}(x|\phi)$. We address these issues by constructing a learnable bijection, g_π between samples from q_π and samples from a simple noise distribution q_0, as illustrated in Fig. 4.4b; in our construction, the bijection is intepreted as a simulator mapping noise to simulated outcomes. This assumption allows us to evaluate the required expressions and derivatives via the change-of-variables formula and the *reparameterization trick*.

Specifically, let $g_\pi(z; \phi) : \mathbb{R}^{T\times 2} \to \mathbb{R}^{T\times 2}$ be a simulator mapping noise sequences $z \sim q_0$ and scene context ϕ to forecasted outcomes x. Then the distribution of forecasted outcomes $q_\pi(x|\phi)$ is fully determined by q_0 and g_π: this distribution is known as the *pushforward* of q_0 under g_π, as we are using g_π to "push forward" a distribution defined on the domain of g_π to one defined on its codomain. If g_π is differentiable and invertible, then q_π can be derived from the change-of-variables formula for multivariate integration:

$$q_\pi(x|\phi) = q_0\big(g_\pi^{-1}(x; \phi)\big)\big|\det J_{g_\pi}(g_\pi^{-1}(x; \phi))\big|^{-1}, \tag{4.6}$$

where $J_{g_\pi}(g_\pi^{-1}(x; \phi))$ is the Jacobian of g_π evaluated at $g_\pi^{-1}(x; \phi)$. This resolves both of the aforementioned issues: we can evaluate q_π and we can rewrite $\mathbb{E}_{x\sim q_\pi} \log \tilde{p}(x)$ as $\mathbb{E}_{z\sim q_0} \log \tilde{p}(g_\pi(z; \phi))$, since $g_\pi(z; \phi) \sim q_\pi$. The latter allows us to move derivatives wrt. π inside the expectation, as q_0 does not

depend on π. Eq. equation 4.5 can then be rewritten as:

$$\min_{\pi} -\mathbb{E}_{\phi\sim\Phi}\mathbb{E}_{x\sim p(\cdot|\phi)} \log \frac{q_0(g_\pi^{-1}(x;\phi))}{\left|\det J_{g_\pi}(g_\pi^{-1}(x;\phi))\right|} - \beta\mathbb{E}_{z\sim q_0} \log \tilde{p}(g_\pi(z;\phi)|\phi) \tag{4.7}$$

Fig. 4.4a illustrates how this representation aids learning.

We note ours is not the only way to represent q_π and optimize Eq. equation 4.5. As long as q_π is analytically differentiable in the parameters, we may also apply REINFORCE [240] to obtain the required parameter derivatives. However, empirical evidence and some theoretical analysis suggests that the reparameterization-based gradient estimator typically yields lower-variance gradient estimates than REINFORCE [61]. This is consistent with the results we obtained in Sec. 2.3.

4.3.1.1 An invertible, differentiable simulator.

In order to exploit the pushforward density formula equation 4.6, we must ensure g_π is invertible and differentiable. Inspired by [47, 98], we define g_π as an autoregressive map, representing the evolution of a controlled, discrete-time stochastic dynamical system with additive noise. Denoting $[x_1, \ldots, x_{t-1}]$ as $x_{1:t-1}$, and $[x_{1:t-1}, \phi]$ as ψ_t, the system is:

$$x_t \triangleq \mu_t^\pi(\psi_t;\theta) + \sigma_t^\pi(\psi_t;\theta)z_t, \tag{4.8}$$

where $\mu_t^\pi(\psi_t;\theta) \in \mathbb{R}^2$ and $\sigma_t^\pi(\psi_t;\theta) \in \mathbb{R}^{2\times 2}$ represent the stochastic one-step *policy*, and θ its parameters. The context, ϕ, is given in the form of a past trajectory $x_{\text{past}} = x_{-H_{\text{past}}+1:0} \in \mathbb{R}^{2H_{\text{past}}}$, and overhead feature map $M \in \mathbb{R}^{H_{\text{map}}\times W_{\text{map}}\times C}$: $\phi = (x_{\text{past}}, M)$. Note that the case $\sigma^\pi = 0$ would correspond to simply evolving the state by repeatedly applying μ^π—though this case is not allowed, as then g_π would not be invertible. However, as long as σ_t^π is invertible for all x, then g_π is invertible, and it is differentiable in x as long as μ^π and σ^π are differentiable in x. Since x_{τ_1} is not a function of x_{τ_2} for $\tau_1 < \tau_2$, the determinant of the Jacobian of this map is easily computed, because it is triangular (see supplement). Thus, we can easily compute terms in Eq. 4.7 via the following:

$$[g_\pi^{-1}(x)]_t = z_t = \sigma_t^\pi(\psi_t;\theta)^{-1}(x_t - \mu_t^\pi(\psi_t;\theta)) \tag{4.9}$$

$$\log\left|\det J_{g_\pi}(g_\pi^{-1}(x;\phi))\right| = \sum_t \log\left|\det\big(\sigma_t^\pi(\psi_t;\theta)\big)\right| \tag{4.10}$$

We note that q_π can also be computed via the chain rule of probability. For instance, if $z_t \sim$ is standard normal, then the marginal distributions are

$$q_\pi(x_t|\psi_t) = \mathcal{N}(x_t; \mu = \mu_t^\pi(\psi_t;\theta), \Sigma = \sigma_t^\pi(\psi_t;\theta)\sigma_t^\pi(\psi_t;\theta)^\top). \tag{4.11}$$

However, since it is still necessary to compute g_π in order to optimize $H(q_\pi, \tilde{p})$, we find it simplifies the implementation to compute q_π in terms of g_π.

Our path distribution can be thought of being *parameterized* by a continuous action-space policy, in the following way. The output of our stochastic policy is a distribution over continuous actions: $\pi(a_{t-1}|\psi_{t-1};\theta)$. The state-state transition dynamics in general continuous-action RL problems can be written as:

$$p(x_t|\psi_{t-1}) = \int p(x_t|\psi_{t-1}, a_{t-1})\pi(a_{t-1}|\psi_{t-1};\theta)\mathrm{d}a_{t-1}$$

By taking $p(x_t|\psi_{t-1}, a_{t-1}) = \delta(x_t - a_{t-1})$, *i.e.*, assuming the policy can fully control the state dynamics, in that it chooses an action, and the next state is the Dirac delta function about that chosen action, we receive $p(x_t|\psi_{t-1}) = \pi(x_t|\psi_{t-1})$. This means that $q_\pi(x_t|\psi_t) = \pi(x_t|\psi_{t-1};\theta)$. Therefore, we can think of the policy as the one-step marginal $q_\pi(x_t|\psi_t) = N(x_t; \mu_t^\pi(\psi_t), \sigma_t^\pi(\psi_t)\sigma_t^\pi(\psi_t)^\top)$.

4.3.1.2 Derivation of Jacobian and its determinant

Given the recursive rollout equation :

$$x_t = 2x_{t-1} - x_{t-2} + \mu_t(\psi_t) + \sigma_t(\psi_t)z_t,$$

then

$$J_{g_\pi}(g_\pi^{-1}(x)) = \frac{\mathrm{d}g_\pi}{\mathrm{d}g_\pi^{-1}(x)} = \begin{bmatrix} \sigma_1^\pi(\psi_1), & 0 & \dots & 0 \\ \frac{\mathrm{d}x_2}{z_1} & \sigma_2^\pi(\psi_t) & \dots & 0 \\ \vdots & & \ddots & 0 \\ \frac{\mathrm{d}x_T}{z_1} & \frac{\mathrm{d}x_T}{z_2} & \dots & \sigma_T^\pi(\psi_T) \end{bmatrix}.$$

Therefore,

$$\log \left| \det \left(\frac{\mathrm{d}g_\pi}{\mathrm{d}g_\pi^{-1}(x)} \right) \right| = \log \left| \prod_{t=1}^{T} \det(\sigma_t^\pi(\psi_t)) \right| = \sum_{t=1}^{T} \log \left| \det(\sigma_t^\pi(\psi_t)) \right|.$$

4.3.1.3 Prior approximation of the data distribution.

Evaluating $H(q_\pi, p)$ directly is unfortunately impossible, since we cannot evaluate the data distribution p's PDF. We therefore propose approximating it with a very simple density estimator $\tilde{p} \approx p$ trained independently and then fixed while training q_π. Simplicity reduces sample-induced variance in fitting $\tilde{p}$—crucial, because if $\tilde{p}$ severely underestimates p in some region R due to sampling error, then $H(q_\pi, \tilde{p})$ will erroneously assign a disproportionate penalty to samples from q_π landing in R.

We consider two options for $\tilde{p}$—first, simply using a kernel density estimator with a relatively large bandwidth. Since we have only one training sample per episode, this reduces to a single-kernel model. Choosing an isotropic Gaussian kernel, $H(q_\pi, \tilde{p})$ becomes $\mathbb{E}_{\hat{x}\sim q_\pi(\cdot|\phi)}\|x - \hat{x}\|^2/\gamma^2$, where (x, ϕ) constitutes an episode from the data. The net objective equation 4.5 in this case corresponds to $H(p, q_\pi)$ plus a mean squared distance penalty between model samples and data samples.

The second possibility is making an i.i.d. approximation; *i.e.*, parameterizing $\tilde{p}$ as $\tilde{p}(x \mid \phi) = \prod_t \tilde{p}_c(x_t \mid \phi)$. We proceed by discretizing x_t in a large finite region centered at the vehicle's start location; $\tilde{p}_c$ then corresponds to a categorical distribution with L classes representing the L possible locations. Training the i.i.d. model can then be reduced to training $\tilde{p}_c$ via logistic regression:

$$\min_{\tilde{p}} -\mathbb{E}_{x\sim p} \log \tilde{p}(x) = \max_{\theta} \mathbb{E}_{x\sim p} \sum_t -C_\theta(x_t, \phi) - \log \sum_{y=1}^{L} \exp -C_\theta(y, \phi), \tag{4.12}$$

where $C_\theta = -\log \tilde{p}_c$ can be thought of as a *spatial cost function* with parameters θ. We found it useful to decompose $C_\theta(y)$ as a sum $C_\theta^0(y) + C_\theta^1(y, \phi)$, where $C_\theta^0 \in \mathbb{R}^L$ is thought of as a non-contextual location prior, and $C_\theta^1(y, \phi)$ has the form of a convolutional neural network acting on the spatial feature grid in ϕ and producing a grid of scores $\in \mathbb{R}^L$. Fig. 4.5 shows example learned $C_\theta^1(\cdot, \phi)$.

4.3.2 Policy modeling

We turn to designing learnable functions μ_t^π and σ_t^π. Across our three models, we use the following expansion: $\mu_t^\pi(\psi_t) = 2x_t - x_{t-1} + \hat{\mu}_t^\pi(\psi_t)$. The first terms correspond to a *constant velocity step* $(x_t + (x_t - x_{t-1}))$, and let us interpret $\hat{\mu}_t^\pi$ as a *deterministic acceleration*. Altogether, the update equation (Eq. 4.8) mimics Verlet integration [230], used to integrate Newton's equations of motion.

(a) CALIFORECASTING Prior Examples (b) KITTI Prior Examples

Figure 4.5: The prior penalizes positions corresponding to obstacles (white: high cost, black: low cost). The demonstrated expert trajectory is shown in each scene.

"Linear": The simplest model uses $\hat{\mu}_t^\pi, S_t$ linear in ψ_t:

$$\hat{\mu}_t^\pi(\psi_t) = Ah_t + b_0, \quad S_t(\psi_t) = Bh_t + b_1, \tag{4.13}$$

with $A \in \mathbb{R}^{2\times 2H}$, $h_t = x_{t-H:t-1} \in \mathbb{R}^{2H}$, $B \in \mathbb{R}^{4\times 2H}$, $b_i \in \mathbb{R}^{2H}$, and $S_t(\psi_t) \in \mathbb{R}^{2\times 2}$. To ensure positive-definiteness of σ_t^π, we use the matrix exponential [136]: $\sigma_t^\pi = \text{expm}(S_t + S_t^\top)$, which we found to optimize more efficiently than $\sigma_t^\pi = S_t S_t^\top$. Here, H is taken to be the maximum past size of 20. To enhance numerical stability, we "soft-clipped" the input S of $\text{expm}(S + S^\top)$ in the following elementwise transformation, which prevents σ_t^π from shrinking arbitrarily small. $\text{softclip}(S, L) = \frac{S}{\text{softmax}(1, \|S\|_F / L)}$. We used $L = 5$.

"Field": The Linear model ignores M: it has no environment perception. We designed a CNN model that takes in M and outputs $O \in \mathbb{R}^{H_{\text{map}} \times W_{\text{map}} \times 6}$. The 6 channels in O are used to form the 6 components of μ_t^π and S_t in the following way. To ensure differentiability, the values in O are bilinearly interpolated at the current rollout position, x_t in the spatial dimensions (H_{map} and W_{map}) of O. and Tab. 4.1 for the parameters of each layer. The input H and W dimensions are downsampled to 64×64, and the output dimensions are upsampled back to 100×100. In addition to the LIDAR and semantic segmentation channels used in the input, we added an $H \times W \times 1$ channel of grid (pixel) coordinates, an $H \times W \times 1$ channel of pixel distances to the car origin in

Table 4.1: R2P2 Field Architecture. The input H and W dimensions are downsampled to 64×64, and the output dimensions are upsampled back to 100×100. s^+ stands for the softplus layer: $s^+(x) = \log(1 + \exp x)$.

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13
Kernel Size	3	3	3	3	3	3	3	3	3	3	3	3	1
Dilation	1	1	1	1	2	4	8	4	2	1	1	1	1
Channels	32	32	32	32	32	32	32	32	32	32	32	32	6
Activation	s^+	s^+	s^+	s^+	s^+	s^+	s^+	s^+	s^+	s^+	tanh	tanh	Identity

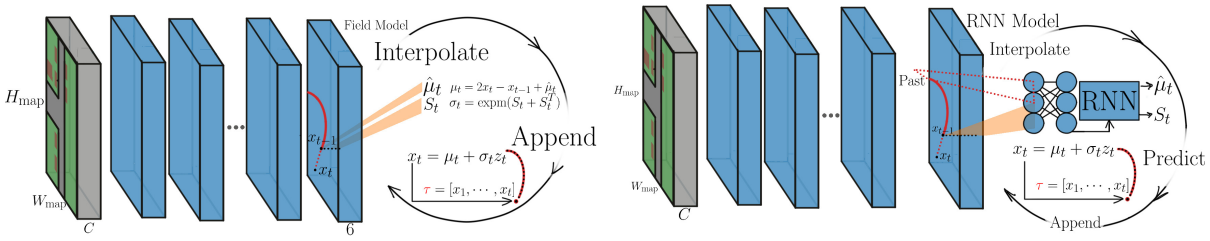


Figure 4.6: RNN and CNN Policy models. The Field model produces a map of values to use for producing μ^π, σ^π through interpolation. The RNN model uses the same base as the Field model as well as information from the past trajectory to decode a featurized context representation and previous state to next μ^π, σ^π.

pixel coordinates, (H/2, 0). Finally, a signed distance transform feature is added to the input, which takes the road channel from the segmentation as input, and outputs the signed distance to the road at each pixel. The final input array is of shape $H \times W \times (1 + C_s + 3)$, with $H = W = 50$, 1 layer for the LIDAR map, C_s channels for the semantic segmentation, and $C_s = 18$ in the CALIFORECASTING model, $C_s = 12$ for the KITTI model.

"RNN": The Linear and Field models reason with different contextual inputs: Linear uses the past, and CNN uses the feature map M. We developed a joint model to reason with both. M is passed through a CNN similar to Field's. The past is encoded with a GRU-RNN. Both featurizations inform a GRU-RNN that produces μ_t^π, S_t. See Figure 4.6 for a diagram of the architecture. The R2P2 RNN architecture builds upon the Field architecture, using the same CNN architecture settings until the last layer. The past encoding used is produced by a GRU RNN over the past states with 150 units. The last layer from the CNN architecture is flattened and concatenated with the past encoding to form the "static" contextual input, which is passed through a 2-layer MLP with 50 hidden and 50 output units, with softplus activations. The "dynamic" input, in the sense of its dependence on t, is formed from a fixed-length zero-padded vector of flattened previous states $x_1, \ldots, x_{t-1}$. This input is passed into a GRU RNN cell to "decode" $x_1 \ldots, x_T$ to a 150 dimensional vector, which is concatenated with the "static" output to form the "joint" feature, which is passed through independent MLPs to form the μ_t^π, σ_t^π.

4.3.3 GAIL and Differentiable GAIL

As a deep generative approach to imitation learning, our method is comparable to Generative Adversarial Imitation Learning (GAIL [83]). GAIL is model-free: it is agnostic to model dynamics. However, this flexibility requires an expensive model-free policy gradient method, whereas the approach we have proposed is fully differentiable. The model-free approach is significantly disadvantaged in sample complexity [96, 164] in theory and practice. By assuming the dynamics are

known and differentiable, as described in Sec. 4.3.1.1, we can also derive a version of GAIL that does not require model-free RL, since we can apply the reparameterization trick to differentiate the generator objective with respect to the policy parameters. A similar idea was explored for general imitation learning in [17]. We refer to this method as **R2P2 GAIL**. As our experiments show, R2P2 GAIL significantly outperforms standard GAIL, and our main model (R2P2) significantly outperforms and is easier to train than both GAIL and R2P2 GAIL.

4.3.3.1 R2P2 GAIL derivation

Formally, the GAIL policy seeks to optimize the following expected γ-discounted log-discriminator returns objective:

$$\max_\theta \mathbb{E}_{x\sim q_\theta} J(x) = \int dx q_\theta J(x) = \int dx q_\theta \sum_{t=1}^{T} \gamma^{t-1} \log D_\omega(x_t, a_t) \tag{4.14}$$

In GAIL, q_θ includes unknown model dynamics, thus $\nabla_\theta q_\theta$ cannot be computed. Optimization of Eq. 4.14 is done through the policy gradient shown in Eq 4.15. R2P2 GAIL *can* compute $\nabla_\theta q_\theta$ through its use of differentiable dynamics inside q_θ. The R2P2 GAIL gradient of Eq. 4.14 is shown in Eq. 4.16.

$$\mathbb{E}_{x\sim q_\theta}\left[J(x)\nabla_\theta \log \pi_\theta(a_t|x)\right] \tag{4.15}$$

$$\mathbb{E}_{z\sim q_0} \nabla_\theta J(g_\theta(z)) \tag{4.16}$$

Therefore, we can directly optimize the objective without relying on the indirect optimization approach of policy gradients. We verify in our experiments that our approach is stabler and achieves better performance.

The GAIL training objective may be expressed as follows [121]:

$$\min_\theta \max_\omega \mathbb{E}_{\pi_\theta} \log D_\omega(s,a) + \mathbb{E}_{\pi_E} \log(1 - D_\omega(s,a)) - \lambda H(\pi), \tag{4.17}$$

where (s,a) are understood to be drawn from the marginal state-action distributions associated with either the model policy π_θ or the expert policy π_E, as indicated by subscripts of $\mathbb{E}$. The following equivalent expression makes this more explicit:

$$\min_\theta \max_\omega \left[\mathop{\mathbb{E}}_{\substack{s_t\sim(q_{\pi_\theta})_t \\ a_t\sim(\pi_\theta)_t \\ t\sim\text{Unif}(\{1,\ldots,T\})}} \log D_\omega(s_t,a_t) + \mathop{\mathbb{E}}_{\substack{s_t\sim(q_{\pi_E})_t \\ a_t\sim(\pi_E)_t \\ t\sim\text{Unif}(\{1,\ldots,T\})}} \log(1 - D_\omega(s_t,a_t)) \right] - \lambda H(\pi_\theta),$$

where $(q_\pi)_t$ denotes the marginal distribution of states at time t induced by rolling-out policy π, $(\pi)_t$ denotes the expected distribution of actions at time t, and Unif represents the uniform distribution. From the perspective of the outer optimization (i.e., holding ω fixed), we observe that the objective function has a form commonly treated in reinforcement learning: $\min_\theta \mathbb{E}_{x\sim q_\pi} J_\pi(x)$, where $J_\pi(x)$ is the interior expression. Using the pushforward reparameterization, the outer optimization may be written as

$$\min_\theta \mathop{\mathbb{E}}_{\substack{z\sim q_b \\ t\sim\text{Unif}(\{1,\ldots,T\})}} \log D(g_{\pi_\theta}(z)_t, \pi_\theta(z)_t) - \lambda \log H(\pi_\theta(z)_t), \tag{4.18}$$

where $g_{\pi_\theta}(z)_t$ denotes the roll-out of policy π_θ with random noise sequence z, evaluated at time t; and $\pi_\theta(z)_t$ denotes the policy evaluated with z at time t. Here, we regard $\pi_\theta(z)$ as a deterministic

function that returns the action a_t given the random noise sequence z. Using the form of g_π, we have

$$\pi_\theta(z)_t := \mu_t^\theta(x_{1:t-1}, \phi) + \sigma_t^\theta(x_{1:t-1})z_t,$$

since we identify the states with the actions.

From Eq. equation 4.18, it is easy to see that the differentiability of g_{π_θ} wrt. θ trivially allows us to obtain a stochastic gradient of Eq. equation 4.18 wrt. θ by moving the derivative inside the integral.

4.4 Experiments

We implemented R2P2 and baselines with the primary aim of testing the following hypotheses. 1) The ability to exactly evaluate the model PDF should help R2P2 obtain better solutions than methods that do not use exact PDF inference (which includes GAIL). 2) The optimization of $H(p, q_\theta)$ should be correlated with the model's ability to cover the training data, in analogy to recall in binary classification. 3) Including $H(q_\theta, \tilde{p})$ in our objective should improve sample quality relative to methods without this term, as it serves a purpose analogous to precision in binary classification. 4) R2P2 GAIL will outperform GAIL through its more efficient optimization scheme.

4.4.1 The CALIFORECASTING Dataset

Current public datasets such as KITTI are suboptimal for the purpose of validating these hypotheses. KITTI is relatively small and was not designed with forecasting in mind. It contains relatively few episodes of subjectively interesting, nonlinear behavior. For this reason, we collected a novel dataset specifically designed for the ego-motion forecasting task, which we make public. The data is similar to KITTI in sensor modalities, but the data was collected so as to maximize the number of intersections, turning, and other subjectively interesting episodes. The data was collected with a sensor platform consisting of a Ford Transit Connect van with two Point Grey Flea3 cameras mounted on the roof in a wide-baseline configuration, in addition to a roof-mounted Velodyne VLP16 LIDAR unit and an IMU. The initial version of the dataset consists of three continuous driving sequences, each about one hour long, collected in mostly suburban areas of northern California (USA). The data was post-processed to produce a collection of episodes in the previously described format. The overhead feature map was populated by pretraining a semantic segmentation network [219], evaluating it on the sequences, correlating them with the LIDAR point cloud, and binning the resulting semantic segmentation scores in addition to a height-above-ground plane feature. With a subsampling scheme of 2Hz, CALIFORECASTING consists of over 10,000 training, 1,200 validation and 1,200 testing examples. The KITTI splits, in comparison, are about 3,100 training, 140 validation, and slightly less than 500 test examples with a subsampling scheme of 1Hz.

4.4.2 Metrics and Baselines

4.4.2.1 Metrics

Our primary metrics are the cross-entropy distribution metrics $H(p, q_\theta)$ and $H(q_\theta, \tilde{p})$. Note that $H(p, q_\theta)$ is lower-bounded by the entropy of p, $H(p)$, by Gibbs' inequality. Subtracting this quantity (computing KL) would be ideal; unfortunately, since $H(p)$ is unknown, we simply report $H(p, q_\theta)$. We also note that cross-entropy is *not* coordinate-invariant: we use path coordinates in an ego-centric frame that is a rotation and translation away from UTM coordinates (in meters) and report cross-entropy values for path distributions in this frame.

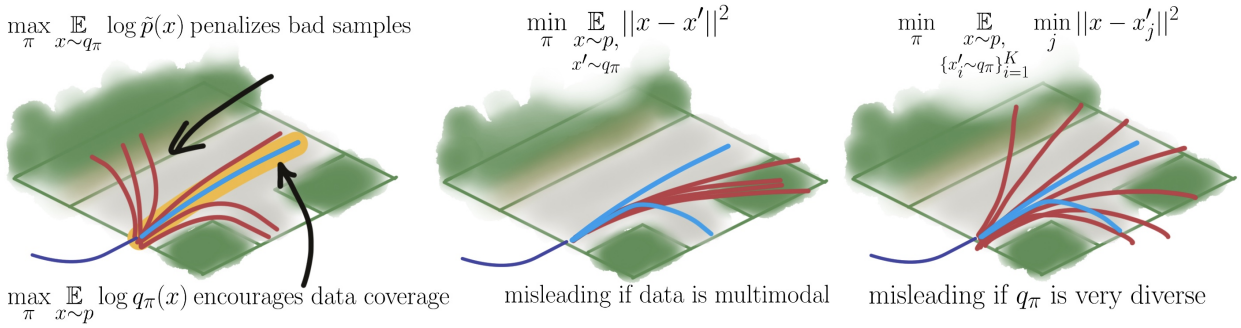


Figure 4.7: Possible objectives and their attributes. $\min_\theta H(p, q_\theta)$ encourages data coverage, $\min_\theta H(q_\theta, \tilde{p})$ penalizes bad samples. Measuring mean squared error is misleading when the data is multimodal, and measuring mean squared error of the best sample fails to measure quality of samples far from the demonstrations. This Figure is based on graphics created by Paul Vernaza.

A subtle related issue is that $H(p, q_\theta)$ may be unbounded below since $H(p)$ may be arbitrarily negative. This phenomenon arises when the support of p is restricted to a submanifold—for example, if for $x \sim p$ and $x_1 - x_2 = b$, the distribution $q(x) \propto \exp(-\|x_1 - x_2 - b\|^2/\epsilon^2 + \|x\|^2/2)$ achieves arbitrarily low values of $H(p, q_\theta)$. We resolve this by slightly perturbing training and testing samples from p: *i.e.* instead of computing $H(p, q_\theta)$, we compute $-\mathbb{E}_{\eta\sim\mathcal{N}(0,\epsilon I)}\mathbb{E}_{x\sim p} \log q(x + \eta)$ for $\epsilon = 0.001$. This is lower-bounded by $H(\mathcal{N}(0, \epsilon I))$, which resolves the issue.

4.4.2.2 Lower-bounding the perturbed cross-entropy

The naïve cross-entropy metric $-\mathbb{E}_{x\sim p} \log q(x)$ is unbounded below when the entropy of p is unbounded below. This happens in practice, for example, when the data deterministically satisfies some linear or nonlinear constraint. To avoid this problem, we employ a perturbed version of the cross-entropy metric for both training and evaluation. Specifically, we use the metric

$$-\mathbb{E}_{\eta\sim\mu}\mathbb{E}_{x\sim p} \log q(x - \eta). \tag{4.19}$$

In practice, we choose the perturbation distribution μ to be Gaussian with zero mean and covariance ϵI for simplicity.

We now show that the value of the perturbed cross-entropy is lower-bounded by the entropy of the perturbation distribution, which eliminates singularity of the metric, as long as the perturbation distribution has finite entropy. We first note that Eq. equation 4.19 can be written as $H(\tilde{p}, q)$, where $\tilde{p}$ is the convolution of p with μ: $\tilde{p}(x) \triangleq \int p(y)\mu(x - y)\,\mathrm{d}y = \mathbb{E}_{y\sim p}\mu(x - y)$. Gibbs' inequality implies $H(\tilde{p}, q) \geq H(\tilde{p})$, so it is sufficient to show that $H(\tilde{p}) \geq H(\mu)$. Observe that

$$H(\tilde{p}) = -\int \mathbb{E}_{y\sim p}\mu(x - y) \log \mathbb{E}_{y\sim p}\mu(x - y)\,\mathrm{d}x \tag{4.20}$$

$$\geq -\mathbb{E}_{y\sim p} \int \mu(x - y) \log \mu(x - y)\,\mathrm{d}x \tag{4.21}$$

$$= -\mathbb{E}_{y\sim p} \int \mu(z) \log \mu(z)\,\mathrm{d}z \tag{4.22}$$

$$= H(\mu), \tag{4.23}$$

where the inequality results from applying Jensen's inequality to the entropy function (which is concave), and we subsequently applied the change of variables $z = x - y$. Note that we have

applied Jensen's inequality in the following way:

$$E_{y\sim p}H(\mu(\cdot - y)) \leq H(E_{y\sim p}\mu(\cdot - y)), \tag{4.24}$$

where $\mu(\cdot - y)$ denotes the distribution $\tilde{\mu}(x|y) := \mu(x - y)$.

4.4.2.3 Cross-entropy is not coordinate-invariant

Some care must be taken when reporting cross-entropy values because cross-entropy is not coordinate-invariant; this implies that the same model will achieve different cross-entropy values depending on what units the data are expressed in, for example. Fortunately, it is fairly simple to compute how the cross-entropy changes under coordinate transformations. Suppose $p : X \to \mathbb{R}^+$ and $q : X \to \mathbb{R}^+$ are distributions on X, and $f : X \to Z$ is a differentiable, invertible coordinate transformation from X to Z. We wish to compute $H(f_*p, f_*q)$ in terms of $H(p, q)$, where $f_*\mu$ represents the pushforward measure of μ under the map f. Using the notation $\mathrm{d}f_x$ to represent the Jacobian of f evaluated at point x, direct computation shows:

$$H(f_*p, f_*q) = -\int p(f^{-1}(z))|\det \mathrm{d}f_{f^{-1}(z)}|^{-1} \log\left(q(f^{-1}(z))|\det \mathrm{d}f_{f^{-1}(z)}|^{-1}\right) \mathrm{d}z \tag{4.25}$$

$$= -\int p(x) \log(q(x))|\det \mathrm{d}f_x|^{-1} \mathrm{d}x \tag{4.26}$$

$$= H(p, q) + \mathbb{E}_{x\sim p} \log|\det \mathrm{d}f_x|, \tag{4.27}$$

where the second line follows from using f to change variables in the integral from z to x. For example, if $z = f(x) = cx$, and $x \in \mathbb{R}^d$, then $\mathbb{E}_{x\sim p} \log|\det \mathrm{d}f_x| = d \log|c|$. Therefore, we could make $H(f_*p, f_*q)$ arbitrarily negative by setting c to a very small positive number.

We include two commonly used sample metrics [8, 24, 77, 113, 188], despite the shortcomings illustrated in Fig. 4.7. We measure the quality of the "best" sample from K samples from q_θ: $\hat{X}$, relative to the demonstrated sample x via $\mathbb{E}_{\hat{X}_k\sim q_\theta} \min_{\hat{x}\in\hat{X}_k} \|x - \hat{x}\|^2$ (known as "minMSD"). This metric fails to measure the quality of *all* of the samples, and thus can be exploited by an approach that predicts samples that are mostly poor. Additionally, we measure the mean distance to the demonstration of all samples in $\hat{X}$: $\frac{1}{K}\sum_{k=1}^{K} \|x - \hat{x}_k\|^2$ (known as "meanMSD"). This metric is misleading if the data is multimodal, as the metric rewards predicting the mean, as opposed to covering multiple outcomes. *Due to the deficiencies of these common sample-based metrics for measuring the quality of multimodal predictions, we advocate supplementing sample-based metrics with the complementary cross-entropy metrics used in this work.*

4.4.2.4 Baselines.

We construct a simple a unimodal baseline: given the context, the distribution of trajectories is given as a sequence of Gaussian distributions. This is called the **Gaussian Direct Cross-Entropy (DCE-G)**. As discussed in Section 4.3.3, we apply **Generative Adversarial Imitation Learning (GAIL)**, along with our modified GAIL framework, R2P2 GAIL. We constructed several variants of GAIL: with and without the (improved) Wasserstein-GAN [12, 74] parameterization, with and without our novel **R2P2 GAIL** formulation, and using the standard MLP discriminator, versus a CNN-based discriminator with a similar architecture to the Field model. **Conditional Variational Autoencoders (CVAEs)** are a popular approach for modeling generative distributions conditioned on context. We follow the CVAE construction of [113] in our implementation. One key distinguishing factor is that CVAEs cannot perform *exact inference* by construction: given an arbitrary sample, a CVAE cannot

	MLP Discriminator			CNN MLP Discriminator	
Layer	1	2	3	1	2
Units	100	100	1	128	1
Activation	tanh	tanh	Identity or softmax	tanh	Identity or softmax

Table 4.2: GAIL Discriminators

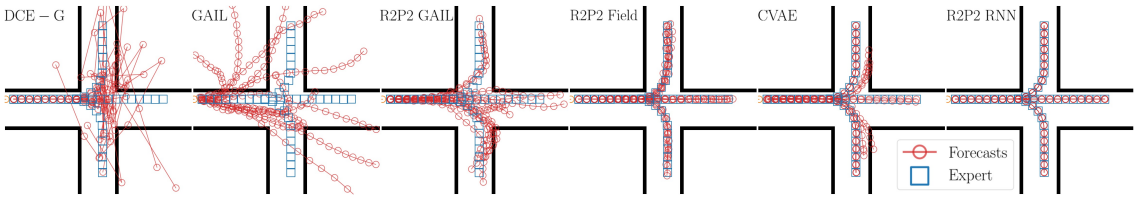


Figure 4.8: CROSS Trimodal Evaluation. *Top:* Qualitative results. *Bottom:* Quantitative results. A * indicates R2P2, and a † indicates using a WGAN Discriminator.

Approach	DCE-G	GAIL	GAIL†	CVAE [113]	GAIL*	GAIL*†	Linear*	Field*	RNN*
Test $-H(p, q_\theta)$	-0.005	14.10	16.71	–	40.79	36.53	59.20	98.99	**120.7**
Test minMSD	56.90	28.29	21.54	0.001	28.61	1.264	28.54	1.179	**0.0006**

produce a PDF value. Quantification of CVAE performance is thus required to be approximation-based, or sample-based. Our approaches are implemented in Tensorflow [1]. Architectural details of each approach are discussed below.

For the GAIL discriminator, we use the MLP form as depicted in Table 4.2. This is the original form of the GAIL Discriminator [83]. The final activation is Softmax in the case of the original formulation, and Identity in the case of the WGAN-variant.

In order to build a context-dependent discriminator (a function of the side information, ϕ), we used the same CNN architecture for the R2P2 Field model, except the output layer has 32 final channels. "State features" f_t are extracted from this output layer by the same bilinear interpolation process as in the Field model. These features are then passed into the MLP shown in Table 4.2. As before, the final activation is Softmax in the case of the original formulation, and Identity in the case of the WGAN-variant.

For DCE, we employ the same past-encoding technique as previously described. The past encoding state is passed into a GRU RNN decoder that takes a one-hot vector that indicates the t of the rollout. The decoder produces 5-dimensional vectors, with the first two components used to parameterized μ_t^π, and the latter three components used to parameterize the upper triangle of σ_t^π.

4.4.3 CROSS Trimodal Experiments

Our first set of experiments is designed to test the multimodal modeling capability of each approach in an easy domain. The contextual information is fixed – a single four-way intersection, along with three demonstrated outcomes: turning left, turning right, and going straight. Figure 4.8 shows qualitative and quantitative results. We see that several approaches fail to model multimodality well in this scenario. RNN. The models that can perform exact inference (all except CVAE) cover the modes with different success, as measured by Test $-H(p, q_\theta)$. We observe the models minimizing $H(p, q_\theta)$ cover the data well, supporting hypothesis 2 (coverage hypothesis), and outperform both GAIL approaches, supporting hypothesis 1 (exact inference hypothesis). We observe R2P2 GAIL outperforms GAIL in this scenario, supporting hypothesis 4 (optimization hypothesis). We also note

Table 4.3: CALIFORECASTING and KITTI evaluation, $K = 12$

CALIFORECASTING Approach	Test $-H(p, q_\theta)$	Test minMSD	Test meanMSD	Test $-H(q_\theta, \tilde{p})$
DCE-G	-1.604 ± 0.02	4.953 ± 0.18	11.66 ± 0.27	-129.2 ± 0.43
GAIL-WG [83]	27.43 ± 0.03	9.117 ± 0.27	36.77 ± 2.50	-221.5 ± 2.40
CVAE [113]	$\approx 10.1 \pm 0.9$	1.680 ± 0.12	9.961 ± 0.25	-122.2 ± 0.48
R2P2 GAIL-WG	45.55 ± 0.07	5.529 ± 0.33	25.12 ± 0.80	-152.1 ± 1.00
R2P2 GAIL-WG CNN	43.55 ± 0.08	4.937 ± 0.26	26.59 ± 0.96	-154.3 ± 1.20
R2P2 Linear	64.02 ± 0.11	2.339 ± 0.14	10.51 ± 0.39	-144.5 ± 1.00
R2P2 Linear $\beta = 0.1$	61.57 ± 0.10	2.387 ± 0.13	11.27 ± 0.44	-134.1 ± 0.76
R2P2 Field	54.56 ± 0.11	2.171 ± 0.13	11.59 ± 0.39	-142.5 ± 0.75
R2P2 Field $\beta = 0.1$	53.88 ± 0.11	2.162 ± 0.11	10.87 ± 0.39	-132.8 ± 0.54
R2P2 RNN	$\mathbf{70.20} \pm 0.11$	$\mathbf{1.530} \pm 0.12$	11.25 ± 0.29	-125.0 ± 0.53
R2P2 RNN $\beta = 0.1$	66.89 ± 0.12	1.860 ± 0.14	10.68 ± 0.30	$-\mathbf{119.0} \pm 0.44$
R2P2 RNN $\gamma = 1.0$	65.12 ± 0.12	1.661 ± 0.11	$\mathbf{8.542} \pm 0.22$	-124.8 ± 0.48
KITTI Approach	Test $-H(p, q_\theta)$	Test minMSD	Test meanMSD	Test $-H(q_\theta, \tilde{p})$
DCE-G	-1.884 ± 0.03	6.217 ± 0.30	15.20 ± 0.62	-137.0 ± 0.72
GAIL-WG [83]	39.53 ± 0.11	5.517 ± 0.34	20.08 ± 2.00	-188.8 ± 1.76
CVAE [113]	$\approx 9.22 \pm 0.9$	1.436 ± 0.15	9.593 ± 0.52	-133.8 ± 1.21
R2P2 GAIL-WG	47.45 ± 0.16	4.062 ± 0.25	13.80 ± 1.10	-168.9 ± 1.50
R2P2 GAIL-WG CNN	42.49 ± 0.12	4.601 ± 0.30	19.87 ± 1.34	-164.2 ± 1.43
R2P2 Linear	62.39 ± 0.14	2.438 ± 0.16	16.16 ± 1.26	-163.4 ± 1.50
R2P2 Linear $\beta = 0.1$	63.82 ± 0.16	2.587 ± 0.15	28.33 ± 1.40	-151.1 ± 1.40
R2P2 Field	64.71 ± 0.18	1.717 ± 0.13	10.34 ± 0.59	-139.2 ± 1.10
R2P2 Field $\beta = 0.1$	62.79 ± 0.29	1.639 ± 0.13	10.92 ± 0.59	$-\mathbf{126.9} \pm 0.77$
R2P2 RNN	$\mathbf{67.70} \pm 0.20$	1.574 ± 0.15	10.46 ± 0.57	-131.6 ± 0.91
R2P2 RNN $\beta = 0.3$	65.80 ± 0.21	$\mathbf{1.282} \pm 0.09$	$\mathbf{9.352} \pm 0.55$	-130.8 ± 0.87

the failure of DCE-G: its unimodal model is too restrictive for covering the diverse demonstrated behavior.

4.4.4 CALIFORECASTING Experiments and KITTI Experiments

We conducted larger-scale experiments designed to test our hypotheses. First, we trained $\tilde{p}$ on each dataset by the procedure described in Sec. 4.3.1.3. As discussed, our goal was to develop a simple model to minimize overfitting: we used a 3-layer Fully Convolutional NN. In the resulting spatial "cost" maps, we observe the model's ability to perceive obstacles in its assignment of low cost to on-road regions, and high-cost to clearly visible obstacles (*e.g* Fig. 4.5). We performed hyperparameter search for each method, and report the mean and its standard error of test set metrics corresponding to each method's best validation loss in Table 4.3. These results provide us with a rich set of observations. Of the three baselines, none catastrophically failed, with CVAE most often generating the cleanest samples. Across datasets and metrics, our approach achieves performance superior to the three baselines and our improved GAIL approach. By minimizing $H(p, q_\theta)$, our approach results in higher Test $-H(p, q_\theta)$ than all GAIL approaches, supporting the coverage and optimization hypotheses. We find that by incorporating our prior with nonzero β, hypothesis 3 is supported: our model architectures can improve the quality of its samples as measured by the Test $-H(q_\theta, \tilde{p})$. We observe that our GAIL optimization approach yields higher Test $-H(p, q_\theta)$, supporting hypothesis 4. We plot means and its standard error of the minMSD metrics as a function of K in Fig 4.12 for all 3 datasets.

We also find that qualitatively, our approach usually generates the best samples with diversity along multiple paths and precision in its tendency to avoid obstacles. Fig. 4.10 illustrates results

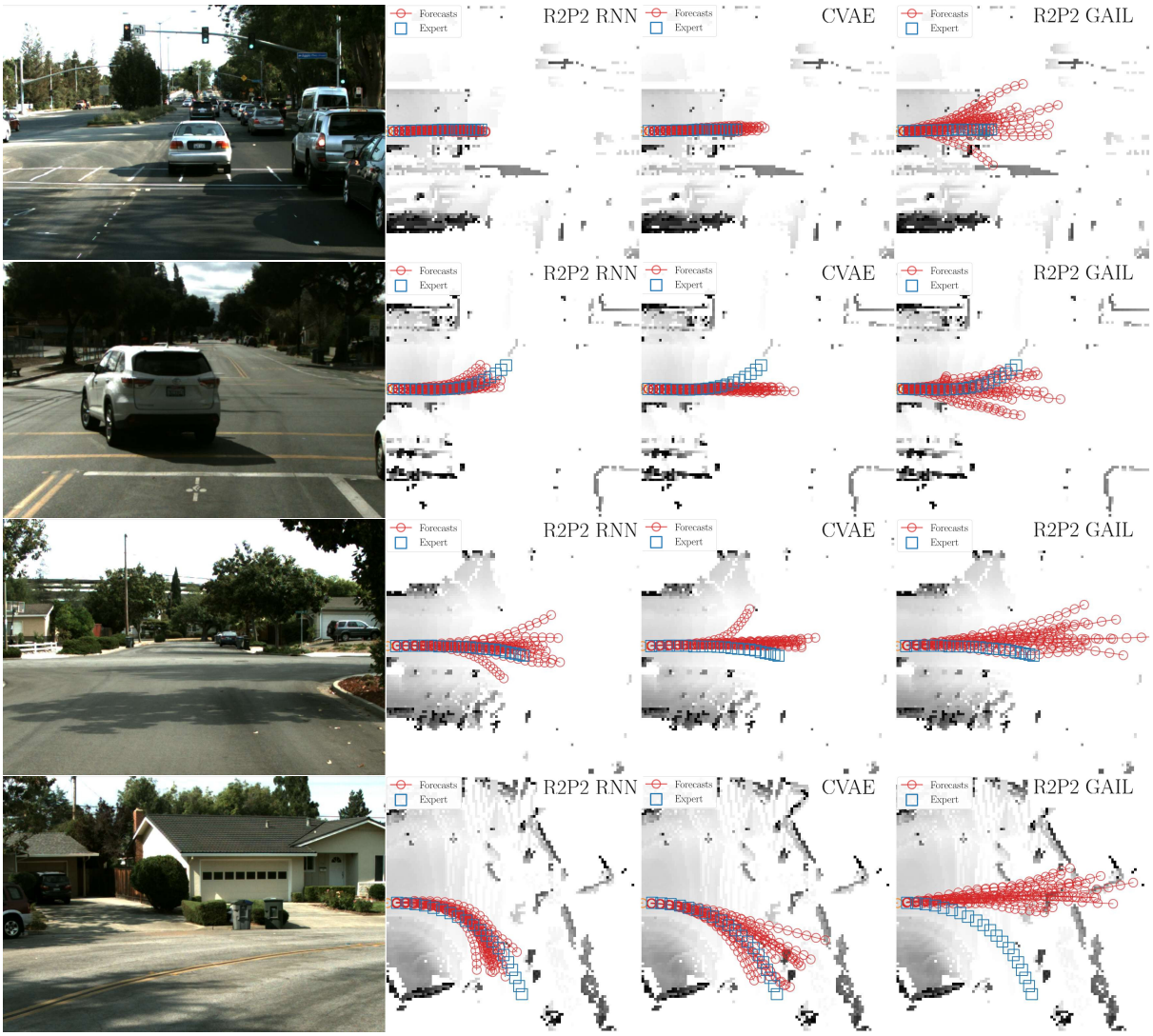


Figure 4.10: CALIFORECASTING Results. Comparison of R2P2 RNN (middle-left), CVAE (middle-right), and R2P2 GAIL (right). Trajectory samples are overlaid on overhead LIDAR map, colored by height. *Bottom two rows:* Comparison of $\beta = 0$ (top) and $\beta = 0.1$ (bottom), overlaid on $\tilde{p}$ cost map. The cost map improves sample quality.

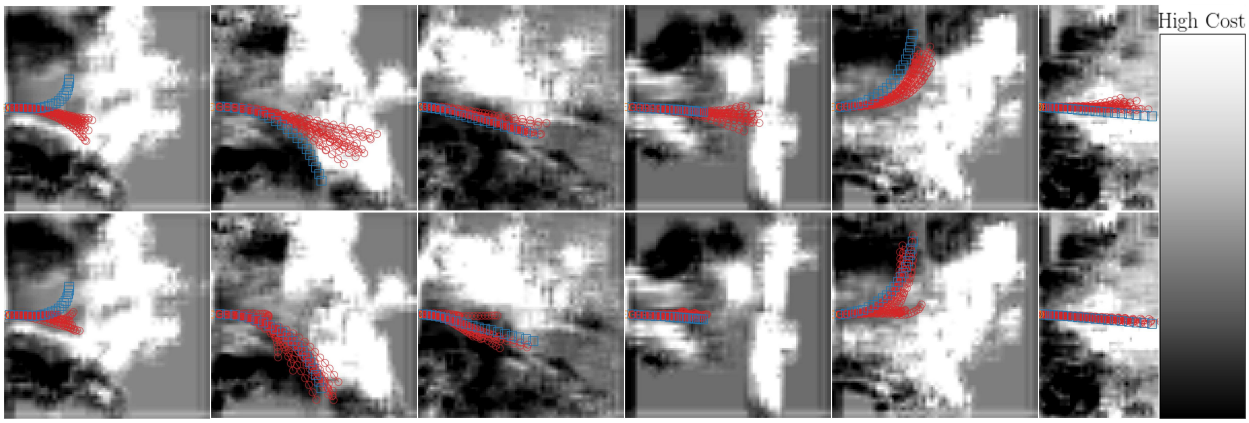


Figure 4.11: Comparison of using β on CALIFORECASTING test data. *Top row:* With $\beta = 0$, some trajectories are forecasted into obvious obstacles. *Bottom row:* With $\beta \neq 0$, many forecasted trajectories do not hit obstacles.

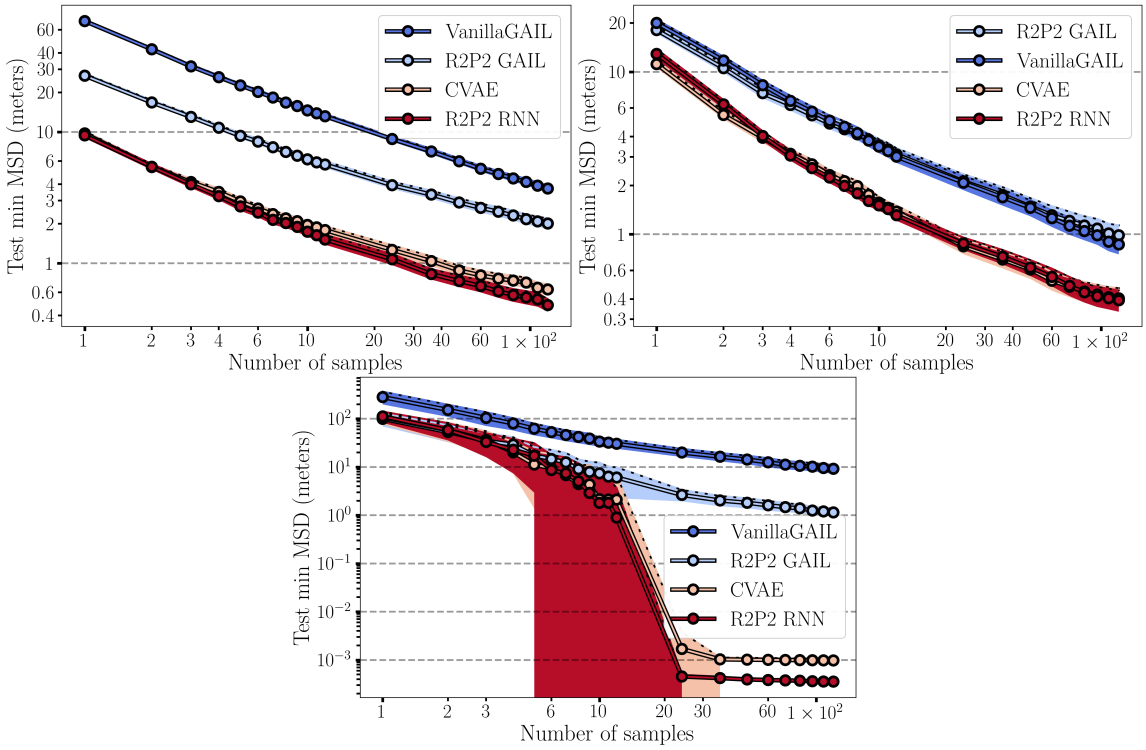


Figure 4.12: Test $\min_k$ MSD vs. K on CROSS, CALIFORECASTING, and KITTI.

on our dataset for our method, CVAE, and our improved GAIL approach. Fig. 4.11 illustrates qualitative examples for how incorporating nonzero β can improve sample quality.

4.5 Discussion

This work has raised the previously under-appreciated issue of balancing diversity and precision in probabilistic trajectory forecasting. We have proposed training a policy to induce a simulated-outcome distribution that minimizes a symmetrized cross-entropy objective. The key technical step that made this possible was a parameterizing the model distribution as the pushforward of a simple base distribution under the simulation operator. The relationship of this method to deep generative models was noted, and we showed that part of our full model enhances an existing deep imitation learning method. Empirically, we demonstrated that the pushforward parameterization enables reliable optimization of the objective, and that the optimized model has the desired characteristics of both covering the training data and generating high-quality samples. Finally, we introduced a novel large-scale, real-world dataset designed specifically for the vehicle ego-motion forecasting problem.

With respect to Chapter 2 and Chapter 3, we now have an approach to model finer-grained behavior than discrete actions. The ability to model high-dimensional sequences of continuous behavior is more general than the ability to either model a singular discrete action or a trajectory of discrete actions. Although discrete models can be made higher-dimensional with finer levels of discretization and classification, this requires choosing the level of discretization and finer-grained labels of behavior. It can be difficult to classify behaviors into discrete entities during labeling, as it requires drawing *subjective* boundaries between types of behavior. In contrast, recording the *pose* or the *positions* of the relevant entities is generally well-defined *objectively*. We believe that modeling the future poses of relevant entities is a more general, and therefore useful, problem to solve than modeling discrete sequences of behavior. Longer timescales at which it would make more sense to model symbolic behaviors are outside the scope of this thesis (e.g. consider planning a trip from one city to another).

One drawback of this work is that the approach for learning $\tilde{p}$ was somewhat heuristic. We now turn towards a new approach that continuously updates its approximation of p, similar to the style of other generative adversarial approaches.

4.6 Improving The Reverse KL Approximation

Although recent progress in GANs and variational methods have significantly advanced the capabilities of generative models for high-dimensional data, many issues still limit the practical application of such methods. In practice, GANs typically suffer from mode loss, whereas VAEs suffer from poor sample quality compared to GANs [12, 88, 243]. Several factors may be identified as contributing to these issues: first, the training loss may optimize for sample quality at the expense of mode coverage or vice-versa; second, the variance of stochastic gradient estimates may be too high to admit efficient stochastic optimization; finally, the models may not admit a good regularization scheme via the imposition of appropriate inductive biases.

We advocate a novel approach to address these concerns, with a particular focus on the latter issue of regularization. Let $p : \mathbb{R}^N \to \mathbb{R}^+$ denote the PDF of a continuous data distribution and $q : \mathbb{R}^N \to R^+$ denote the PDF of a learned model. Following recent work [172], we advocate training a generative model to minimize the *Jeffrey* (symmetric KL) divergence $\min_q \mathrm{KL}(p, q) + \mathrm{KL}(q, p)$, representing q in a way that enables it to be efficiently evaluated at any point (i.e., by representing

q as a *pushforward* distribution induced by an invertible warp [47, 97, 166]). In this work, we propose the key innovation of applying Fenchel-duality-based variational inference to $\mathrm{KL}(q, p)$, which allows the latter to be optimized without having to explicitly evaluate p. This yields the following variational approximation of the Jeffrey divergence, which constitutes the training loss for our method:

$$\min_{q \in \mathcal{Q}} \mathbb{E}_{\hat{x} \sim p} \log \frac{p(\hat{x})}{q(\hat{x})} + \sup_{\nu > 0} -\mathbb{E}_{\hat{x} \sim p} \frac{q(\hat{x})}{\nu(\hat{x})} + \mathbb{E}_{x \sim q} \log \frac{q(x)}{\nu(x)}, \tag{4.28}$$

where the variational parameters consist of the function ν. We now assert and later elaborate the following properties of the Jeffrey divergence and its variational approximation equation 4.28: first, $\mathrm{KL}(p, q)$ essentially prevents mode loss, whereas $\mathrm{KL}(q, p)$ prevents the generation of any samples unlikely under the data [25, 88]; second, an exactly unbiased stochastic gradient of $\mathrm{KL}(p, q)$ may be obtained without variational inference; finally, the optimal value of ν is p. The second key innovation of our work is to recognize that—in contrast to traditional GANs, which admit no comparable regularization principle—this model may be effectively regularized by imposing any domain-specific structure possessed by p on ν, which we thence interpret as a *structured Gibbs distribution*.

These concepts are illustrated in Fig. 4.13. Figure 4.13a shows the result of training a model q to optimize $\mathrm{KL}(p, q)$, where q is represented as the pushforward of a Gaussian base distribution under an autoregressive warp in two dimensions [47, 97, 166]. We observe that the generated samples effectively cover all the data modes, since a huge penalty is incurred if $q(x)$ is low for any data point $x \sim p$; however, the generator additionally places mass outside the support of the data distribution p, because shifting half of q's mass onto the support of p decreases $\mathrm{KL}(p, q)$ by no more than $\log 2$; therefore, this objective effectively under-constrains q. We can resolve the ambiguity by adding the variationally-approximated $\mathrm{KL}(q, p)$ term, which trains $\log \nu$ to approximate $\log p$ while penalizing the generation of samples where $\log \nu$ is low, as shown in Fig. 4.13b-d. By choosing different forms for $\log \nu$, we see that the ambiguity can be resolved in different ways: in Fig. 4.13b, $\log \nu$ is chosen to have contours roughly matching the support of p, whereas Fig. 4.13c-d show the cases where $\log \nu$ is represented as quadratic and as the output of a multi-layer perceptron, respectively. We see that matching the shape of $\log \nu$ to the shape of p produces qualitatively good samples, whereas putting too little constraint on the shape of $\log \nu$ (as in Fig. 4.13d) yields qualitatively worse results. Estimated cross-entropy values are also shown for each model, which again demonstrates how $\mathrm{KL}(p, q)$ is largely insensitive to the overall shape of the model q, as long as q covers the modes of p.

Our method may also be viewed as incorporating a kind of f-GAN [139] optimizing a particular loss (Jeffrey divergence), using a (invertible) generator that admits exact PDF evaluation, and reparameterizing the discriminator in a certain way. The Fenchel-variational view espoused in [139] reveals the optimal discriminator T to be a function of the odds ratio: $T^* = h(q/p)$ for some function h determined by the particular f-divergence chosen. We observe that if q can be evaluated analytically, then we might as well represent the discriminator in terms of q and some function ν, where ν directly approximates p. This change of representation allows us to take any known regularities of p and impose them on ν; e.g., if p is known to be translation-invariant, then we can safely impose translational invariance on ν without imposing any undue constraints on our model. Note that we cannot similarly impose p's structure on T, which may be a complex function even if p and q are simple: for example, even if p and q are both bounded above, T may be unbounded, since it is a function of the ratio of the two.

Fig. 4.14 illustrates how ν is structured for our application domain of vehicle trajectory forecasting: ν is represented as a sum of learned spatial rewards, which penalizes trajectories according to a

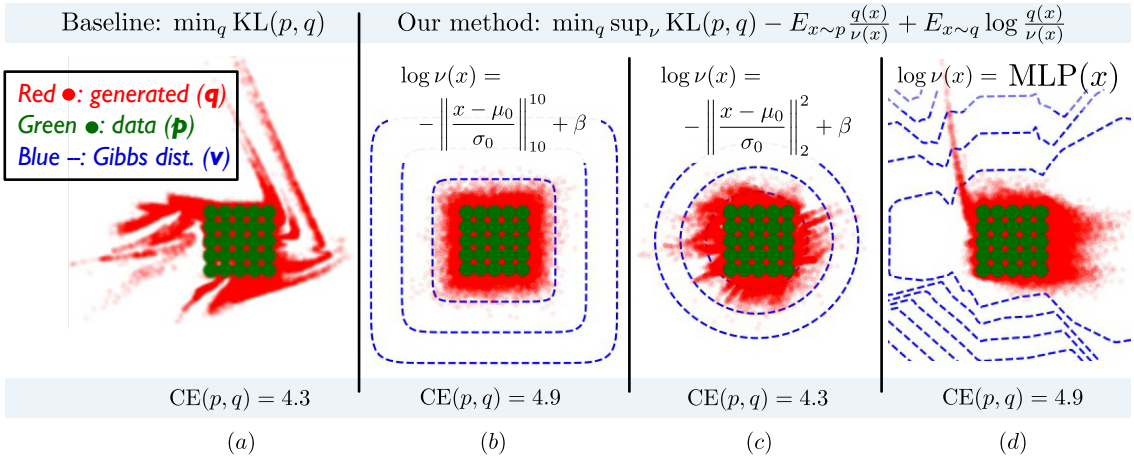


Figure 4.13: Different trained models for a 2D toy problem. The data distribution (samples shown as green dots) consists of a mixture of isotropic Gaussian distributions, arranged in a square grid. Four different trained models (generator samples shown as red dots) are shown: (a) result of training a model to optimize KL(p,q), (b-d) result of training models with our method (fine-tuning result of (a)), varying the form of the structured Gibbs distribution ν. Blue lines show contours of learned structured Gibbs distributions ν. In (b-c), the learned parameters of ν are μ_0, σ_0, and β

.

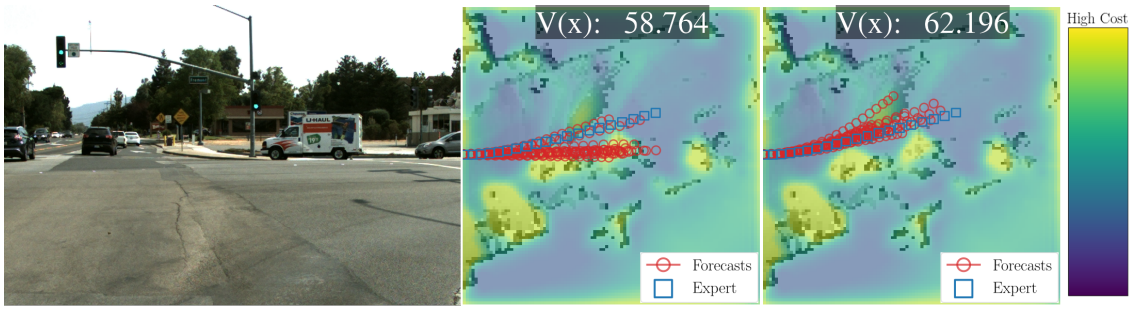


Figure 4.14: Our method applied to forecast ego-vehicle trajectories, showing input image and overhead views with the learned cost $-\log(\nu)$ (overlaid on LIDAR map). Middle shows samples from current q (red), and true future path (cyan). Note that ν has learned to penalize regions with obstacles ($V(x) = \log\nu(x)$, *i.e.* higher $V(x)$ is lower traversal cost). Right: after incorporating the learned ν, q and its samples are shifted to avoid high-cost regions, corresponding to suppression of spurious modes.

learned function over spatial positions. Intuitively, this prevents trajectories from colliding with obstacles, while simultaneously learning the concept of an obstacle.

4.7 Symmetric KL Learning Approach

Optimizing Eq. equation 4.28 is straightforward except for one subtle point: we must be able to evaluate q pointwise and differentiate the expression $\mathbb{E}_{x\sim q}\log(q(x)/\nu(x))$ with respect to the parameters of q. Inspired by prior work [47, 97, 166], we solve both these problems by representing q as the *pushforward* of a simple distribution under an invertible warp (also known as a normalizing flow). Suppose μ is a distribution over a set Z and $g : Z \to X$ is a function (the *generator*) with domain Z. Then we can define a measure on X as the distribution of $g(z)$ sampling z from μ—this distribution, denoted here by $g|_\mu$, is referred to as the pushforward of μ under g.

Now suppose g is parameterized by θ and differentiable in x and θ. By representing q as $q_\theta = g_\theta|_\mu$, for some simple distribution μ, we can move the derivative wrt. θ inside the expectation by exploiting the property $\mathbb{E}_{x\sim g_\theta|_\mu} f(x) = \mathbb{E}_{z\sim\mu} f(g_\theta(z))$ for all functions f:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\,\mathbb{E}_{x\sim q}\log\frac{q(x)}{\nu(x)} = \frac{\mathrm{d}}{\mathrm{d}\theta}\,\mathbb{E}_{x\sim q_\theta|_\mu}\log\frac{q(x)}{\nu(x)} = \mathbb{E}_{x\sim\mu}\,\frac{\mathrm{d}}{\mathrm{d}\theta}\log\frac{q_\theta(g_\theta(x))}{\nu(g_\theta(x))}. \tag{4.29}$$

This is well-known as the *reparameterization trick*; it allows us to obtain a low-variance, unbiased estimate of the parameter derivatives for learning with SGD. However, one problem remains: evaluating $q_\theta(x) = (g_\theta|_\mu)(x)$, which appears in both the first and last terms of equation 4.28. This is solved by assuming that g_θ is invertible: $\hat{z} := g_\theta^{-1}(\hat{x})$. Thus, we have an analytic formula for the pushforward density:

$$q_\theta(g_\theta(z)) = (g_\theta|_\mu)(g_\theta(z)) = \mu(z)\left|(\mathrm{d}g_\theta)_z\right|^{-1}, \tag{4.30}$$

where $|(\mathrm{d}g_\theta)_z|$ represents the determinant of the Jacobian of g_θ evaluated at the point z. This finally allows us to rewrite Eq. equation 4.28 in the following explicit form, after performing some simplifications:

$$-\max_\theta \inf_\phi \mathbb{E}_{\hat{x}\sim p}\log\frac{\mu(\hat{z})}{|(\mathrm{d}g_\theta)_{\hat{z}}|} + \frac{\mu(\hat{z})}{|(\mathrm{d}g_\theta)_{\hat{z}}|\nu_\phi(\hat{x})} + \mathbb{E}_{z\sim\mu}\log\frac{\mu(z)}{|(\mathrm{d}g_\theta)_z|\nu_\phi(g_\theta(z))}. \tag{4.31}$$

A pseudocode summary of our method is given in Algorithm 3. X and Z denote batches of observed and latent samples, respectively, while subscripts D and G denote either data or generated samples. A few implementation issues are noted here. Applying the method to a particular problem requires the implementation of the invertible generator $G(\cdot;\theta)$, the structured Gibbs energy $\log\nu(\cdot;\phi)$, and the base distribution μ. In order to avoid numerical issues when q is not absolutely continuous wrt. ν (i.e., when ν is 0 but q is not), we reparameterize ν as $\nu \leftarrow \alpha q + \nu$, where α is a small number. Given this assumption, the quantity $q(x)/(\alpha q(x)+\nu(x))$ can be rewritten in terms of the sigmoid σ. Optimization proceeds by alternating between minimizing the loss equation 4.28 in the generator parameters θ and maximizing it in the energy parameters ϕ. However, as noted in Alg. 3, we may alternatively minimize a different objective for the generator: namely, $-\mathbb{E}_{x\sim p}\log q(x) + \mathbb{E}_{x\sim q}\log(q(x)/\nu(x))$. The rationale for the alternative objective is that, as noted in Sec. 4.7.2, the inner minimization may be viewed as fitting ν to p—in which case, we may approximate $\mathrm{KL}(q,p)$ as $\mathbb{E}_{x\sim q}\log(q(x)/\nu(x))$. The alternate generator objective was used for the toy experiment in Fig. 4.13, whereas the original objective was used for the trajectory forecasting experiments.

Algorithm 3 Pseudocode for C3PO implementation

Require: X_D: a batch of training data, Z_G: batch of generator noise samples from μ

$Z_D, \det dg_{X_D}^{-1} \leftarrow G^{-1}(X_D;\theta)$ {$G^{-1}(x)$ returns $g_\theta^{-1}(x)$ and log det. of Jacobian of g_θ^{-1} at x}
$X_G, \det dg_{Z_G} \leftarrow G(Z_G;\theta)$ {$G(z)$ returns $g_\theta(z)$ and log det. of Jacobian of g_θ at z}
$\log q(X_D) \leftarrow \log \mu(Z_D) + \log|\det dg_{X_D}^{-1}|$ {Generator PDF at data samples}
$\log q(X_G) \leftarrow \log \mu(Z_G) + \log|\det dg_{Z_G}|$ {Generator PDF at generator samples}
$\log q/\nu(X_D) \leftarrow \log\alpha + \log q(X_D) - \log\nu(X_D;\phi)$
$\log q/\nu(X_G) \leftarrow \log\alpha + \log q(X_G) - \log\nu(X_G;\phi)$
$L \leftarrow \text{BatchMean}(-\log q(X_D) - \alpha^{-1}\sigma(\log q/\nu(X_D)) + \log\sigma(\log q/\nu(X_G)) - \log\alpha)$
for $i \leftarrow 1 \dots N$ **do**
  $\theta \leftarrow \theta - \beta\nabla_\theta L$ {Alternative generator loss: $L := -\log q(X_D) + \log\sigma(\log q/\nu(X_G))$}
  for $j \leftarrow 1 \dots M$ **do**
    $\phi \leftarrow \phi + \beta\nabla_\phi L$
  end for
end for

4.7.1 Derivations and Interpretations

Equation equation 4.28 can be derived via a variational lower bound derived from Fenchel conjugacy, using a technique similar to [138, 139]. Our approach of pairing this convex conjugate with the pushforward direct density estimation motivates our method's name: Convex Conjugate Coupled Pushforward Optimization (C3PO). Observe that the Jeffrey divergence can be written as $\min_q -\mathbb{E}_{\hat{x}\sim p}\log q(\hat{x}) + \mathbb{E}_{x\sim q}\log q(x) - \mathbb{E}_{x\sim q}\log p(x)$. Since q can be sampled, evaluated, and differentiated (Sec. 4.7), but p can only be sampled, our goal in this section is to convert $-\mathbb{E}_{x\sim q}\log p(x)$ to something expressable in terms of expectations wrt. p and q. This is achieved by applying the Fenchel-Young inequality to the function $f(p) := -\log p$, which yields $-\log p \geq \sup_{\lambda<0}\lambda p - (-1 - \log(-\lambda))$. Substituting this inequality in place of $-\log p(x)$ yields the following lower bound of $-\mathbb{E}_{x\sim q}\log p(x)$:

$$\int_X q(x)(\sup_{\lambda<0}\lambda p(x) + \log(-\lambda) + 1)\,\mathrm{d}x = 1 + \sup_{\lambda<0}\int_X q(x)\left(\lambda(x)p(x) + \log(-\lambda(x))\right)\mathrm{d}x, \tag{4.32}$$

where the sup on the right-hand side is taken over all functions $\lambda : X \to \mathbb{R}^-$ mapping the domain to a negative scalar. Observing that $\int q(x)\lambda(x)p(x)\,\mathrm{d}x$ can be written either as $\mathbb{E}_{x\sim q}\lambda(x)p(x)$ or $\mathbb{E}_{\hat{x}\sim p}\lambda(\hat{x})q(\hat{x})$, and making the substitution $\nu = -1/\lambda$, we have the equivalent bound

$$-\mathbb{E}_{x\sim q}\log p(x) \geq 1 + \sup_{\nu>0}\mathbb{E}_{\hat{x}\sim p} - \frac{q(\hat{x})}{\nu(\hat{x})} - \mathbb{E}_{x\sim q}\log\nu(x). \tag{4.33}$$

Substituting this the Jeffrey divergence yields Eq. equation 4.28.

4.7.2 Interpretation as Learning a Gibbs Distribution

Although we have so far viewed our goal as primarily learning q, we now observe that our method can also be interpreted primarily as a way of learning ν as a Gibbs distribution approximating p. Learning a Gibbs distribution, or a more general *energy-based model* [112], is a very general and effective way to impose strong domain-specific regularization on probability distributions over high-dimensional data. Generally, this is achieved by structuring the energy function $V_\phi := \log\nu_\phi$ to assign similar energies to similar examples; for example, a convolutional neural network might constitute a good energy for image generation, since the structure of a CNN encodes some degree

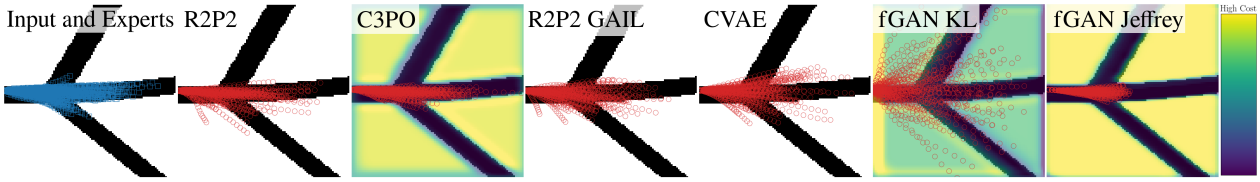


Figure 4.15: Comparison of methods on a test scene from the BEVWORLD1K dataset. Left column: The input BEV map (roads in black), accompanied by 100 demonstrations of possible behavior (in blue). Each method is visualized with 60 of each samples. For the methods that learn a cost map, the learned cost map is blended with the input BEV feature map.

of translational invariance. Unfortunately, inference and learning with high-dimensional Gibbs distributions is difficult.

Our method can be viewed as a way to train a Gibbs distribution that circumvents some of these difficulties. In this view, the inner optimization in Eq. equation 4.28 is interpreted as minimizing a weighted divergence between p and ν; the weights are exactly q. Specifically, we observe that the inner optimization in Eq. equation 4.28 minimizes the following weighted Itakura-Saito divergence [56, 90]:

$$\min_{\phi} \int q(x) \left(\frac{p(x)}{\nu_\phi(x)} - \log \frac{p(x)}{\nu_\phi(x)} - 1 \right) \mathrm{d}x \tag{4.34}$$

It would seem reasonable to choose the weights q as $q = p$; since p cannot be evaluated directly, alternating optimization is a sensible alternative. Minimizing Eq. equation 4.34 intuitively gives us the best Gibbs approximation of p over the support of q.

Intuition for the learning rule of ν can be obtained by computing the *functional gradient* of Eq. equation 4.28 with respect to ν (i.e., differentiating with respect to $\nu(x), \forall x$. The functional gradient ${}^{\delta}/_{\delta\nu}$ expresses the direction in which ν should be moved at each point x in order to optimally decrease the objective. Observing that $E_{x\sim p} q(x)/\nu(x) = E_{x\sim q} p(x)/\nu(x)$, we obtain the following for the functional gradient of the objective wrt. $\log \nu$: ${}^{\delta C}/_{\delta \log \nu}(x) = q(x) \left({}^{p(x)}/_{\nu(x)} - 1 \right)$. We therefore see that minimizing Eq. equation 4.28 in ν raises or lowers $\log \nu$ at each point x according to whether it exceeds $p(x)$, with a learning rate given by $q(x)$, and a unique fixed point (assuming $q > 0$) of $p = \nu$.

4.8 Symmetric KL Experiments and Discussion

We conducted experiments on four datasets of vehicle trajectories, where the objective is to forecast a distribution over the vehicle's future locations $x \in \mathbb{R}^{20\times 2}$ given contextual information about each scene. The contextual information includes 2 seconds of the vehicle's previous position, as well as a Bird's Eye View (BEV) map of visual scene features. Our experiments are designed to quantify two key aspects of generative modeling: the learned model's likelihood of held-out test data (*i.e.* the negative forward cross-entropy $-H(p, q)$), and the quality of samples from the learned model (*i.e.* the negative reverse cross-entropy $-H(q, p)$). Our hypotheses are: *1)* C3PO *will achieve superior sample-quality performance to other methods 2)* C3PO *will learn a high-quality q, partially due to its ability to perform direct density evaluation and optimization of q 3)* C3PO *will learn an interpretable ν that penalizes bad samples.*

Two of the four datasets are synthetic (BEVWORLD 1 and BEVWORLD1K), with known roads, enabling us to construct samples from a reasonable p distribution, approximate p via KDE, and measure the reverse cross-entropy $H(q, p_{\text{KDE}})$. We also calculate the percentage of trajectory points on the road as another measure of sample quality. Details of how we generate BEVWORLD are

Table 4.4: Comparison of methods in two datasets. Left: Single BEVWorld Scene (identical train and test), 300 experts, 1800 policy samples. Right: BEVWorld 100 training scenes, 1000 test scenes, 100 experts/scene, 12 policy samples/scene. Means and their standard errors are reported. Bold indicates the best performing method among methods with nondegenerate $-H(p,q)$ (*i.e.* fGAN Jeffrey is degenerate). CVAE's $-H(p,q)$ is estimated via MC sampling 300 times per scene, see [204].

	BEVWORLD 1			BEVWORLD1K		
Method	$-H(p,q)$	$-H(q,p_{\text{KDE}})$	Road %	$-H(p,q)$	$-H(q,p_{\text{KDE}})$	Road %
R2P2	$\mathbf{83.7} \pm 0.2$	-46.6 ± 0.5	0.988	$\mathbf{96.0} \pm 0.03$	-61.6 ± 0.6	0.922
C3PO (ours)	82.6 ± 0.3	$\mathbf{-44.8} \pm 0.4$	**1.000**	92.7 ± 0.1	$\mathbf{-48.7} \pm 0.3$	**0.989**
R2P2 GAIL	64.6 ± 2.0	-55.6 ± 1.0	0.952	62.0 ± 0.6	-74.6 ± 0.8	0.886
CVAE*	18.6 ± 3.7	-45.3 ± 1.7	0.990	12.5 ± 0.3	-71.3 ± 0.8	0.865
fGAN KL	16.6 ± 0.6	-294.4 ± 22	0.568	18.6 ± 0.03	-303 ± 2.6	0.698
fGAN Reverse KL	-17.3 ± 2.9	-171 ± 8.5	0.706	-11.6 ± 0.2	-235 ± 1.8	0.709
fGAN Jeffrey	$-7e4 \pm 6e3$	-42.0 ± 0.05	1.000	$-5e3 \pm 39$	-46.1 ± 0.08	0.970

Table 4.5: Comparison of methods in two real-world datasets: KITTI and CALIFORECASTING. Means and their standard errors are reported. Bold indicates the best performing method among methods with nondegenerate $-H(p,q)$ (*i.e.* fGAN Jeffrey is degenerate).

	KITTI		CALIFORECASTING	
Method	$-H(p,q)$	$V_\phi^{\text{KITTI}}(q)$	$-H(p,q)$	$V_\phi^{\text{CALIF}}(q)$
R2P2	$\mathbf{63.7} \pm 0.8$	-744 ± 20	$\mathbf{74.1} \pm 0.38$	-6.50 ± 3.9
C3PO (ours)	61.5 ± 0.7	$\mathbf{-457} \pm 13$	73.5 ± 0.4	$\mathbf{57.3} \pm 1.4$
R2P2 GAIL	54.9 ± 0.7	-693 ± 17	46.9 ± 0.3	-61.1 ± 6.1
CVAE*	9.22 ± 0.9	-555 ± 9.9	10.1 ± 0.9	48.3 ± 1.5
fGAN KL	32.9 ± 1.3	-693 ± 10	9.55 ± 0.02	-568 ± 20
fGAN Reverse KL	12.8 ± 0.08	-1362 ± 33	-89.7 ± 3.1	21.8 ± 2.6
fGAN Jeffrey	$-2e4 \pm 2e3$	-195 ± 4.0	$-2e4 \pm 7e2$	69.5 ± 0.1

in the supplement. We also experiment with two real-world datasets: the KITTI dataset, and the CALIFORECASTING dataset [173].

4.8.1 Implementation and Baselines

Given our setting is that of forecasting future vehicle trajectories, we leverage ideas from Inverse Reinforcement Learning [137, 244, 257], to structure our Gibbs energy V as a spatial *cost map*: where $\log \nu_\phi = V_\phi(x) = \sum_{t=1}^{T} R_\phi(x_{t0}, x_{t1}; \text{BEV})$, and $R_\phi(a, b; \text{BEV})$ is the output of a CNN that can be interpolated at 2d positions of the form (a, b). This structure enables ν to penalize trajectories that travel to locations it perceives to be bad, *e.g.* locations with obstacles, or locations far from a perceived road.

We compare our method, C3PO, to several state-of-the-art approaches in imitation learning and generative modeling: Generative Adversarial Imitation Learning (GAIL) [83], f-GAN [139], the CVAE method of DESIRE [113], and R2P2[173]. In each baseline, we use architectures as similar to our own method as possible: the policy (generator) architecture of GAIL and the generator architecture of f-GAN are identical to the generator architecture of our own approach. The same architecture used for V_ϕ in our method was also used for the discriminators in all baselines.

Our implementation of the q architecture is based on R2P2 [172]. One key difference between

our method, C3PO, and R2P2, is that R2P2 does not have an adversarial component; its main focus is learning the forward KL term, the first component of C3PO's objective function. R2P2 starts from a similar objective: the symmetric sum of cross-entropies. However, it relies on a cruder approximation of $H(q, p)$, because it learns an approximating $\tilde{p}$ for $H(q, \tilde{p})$ offline, with no interaction from q.

4.8.2 Synthetic Experiments

BEVWORLD 1contains a single scene with 300 samples from p. This setting is *unconditional*: the contextual information provided is identical, and not directly useful for modeling p. This setting also provides a fairer comparison to fGAN methods, which left the extension of fGAN to contextual settings as future work [139]. Table 4.4 shows the results of BEVWORLD 1 experiments. We observe that R2P2 and C3PO achieve the best $-H(p, q)$ scores, with C3PO outperforming all nondegenerate methods in its sample quality. This evidence supports hypotheses 1) and 2): C3PO achieves superior sample quality and high-quality data density. We also observed C3PO to be very stable throughout training in terms of $H(p, q)$: it was as easy to train it as R2P2 in all experiments.

We also observe that while fGAN KL indirectly learns a q with some support for p, its samples are quite poor. This matches expectations, as the forward KL divergence fails to impose much penalty on sample quality [88]. fGAN Jeffrey is the fGAN method most similar to our approach because it uses the Jeffrey divergence. We observe it to suffer mode collapse in all of our experiments, a similar result to [139], in which fGAN Jeffrey had the worse test-set likelihood.

Next, we consider BEVWORLD1K, in which 100 training scenes, each with 100 samples from p, are used to learn *conditional* generative models. There are 1000 scenes in the test set. Table 4.4 shows the result of these experiments. We observe results similar to BEVWORLD1K, again supporting hypotheses 1) and 2). C3PO can learn in the conditional setting to produce high-quality samples from a distribution with good support of p. We display example results in Fig. 4.15, and observe several methods, including C3PO , learn a cost-map representation that penalizes samples not on the road. This evidence of the learned intuitive perceptual measurement of ν supports hypothesis 3).

4.8.3 Real-world Experiments

We now experiment with real-world data, in which each method is provided with noisy contextual information, and is evaluated by its ability to produce a generative model and high-quality samples. Evaluation of sample quality is more difficult in this scenario: we have only one sample of p in each scene, precluding construction of p_{KDE}, and there are no labels of physical roads, precluding computation of on-road statistics. Fortunately, evaluation of $-H(p, q)$ is still possible.

After observing the learned V_ϕ of C3PO on both synthetic and real data, we found V_ϕ to be generally interpretable and intuitively good: it assigns low cost to roads it learns to perceive and it assigns high cost to obstacles it learns to perceive. We therefore employed our learned model V_ϕ to quantitatively evaluate samples from all methods. Fig. 4.16 illustrates results from several approaches visualized with V and point cloud features from the BEV, at various quantiles of each method's performance under V. Fig. 4.17 (left) illustrates each method on a set of randomly sampled scenes. In both figures, V perceives and assigns penalty to regions around obstacles in the point cloud data. Note that this energy is learned implicitly, because demonstrations from p avoid obstacles. These results provide further support for hypothesis 3).

Quantitative results are shown in Table 4.5. We find that results are, overall, similar to results on the other datasets. C3PO produces a high-performing q in terms of both its likelihood, $-H(p, q)$, as well as the quality of its samples, V. Additionally, we show evaluation of V for the top methods

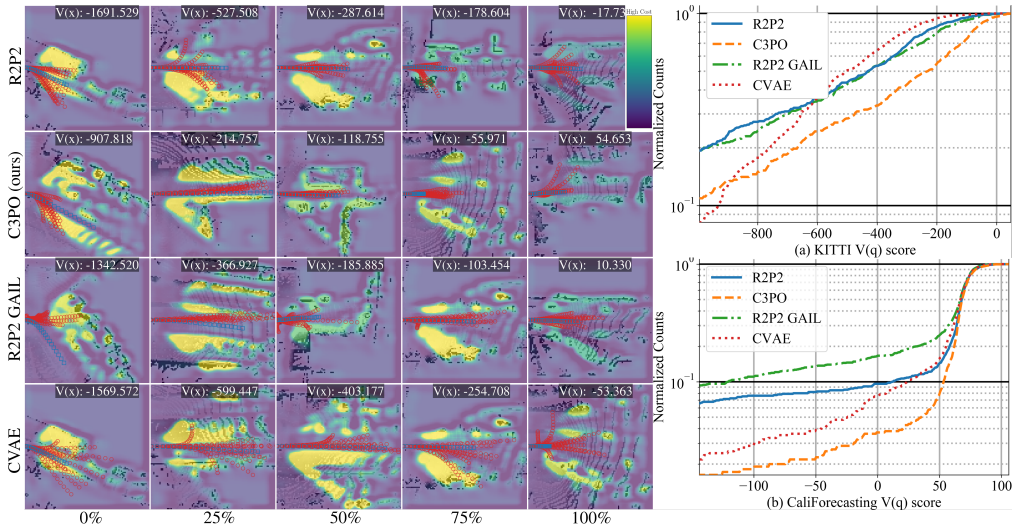


Figure 4.16: **Left:** Comparison of methods under the learned $V_\phi^{\text{KITTI}}(q)$ criterion. Each row corresponds to a method, and each column corresponds to the method's result on the item at a specific level of performance, from worst (left) to best (right) of results on 100 scenes. Each image is composed of the learned $V_\phi^{\text{KITTI}}(q)$ blended with the input BEV features, sample trajectories from each method (red), and the true future (blue). The learned V often penalizes samples that go off of the road or into obstacles inferred from the features. C3PO usually produces the best samples under this metric. **Right:** Evaluation of sample quality on test data from the KITTI dataset (a) and CALIFORECASTING dataset (b). Each approach generated 12 trajectory samples per scene, and the mean V_ϕ score was calculated for each scene. The results are displayed as a normalized cumulative histogram, where the y-value at x is the percentage of scenes that received $V \leq x$. At almost every given V, C3PO is likelier to have more samples above the given V than other methods.

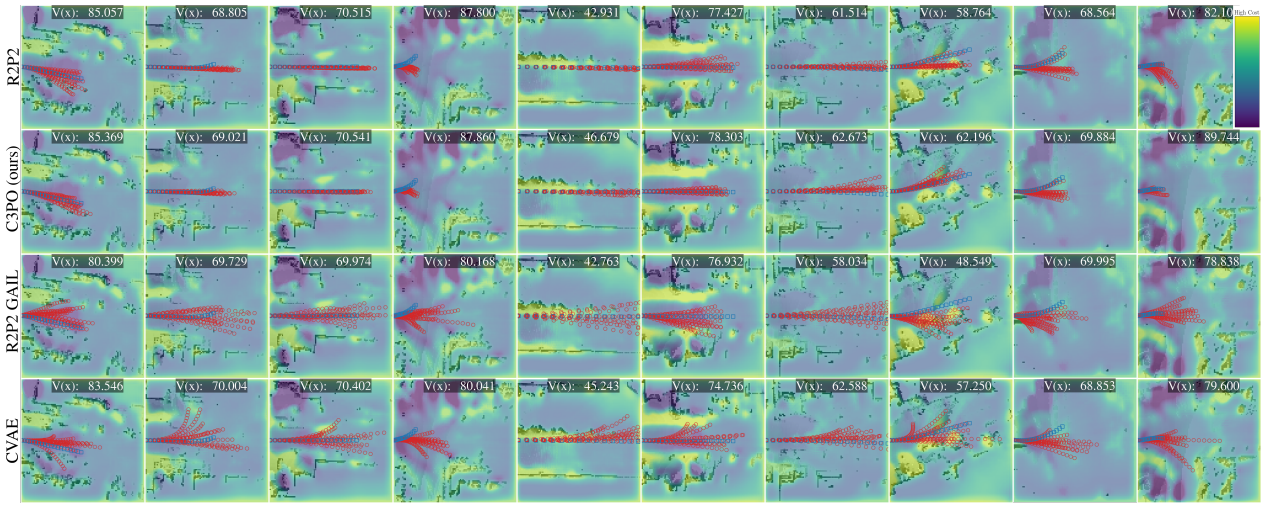


Figure 4.17: Comparison of methods on a set of random scenes from the CALIFORECASTING test set. Scene indices were selected uniformly at random (once) from the possible test indices. Each image shows the learned $V_\phi^{\text{CALIF}}(q)$ blended with the input BEV features, 12 sample trajectories from each method (red), and the true future (blue). V learns to penalize samples that go off of the road or into obstacles.

in Fig. 4.16 (right). Together, these results further strengthen evidence in support of all of our hypotheses.

The initial motivation for our work was the apparent dichotomy between sample quality and mode coverage in existing deep generative models; this phenomenon has been noted and quantified in work such as [19, 87, 88, 133, 243]. Our work synthesizes several techniques from prior work to address these issues, including the Fenchel-variational principle from work such as [138, 139], the well-known reparameterization trick [99, 194], and analytic pushforward-based density estimators such as normalizing flows [97, 166], RealNVP [47], and related models [64, 125, 142, 167, 224]. Comparatively little work has explored combining these methods, with some exceptions. Combining a RealNVP [47] density estimator with a GAN objective was considered in [71]; however, this is susceptible to the problems inherent with GANs mentioned in the introduction. A pushforward-based density estimator was employed in conjunction with variational inference to optimize the classical Bayesian evidence lower bound in [97], but this cross-entropy objective suffers from the previously-mentioned problems with optimizing $KL(p \parallel q)$ alone, as do all methods based on this objective, including [47, 64, 142, 224].

Our work is also comparable to the extensive literature on learning deep graphical models, including variants of Boltzmann machines [6, 189] and various proposals for learning deep CRFs and structured energy-based models [112] based on inference techniques including convex optimization [10, 36, 161, 196], various forms of unrolled inference [182, 237, 251], pseudolikelihood [124], and continuous relaxation [21], among other techniques [87]. Viewed as a way to learn a deep graphical model, the most important distinguishing factor of our work is the fact that our method does not need to perform inference directly, as discussed in Sec. equation 4.7.2. To the extent that q is viewed as (indirectly) performing inference, our method and other deep generative models bears some similarity to the wake-sleep algorithm, which also alternates between optimizing a model distribution and an "inference" distribution using complementary divergences [87]. Score-matching [89] also learns a Gibbs distribution without inference; however, unlike our method, score-matching does not learn an inference distribution.

4.9 Conclusion

We have demonstrated how GANs may be effectively regularized by reparameterizing the discriminator to be a function of the model density and a structured Gibbs density, showing how this may be achieved by optimizing a Jeffrey divergence loss, assuming the generator is invertible, and applying a variational bound based on Fenchel conjugacy. Applied to a trajectory forecasting problem, we observed superior mode coverage and qualitative results compared to traditional GANs. In contrast to the earlier part of this chapter, we developed a more-principled reverse KL approximation approach.

While this chapter and the previous chapters have addressed many types of forecasting, our motivation is to finally *integrate forecasting with control*. It is perhaps standard practice to consider these problems as *separate*, e.g. many autonomous driving companies separate their research efforts into "Perception/Prediction" module development (forecasting) and "Planning" model development (control). We posit that this is a *false dichotomy*: one of the main purposes forecasting is to inform control, therefore, forecasting and control should be *tightly integrated*. With a proper forecasting model, we can *anticipate likely outcomes*, and use this capability to *reason about how to make good outcomes likely and bad outcomes unlikely*. We want to design models to *jointly forecast and plan behaviors in the same framework*. This brings us to Part II.

Part II

Jointly Forecasting and Controlling from High-Dimensional Observations

Chapter 5

Forecasting Observations as Auxiliary Supervision for Implicitly-Planned Control

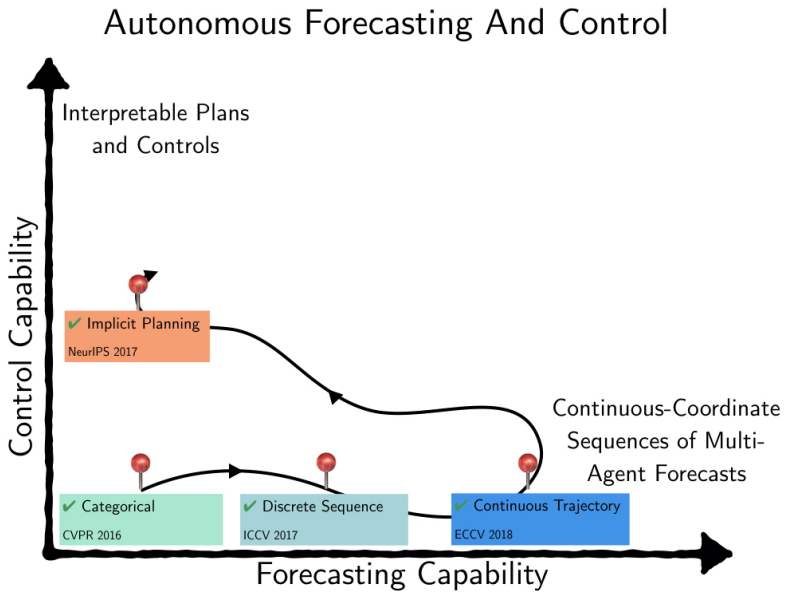


Figure 5.1: This chapter focuses on building a joint forecasting and control approach.

5.1 Introduction

Despite their wide success in a variety of domains, recurrent neural networks (RNNs) are often inhibited by the difficulty of learning an *internal state* representation. Internal state is a unifying characteristic of RNNs, as it serves as an RNN's memory. Learning these internal states is challenging because optimization is guided by the *indirect* signal of the RNN's target task, such as maximizing the cost-to-go for reinforcement learning or maximizing the likelihood of a sequence of words. These target tasks have a *latent state* sequence that characterizes the underlying sequential data-generating process. Unfortunately, most settings do not afford a parametric model of latent state that is available to the learner.

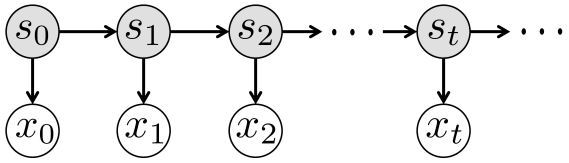
However, recent work has shown that in certain settings, latent states can be characterized by *observations* alone [26, 82, 86] – which are almost always available to recurrent models. In such partially-observable problems (*e.g.* Figure 5.2a), a single observation is not guaranteed to contain enough information to fully represent the system's latent state. For example, a single image of a robot is insufficient to characterize its latent velocity and acceleration. While a latent state parametrization may be known in some domains – *e.g.* a simple pendulum can be sufficiently modeled by its angle and angular velocity $(\theta, \dot{\theta})$ – data from most domains cannot be explicitly parametrized.

In lieu of ground truth access to latent states, recurrent neural networks [111, 216] employ internal states to summarize previous data, serving as a learner's memory. We avoid the terminology "hidden state" as it refers to the internal state in the RNN literature but refers to the latent state in the HMM, PSR, and related literature. Internal states are modified towards minimizing the target application's loss, e.g., minimizing observation loss in filtering or cumulative reward in reinforcement learning. The target application's loss is not directly defined over the internal states: they are updated via the chain rule (backpropagation) through the global loss. Although this modeling is indirect, recurrent networks nonetheless can achieve state-of-the-art results on many robotics [51, 80], vision [140, 143], and natural language tasks [38, 68, 162] when training succeeds. However, recurrent model optimization is hampered by two main difficulties: 1) non-convexity, and 2) the loss does not directly encourage the internal state to model the latent state. A poor internal state representation can yield poor task performance, but rarely does the task objective directly measure the quality of the internal state.

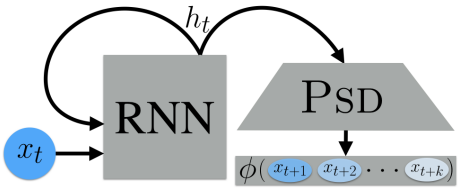
Predictive-State Representations (PSRs) [26, 82, 86] offer an alternative internal state representation to that of RNNs in terms of the available observations. Spectral learning methods for PSRs provide theoretical guarantees on discovering the global optimum for the model and internal state parameters under the assumptions of infinite training data and realizability. However, in the non-realizable setting – *i.e.* model mismatch (*e.g.*, using learned parameters of a linear system model for a non-linear system) – these algorithms lose any performance guarantees on using the learned model for the target inference tasks. Extensions to handle nonlinear systems rely on RKHS embeddings [205], which themselves can be computationally infeasible to use with large datasets. Nevertheless, when these models are trainable, they often achieve strong performance [82, 214]; the structure they impose significantly simplifies the learning problem.

We leverage ideas from the both RNN and PSR paradigms, resulting in a marriage of two orthogonal sequential modeling approaches. When training an RNN, PREDICTIVE-STATE DECODERS (Figure 5.2b) provide direct supervision on the internal state, aiding the training problem. The proposed method can be viewed as an instance of Multi-Task Learning (MTL) [35] and self-supervision [92], using the inputs to the learner to form a secondary unsupervised objective. Our contribution is a general method that improves performance of learning RNNs for sequential prediction problems. The approach is easy to implement as a regularizer on traditional RNN loss functions with little overhead and can thus be incorporated into a variety of existing recurrent models. We situate this method with respect to our other contributions in Fig. 5.1. By learning a policy whose representation predicts features of future observations, we develop an approach to *implicitly forecast and implicitly plan*.

In our experiments, we examine three domains where recurrent models are used to model temporal dependencies: probabilistic filtering, where we predict the future observation given past observations; Imitation Learning, where the learner attempts to mimic an expert's actions; and Reinforcement Learning, where a policy is trained to maximize cumulative reward. We observe that our method improves loss convergence rates and results in higher-quality final objectives in these domains.



(a) The process generating sequential data has latent state s_t which generates the next latent state s_{t+1}. s_t is usually unknown but generates the observations x_t which are used to learn a model for the system.



(b) An overview of our approach for modelling the process from Figure 5.2a. We attach a decoder to the internal state of an RNN to predict statistics of future observations x_t to x_{t+k} observed at training time.

Figure 5.2: Data generation process and proposed model

5.2 Latent State Space Models

To model sequential prediction problems, it is common to cast the problem into the Markov Process framework. Predictive distributions in this framework satisfy the Markov property:

$$P(s_{t+1}|s_t, s_{t-1}, \ldots, s_0) = P(s_{t+1}|s_t) \tag{5.1}$$

where s_t is the latent state of the system at timestep t. Intuitively, this property tells us that the future s_{t+1} is only dependent on the current state[1] s_t and does not depend on any previous state $s_0, \ldots, s_{t-1}$. As s_t is latent, the learner only has access to observations x_t, which are produced by s_t. For example, in robotics, x_t may be joint angles from sensors or a scene observed as an image. A common graphical model representation is shown in Figure 5.2a.

The machine learning problem is to find a model f that uses the latest observation x_t to recursively update an internal state, denoted h_t, illustrated in Figure 5.3. **Note that h_t is distinct from s_t. h_t is the learner's internal state, and s_t is the underlying configuration of the data-generating Markov Process**. For example, the internal state in the Bayesian filtering/POMDP setup is represented as a belief state [222], a "memory" unit in neural networks, or as a distribution over observations for PSRs.

Unlike traditional supervised machine learning problems, learning models for latent state problems must be accomplished without ground-truth supervision of the internal states themselves. Two distinct paradigms for latent state modeling exist. The first are discriminative approaches based on RNNs, and the second is a set of theoretically well-studied approaches based on Predictive-State Representations. In the following sections we provide a brief overview of each class of approach.

5.2.1 Recurrent Models and RNNs

A classical supervised machine learning approach for learning internal models involves choosing an explicit parametrization for the internal states and assuming ground-truth access to these states and observations at training time [43, 102, 117, 159]. These models focus on learning only the recursive model f in Figure 5.3, assuming access to the s_t (Figure 5.2a) at training time. Another class of approaches drop the assumption of access to ground truth but still assume a parametrization of the internal state. These models set up a multi-step prediction error and use expectation maximization to alternate between optimizing over the model's parameters and the internal state values [5, 40, 65].

While imposing a fixed representation on the internal state adds structure to the learning problem, it can limit performance. For many problems such as speech recognition [68] or text

[1] In Markov Decision Processes (MDPs), $P(s_{t+1}|s_t)$ may depend on an action taken at s_t.

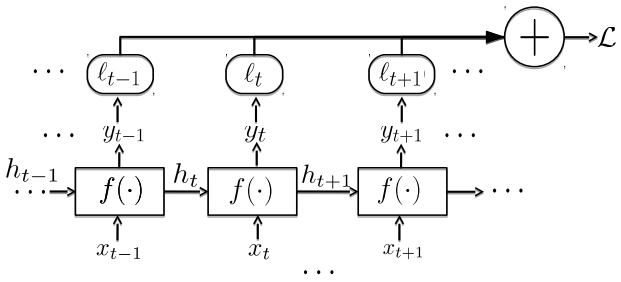


Figure 5.3: Learning recurrent models consists of learning a function f that updates the internal state h_t given the latest observation x_t. The internal state may also be used to predict targets y_t, such as control actions for imitation and reinforcement learning. These are then inputs to a loss function ℓ which accumulate as the multi-step loss $\mathcal{L}$ over all timesteps.

generation [217], it is difficult to fully represent a latent state inside the model's internal state. Instead, typical machine learning solutions rely on the Recurrent Neural Network architecture. The RNN model (Figure 5.3) uses the internal state to make predictions $y_t = f(h_t, x_t)$ and is trained by minimizing a series of loss functions ℓ_t over each prediction, as shown in the following optimization problem:

$$\min_f \mathcal{L} = \min_f \sum_t \ell_t(f(h_t, x_t)) \tag{5.2}$$

The loss functions ℓ_t are usually application- and domain-specific. For example, in a probabilistic filtering problem, the objective may be to minimize the negative log-likelihood of the observations [2, 226] or the prediction of the next observation [140]. For imitation learning, this objective function will penalize deviation of the prediction from the expert's action [180], and for policy-gradient reinforcement learning methods, the objective includes the log-likelihood of choosing actions weighted by their observed returns. In general, the task objective optimized by the network does not directly specify a loss directly over the values of the internal state h_t.

The general difficulty with the objective in Equation (5.2) is that the recurrence with f results in a highly non-convex and difficult optimization [5].

RNN models are thus often trained with backpropagation-through-time (BPTT) [239]. BPTT allows future losses incurred at timestep t to be back-propogated and affect the parameter updates to f. These updates to f then change the distribution of internal states computed during the next forward pass through time. The difficulty is then that small updates to f can drastically change the distribution of h_t, sometimes resulting in error exponential in the time horizon [227]. This "diffusion problem" can yield an unstable training procedure with exponentially exploding or vanishing gradients [22]. While techniques such as truncated gradients [216] or gradient-clipping [150] can alleviate some of these problems, each of these techniques yields stability by discarding information about how future observations and predictions should backpropagate through the current internal state. A significant innovation in training internal states with long-term dependence was the LSTM [85]. Many variants on LSTMs exist (*e.g.* GRUs [37]), yet in the domains evaluated by [69], none consistently exhibit statistically significant improvements over LSTMs.

In the next section, we discuss a different paradigm for learning temporal models. In contrast with the open-ended internal-state learned by RNNs, Predictive-State methods do not parameterize a specific representation of the internal state but use certain assumptions to construct a mathematical structure in terms of the observations to find a globally optimal representation.

5.2.2 Predictive-State Models

Predictive-State Representations (PSRs) address the problem of finding an internal state by formulating the representation directly in terms of observable quantities. Instead of targeting a prediction loss as with RNNs, PSRs define a belief over the distribution of k future observations, $g_t = [x_t^T, ..., x_{t+k-1}^T]^T \in \mathbb{R}^{kn}$ given all the past observations $p_t = [x_0, \dots x_{t-1}]$ [27]. In the case of linear systems, this k is similar to the rank of the observability matrix [14]. The key assumption in PSRs is that the definition of state is equivalent to having *sufficient* information to predict everything about g_t at time-step t [203], *i.e.* there is a bijective function that maps $P(s_t|p_{t-1})$ – the distribution of latent state given the past – to $P(g_t|p_{t-1})$ – the belief over future observations.

Spectral learning approaches were developed to find an globally optimal internal state representation and the transition model f for these Predictive-State models. In the controls literature, these approaches were developed as subspace identification [225], and in the ML literature as spectral approaches for partially-observed systems [26, 28, 86, 241]. A significant improvement in model learning was developed by [27, 82], where sufficient feature functions ϕ (e.g., moments) map distributions $P(g_t|p_t)$ to points in feature space $\mathbf{E}\left[\phi(g_t)|p_t\right]$. For example, $\mathbf{E}\left[\phi(g_t)|p_t\right] = \mathbf{E}\left[g_t, g_t g_t^T|p_t\right]$ are the sufficient statistics for a Gaussian distribution. With this representation, learning latent state prediction models can be reduced to supervised learning.

[82] used this along with Instrumental Variable Regression [29] to develop a procedure that, in the limit of infinite data, and under a linear-system realiziablity assumption, would converge to the globally optimal solution. [214] extended this setup to create a practical algorithm, Predictive-State Inference Machines (PSIMs) [212, 214, 229], based on the concept of inference machines [108, 182]. Unlike in [82], which attempted to find a generative observation model and transition model, PSIMs directly learned the filter function, an operator f, that can *deterministically* pass the predictive states forward in time conditioned on the latest observation, by minimizing the following loss over f:

$$\ell_p = \sum_t \left\|\phi(g_{t+1}) - f(h_t, x_t)\right\|^2, \quad h_{t+1} = f(h_t, x_t) \tag{5.3}$$

This loss function, which we call the *predictive-state loss*, forms the basis of our PREDICTIVE-STATE DECODERS . By minimizing this supervised loss function, PSIM assigns statistical meaning to internal states: it forces the internal state h_t to match sufficient statistics of future observations $\mathbf{E}\left[\phi(g_t)|p_t\right]$ at every timestep t. We observe an empirical sample of the future $g_t = [x_t, \dots, x_{t+k}]$ at each timestep by looking into the future in the training dataset or by waiting for streaming future observations. Whereas [214] primarily studied algorithms for minimizing the predictive-state loss, we adapt it to augment general recurrent models such as LSTMs and for a wider variety of applications such as imitation and reinforcement learning.

5.3 Predictive-State Decoders

Our PREDICTIVE-STATE DECODERS architecture extends the Predictive-State Representation idea to general recurrent architectures. We hypothesize that by encouraging the internal states to encode information sufficient for reconstructing the predictive state, the resulting internal states better capture the underlying dynamics and learning can be improved. The result is a simple-to-implement objective function which is coupled with the existing RNN loss. To represent arbitrary sizes and values of PSRs with a fixed-size internal state in the recurrent network, we attach a decoding module $F(\cdot)$ to the internal states to produce the resulting PSR estimates. Figure 5.4 illustrates our approach.

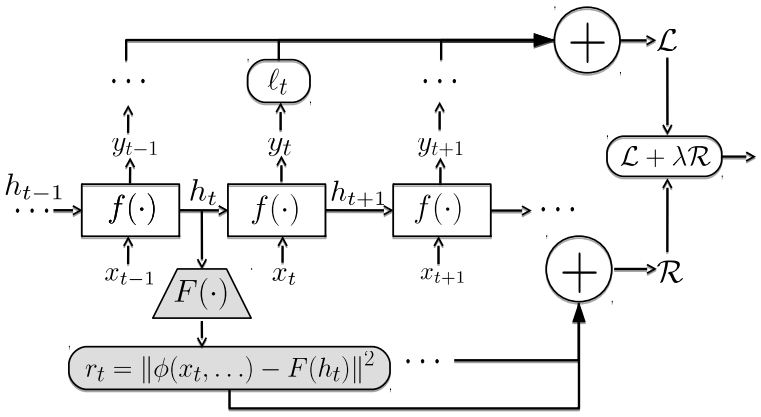


Figure 5.4: Predictive-State Decoders Architecture. We augment the RNN from Figure 5.3 with an additional objective function $\mathcal{R}$ which targets decoding of the internal state through F at each time step to the *predictive-state* which is represented as statistics over the future observations.

Our PSD objective $\mathcal{R}$ is the predictive-state loss:

$$\mathcal{R} = \sum_t \left\| F(h_t) - \phi([x_{t+1}, x_{t+2}, \ldots]) \right\|_2^2, \quad h_t = f(h_{t-1}, x_{t-1}), \tag{5.4}$$

where F is a decoder that maps from the internal state h_t to an *empirical sample* of the predictive-state, computed from a sequence of observed future observations available at training. The network is optimized by minimizing the weighted total loss function $\mathcal{L} + \lambda\mathcal{R}$ where λ is the weighting on the predictive-state objective $\mathcal{R}$. This penalty encourages the internal states to encode information sufficient for directly predicting sufficient future observations. Unlike more standard regularization techniques, $\mathcal{R}$ does not regularize the parameters of the network but instead regularizes the *output variables*, the internal states predicted by the network.

Our method may be interpreted as an instance of Multi-Task Learning (MTL) [35]. MTL has found use in recent deep neural networks [7, 92, 103]. The idea of MTL is to employ a shared representation to perform complementary or similar tasks. When the learner exhibits good performance on one task, some of its understanding can be transferred to a related task. In our case, forcing RNNs to be able to more explicitly reason about the future they will encounter is an intuitive and general method. Endowing RNNs with a theoretically-motivated representation of the future better enables them to serve their purpose of making sequential predictions, resulting in more effective learning. This difference is pronounced in applications such as imitation and reinforcement learning (Sections 5.4.2 and 5.4.3) where the primary objective is to find a control policy to maximize accumulated future reward while receiving only observations from the system. MTL with PSDs supervises the network to predict the future and implicitly the consequences of the learned policy. Finally, our PSD objective can be considered an instance of self-supervision [92] as it uses the inputs to the learner to form a secondary *unsupervised* objective.

As discussed in Section 5.2.1, the purpose of the internal state in recurrent network models (RNNs, LSTMs, deep, or otherwise) is to capture a quantity similar to that of state. Ideally, the learner would be able to back-propagate through the primary objective function $\mathcal{L}$ and discover the best representation of the latent state of the system towards minimizing the objective. However, as this problem highly non-convex, BPTT often yields a locally-optimal solution in a basin determined by the initialization of the parameters and the dataset. By introducing $\mathcal{R}$, the space of feasible models is reduced. We observe next how this objective leads our method to find better models.

5.4 Experiments

We present results on problems of increasing complexity for recurrent models: probabilistic filtering, Imitation Learning (IL), and Reinforcement Learning (RL). The first is easiest, as the goal is to predict the next future observation given the current observation and internal state. For imitation learning, the recurrent model is given training-time expert guidance with the goal of choosing actions to maximize the sequence of future rewards. Finally, we analyze the challenging domain of reinforcement learning, where the goal is the same as imitation learning but expert guidance is unavailable.

PREDICTIVE-STATE DECODERS require two hyperparameters: k, the number of observations to characterize the predictive state and λ, the regularization trade-off factor. In most cases, we primarily tune λ, and set k to one of $\{2, \ldots, 10\}$. For each domain, for each k, there were λ values for which the performance was worse than the baseline. However, for many sets of hyperparameters, the performance exceeded the baselines. Most notably, for many experiments, the convergence rate was significantly better using PSDs , implying that PSDs allows for more efficient data utilization for learning recurrent models.

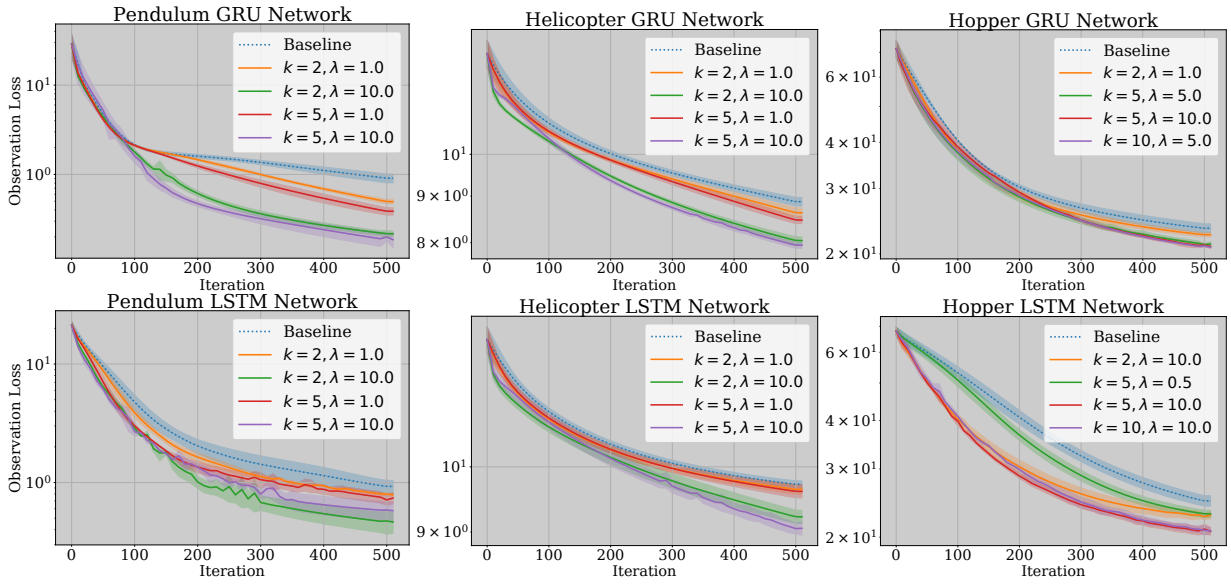
PSDs also require a specification of two other parameters in the architecture: the featurization function ϕ and decoding module F. For simplicity, we use an affine function as the decoder F in Equation (5.4). The results presented below use an identity featurization ϕ for the presented results but include a short discussion of second order featurization. We find that in each domain, we are able to improve the performance of the state-of-the-art baselines. We observe improvements with both GRU and LSTM cells across a range of k and λ. In IL with PSDs , we come significantly closer and occasionally eclipse the expert's performance, whereas the baselines never do. In our RL experiments, our method achieves statistically significant improvements over the state-of-the-art approach of [51, 195] on the 5 different settings we tested.

5.4.1 Probabilistic Filtering

In the probabilistic filtering problem, the goal is to predict the future from the current internal state. Recurrent models for filtering use a multi-step objective function that maximizes the likelihood of the future observations over the internal states and dynamics model f's parameters. Under a Gaussian assumption (e.g. like a Kalman filter [78]), the equivalent objective that minimizes the negative log-likelihood is given as $\mathcal{L} = \sum_t \left\| x_{t+1} - f(x_t, h_t) \right\|^2$.

While traditional methods would explicitly solve for parametric internal states h_t using an EM style approach, we use BPTT to implicitly find an non-parametric internal state. We optimize the end-to-end filtering performance through the PSD joint objective $\min_{f,F} \mathcal{L} + \lambda \mathcal{R}$. Our experimental results are shown in Figure 5.5. The experiments were run with ϕ as the identity, capturing statistics representing the first moment. We tested ϕ as second-order statistics and found while the performance improved over the baseline, it was outperformed by the first moment. In all environments, a dataset was collected using a preset control policy. In the Pendulum experiments, we predict the pendulum's angle θ. The LQR controlled Helicopter experiments [4] use a noisy state as the observation, and the Hopper dataset was generated using the OpenAI simulation [31] with robust policy optimization algorithm [154] as the controller.

We test each environment with Tensorflow's built-in GRU and LSTM cells [1]. We sweep over various k and λ hyperparameters and present the average results and standard deviations from runs with different random seeds. Figure 5.5 baselines are recurrent models equivalent to PSDs with $\lambda = 0$.



(a) Pendulum (b) Helicopter (c) Hopper

Figure 5.5: Loss over predicting future observations during filtering. For both RNNs with GRU cells (*top*) and with with LSTM cells (*bottom*), adding PSDs to the RNN networks can often improve performance and convergence rate.

5.4.2 Imitation Learning

We experiment with the partially observable CartPole and Acrobot domains[2] from OpenAI Gym [31]. We applied the method of AggreVaTeD [215], a policy-gradient method, to train our expert models. AggreVaTeD uses access to a cost-to-go oracle in order to train a policy that is sensitive to the value of the expert's actions, providing an advantage over behavior cloning IL approaches. The experts have access to the *full* state of the robots, unlike the learned recurrent policies.

We tune the parameters of LSTM and GRU agents (e.g., learning rate, number of internal units) and afterwards only tune λ for PSDs . In Figure 5.6, we observe that PSDs improve performance for both GRU- and LSTM-based agents and increasing the predictive-state horizon k yields better results. Notably, PSDs achieves 73% relative improvement over baseline LSTM and 42% over GRU on Cartpole. Difference random seeds were used. The cumulative reward of the current best policy is shown.

5.4.3 Reinforcement Learning

Reinforcement learning (RL) increases the problem complexity from imitation learning by removing expert guidance. The latent state of the system is heavily influenced by the RL agent itself and changes as the policy improves. We use [51]'s implementation of TRPO [195], a Natural Policy Gradient method [95]. Although [195] defines a KL-constraint on policy parameters that affect actions, our implementation of PSDs introduces parameters (those of the decoder) that are unaffected by the constraint, as the decoder does not directly govern the agent's actions.

In these experiments, results are highly stochastic due to both environment randomness and nondeterministic parallelization of `rllab` [51]. We therefore repeat each experiment at least 15 times with paired random seeds. We use $k = 2$ for most experiments ($k = 4$ for Hopper), the identity

[2]The observation function only provides positional information (including joint angles), excluding velocities.

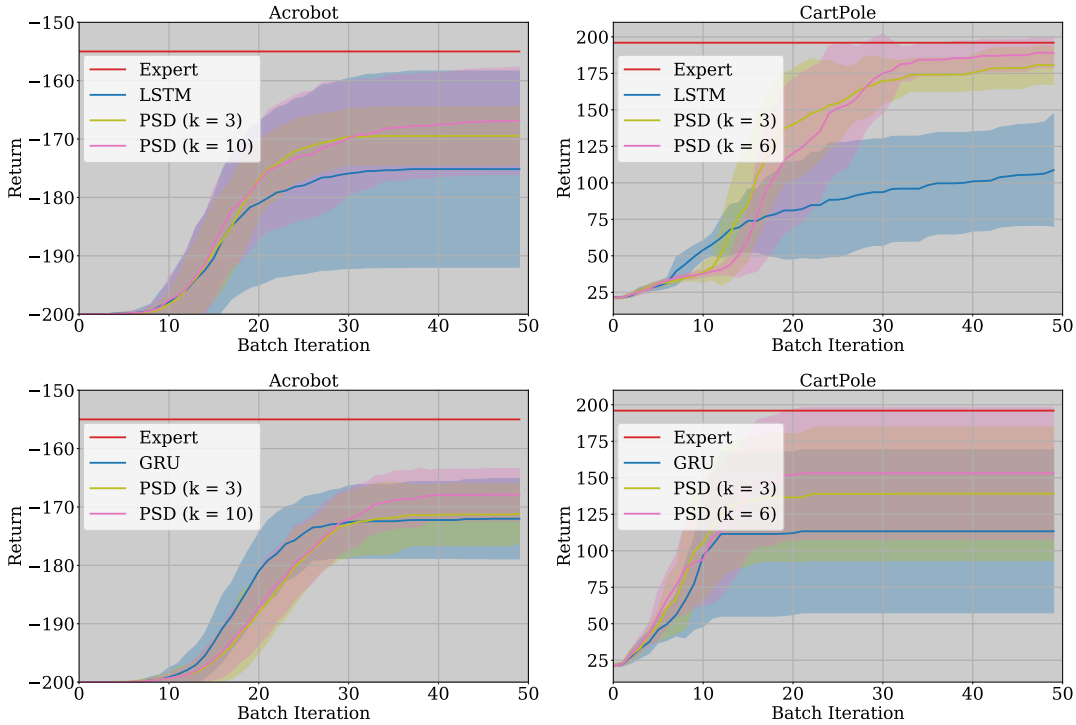


Figure 5.6: Cumulative rewards for AggreVaTeD and AggreVaTeD+PREDICTIVE-STATE DECODERS on partially observable Acrobot and CartPole with both LSTM cells and GRU cells averaged over 15 runs with different random seeds.

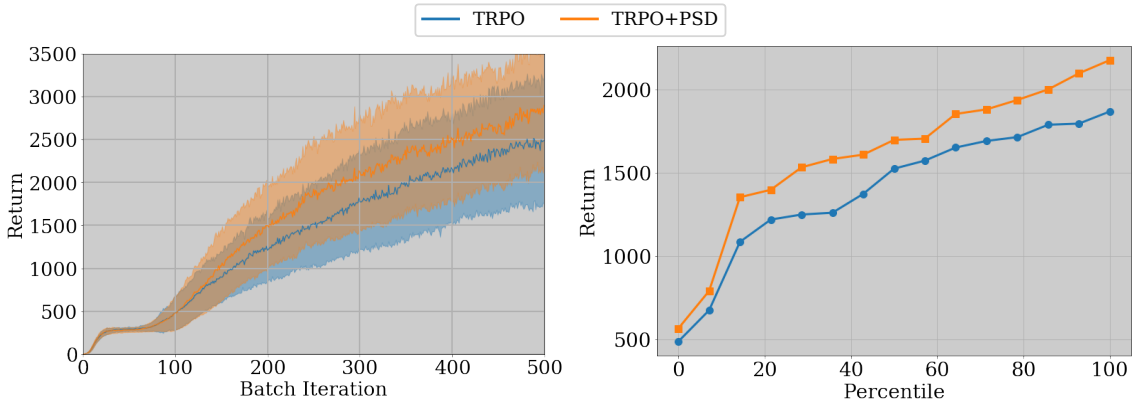


Figure 5.7: Walker Cumulative Rewards and Sorted Percentiles. $N = 15$, $5e4$ TRPO steps per iteration.

featurization for ϕ, and vary λ in $\{10^1, 10^0, \ldots, 10^{-6}\}$, and employ the LSTM cell and other default parameters of TRPO. We report the same metric as [51]: per-TRPO batch average return across learning iterations. Additionally, we report per-run performance by plotting the sorted average TRPO batch returns (each item is a number representing a method's performance for a single seed).

Figures 5.7 and 5.8 demonstrate that our method generally produces higher-quality results than the baseline. These results are further summarized by their means and stds. in Table 5.1. In Figure 5.7, 40% of our method's models are better than the best baseline model. In Figure 5.8c, 25% of our method's models are better than the second-best (98^{th} percentile) baseline model. We

Table 5.1: *Top:* Mean Average Returns $\pm$ one standard deviation, with $N = 15$ for Walker2d† and $N = 30$ otherwise. *Bottom*: Relative improvement of on the means. * indicates $p < 0.05$ and ** indicates $p < 0.005$ on Wilcoxon's signed-rank test for significance of improvement. All runs computed with $5e3$ transitions per iteration, except Walker2d†, with $5e4$.

	Swimmer	HalfCheetah	Hopper	Walker2d	Walker2d†
[195]	91.3 ± 25.5	330 ± 158	1103 ± 264	383 ± 96	1396 ± 396
[195]+PSDs	$\mathbf{97.0 \pm 19.4}$	$\mathbf{372 \pm 143}$	$\mathbf{1195 \pm 272}$	$\mathbf{416 \pm 88}$	$\mathbf{1611 \pm 436}$
Rel. Δ	$6.30\%^*$	$13.0\%^*$	$9.06\%^*$	$8.59\%^*$	$15.4\%^{**}$

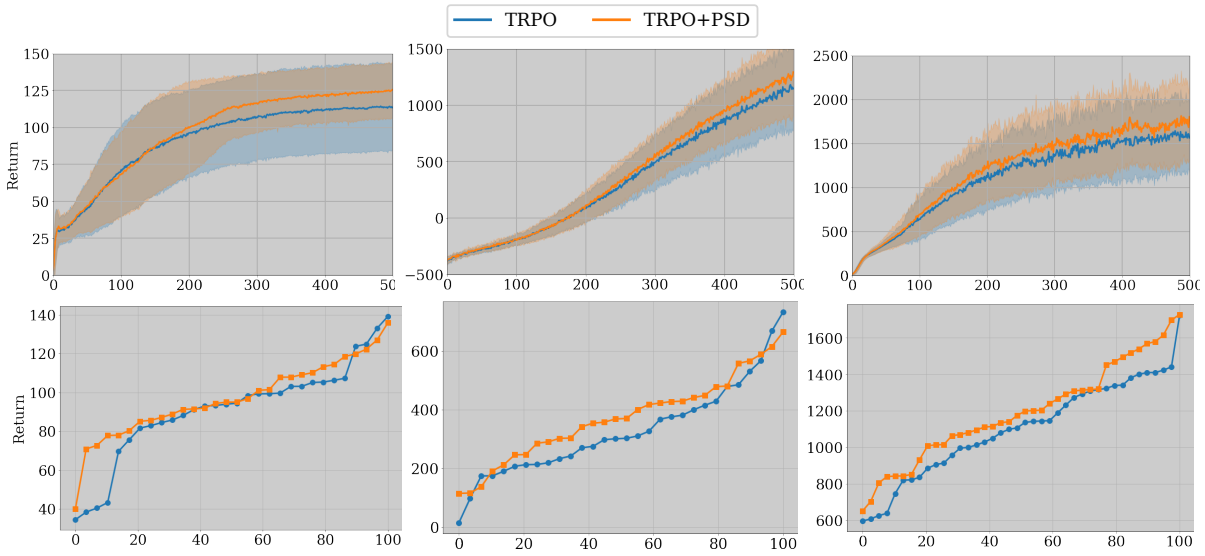
Table 5.2: Variations of RNN units. Mean Average Returns $\pm$ one standard deviation, with $N = 20$. $1e3$ transitions per iteration are used. Our method can improve each recurrent unit we tested.

	InvertedPendulum			Swimmer		
	Basic	GRU	LSTM	Basic	GRU	LSTM
[195]	$\mathbf{820 \pm 139}$	673 ± 268	640 ± 265	66.0 ± 21.4	64.6 ± 55.3	56.5 ± 23.8
[195]+PSDs	820 ± 118	$\mathbf{782 \pm 183}$	$\mathbf{784 \pm 215}$	$\mathbf{71.4 \pm 26.9}$	$\mathbf{75.1 \pm 28.8}$	$\mathbf{61.0 \pm 23.8}$
Rel. Δ	-0.08%	20.4%	22.6%	8.21%	16.1%	7.94%

compare various RNN cells in Table 5.2, and find our method can improve Basic (linear + `tanh` nonlinearity), GRU, and LSTM RNNs, and usually reduces the performance variance. We used Tensorflow [1] and passed both the "hidden" and "cell" components of an LSTM's internal state to the decoder. We also conducted preliminary additional experiments with second order featurization ($\phi(x) = [x, \text{vec}(xx^T)]$). Corresponding to Tab. 5.2, column 1 for the inverted pendulum, second order features yielded 861 ± 41, a 4.9% improvement in the mean and a large reduction in variance.

5.5 Conclusion

We introduced a theoretically-motivated method for improving the training of RNNs. Our method stems from previous literature that assigns statistical meaning to a learner's internal state for modelling latent state of the data-generating processes. Our approach uses the objective in PSIMs and applies it to more complicated recurrent models such as LSTMs and GRUs and to objectives beyond probabilistic filtering such as imitation and reinforcement learning. We show that our straightforward method improves performance across all domains with which we experimented. In Part I of this thesis we focused on progressively designing more capable models to forecast. In this chapter, we discussed a first attempt to jointly forecast and control. However, the two main downsides to this approach are that its predictions of the future are generally *not interpretable*, and that it is restricted to *implicitly plan* its future behavior. These two downsides make it difficult to interrogate and validate the model's understanding of the future. We instead seek to *explicitly plan controls* and *explicitly forecast* behavior.



(a) Swimmer, N=30 (b) HalfCheetah, N=30 (c) Hopper, N=40

Figure 5.8: *Top:* Per-iteration average returns for TRPO and TRPO+PREDICTIVE-STATE DECODERS vs. batch iteration, with $5e3$ steps per iteration. *Bottom:* Sorted per-run mean (across iterations) average returns. Our method generally produces better models.

Chapter 6

Forecasting Motion Trajectories for Explicitly-Planned Control

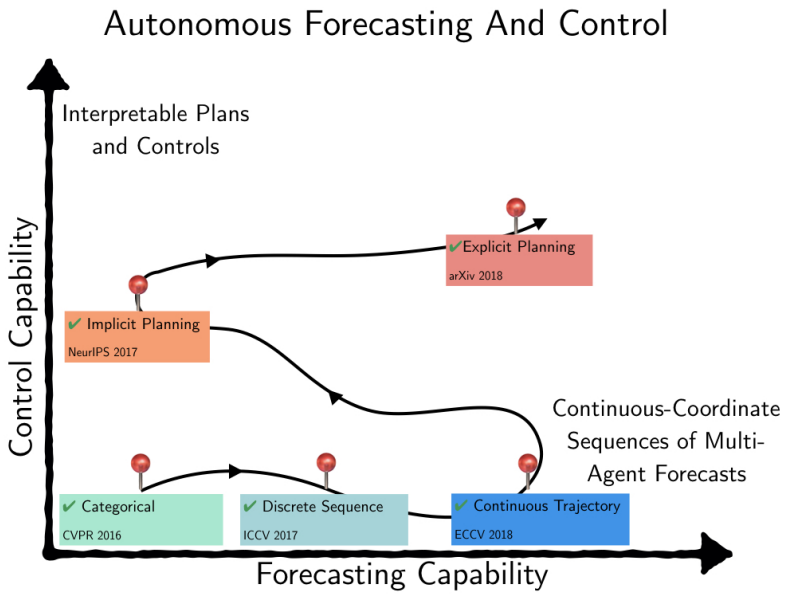


Figure 6.1: This chapter focuses on building a joint forecasting and control approach capable of explicitly planning and interpretably forecasting.

6.1 Introduction

Imitation learning (IL) is a framework for learning a model to mimic behavior. At test-time, the model pursues its best-guess of desirable behavior. By letting the model choose its own behavior, we cannot *direct* it to achieve different goals. While work has augmented IL with goal conditioning [41, 49], it requires goals to be specified during training, explicit goal labels, and are simple (e.g., turning). In contrast, we seek flexibility to achieve *general* goals for which we have *no demonstrations*.

In contrast to IL, planning-based algorithms like model-based reinforcement learning (MBRL) methods do not require expert demonstrations. MBRL can adapt to new tasks specified through reward functions [44, 106]. The "model" is a dynamics model, used to *plan* under the user-supplied reward function. Planning enables these approaches to perform new tasks at test-time. The key

drawback is that these models learn dynamics of *possible* behavior rather than dynamics of *desirable* behavior. This means that the responsibility of evoking desirable behavior is entirely deferred to engineering the input reward function. Designing reward functions that cause MBRL to evoke complex, desirable behavior is difficult when the space of possible *undesirable* behaviors is large. In order to succeed, the rewards cannot lead the model astray towards observations significantly different than those with which the model was trained.

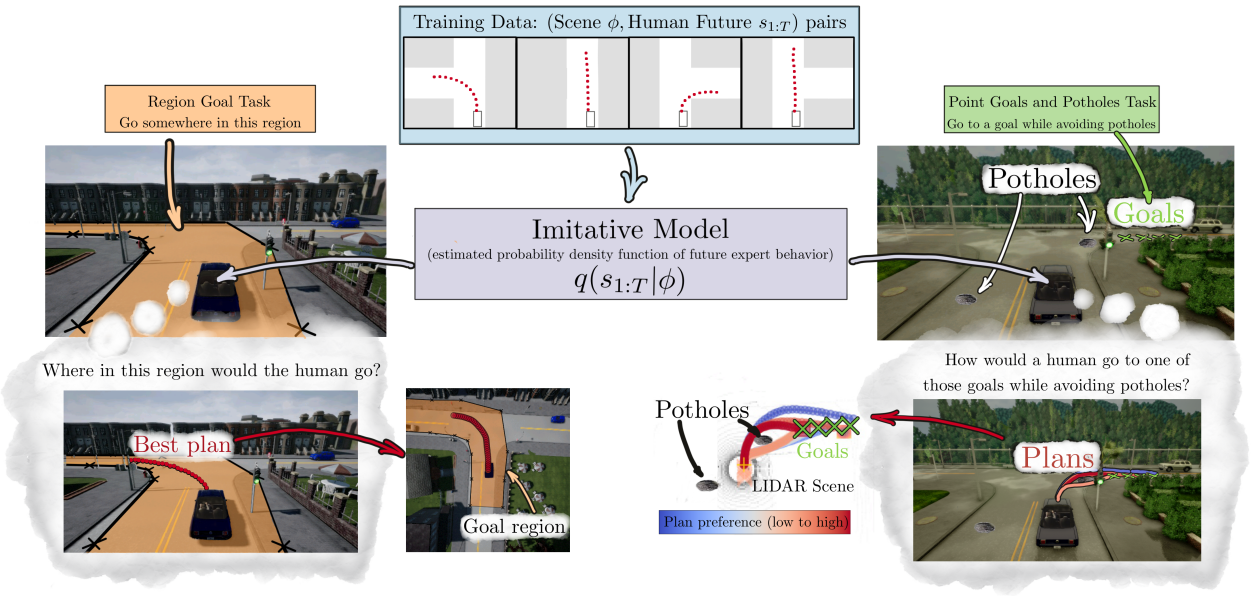


Figure 6.2: Our method: deep imitative models. *Top Center*. We use demonstrations to learn a probability density function q of future behavior and deploy it to accomplish various tasks. *Left*: A region in the ground plane is input to a planning procedure that reasons about how the expert would achieve that task. It coarsely specifies a destination, and guides the vehicle to turn left. *Right*: Goal positions and potholes yield a plan that avoids potholes and achieves one of the goals on the right.

Our goal is to devise an algorithm that combines the advantages of MBRL and IL by offering MBRL's flexibility to achieve new tasks at test-time and IL's potential to learn desirable behavior entirely from offline data. To accomplish this, we first train a model to forecast expert trajectories with a density function, which can *score trajectories and plans by how likely they are to come from the expert*. A probabilistic model is necessary because expert behavior is stochastic: e.g. at an intersection, the expert could choose to turn left or right. Next, we derive a principled probabilistic inference objective to create plans that incorporate both (1) the model and (2) arbitrary new tasks. Finally, we derive families of tasks that we can provide to the inference framework. Our method can accomplish *new tasks specified as complex goals* without having seen an expert complete these tasks before.

We investigate properties of our method on a dynamic simulated autonomous driving task (see Fig. 6.2). Videos are available at `https://sites.google.com/view/imitative-models`. Our contributions are as follows:

1. **Interpretable expert-like plans without reward engineering.** Our method outputs multi-step expert-like plans, offering superior interpretability to one-step imitation learning models. In contrast to MBRL, our method generates expert-like behaviors without reward function crafting.

2. **Flexibility to new tasks:** In contrast to IL, our method flexibly incorporates and achieves goals not seen during training, and performs complex tasks that were never demonstrated, such as navigating to goal regions and avoiding test-time only potholes, as depicted in Fig. 6.2.
3. **Robustness to goal specification noise:** We show that our method is robust to noise in the goal specification. In our application, we show that our agent can receive goals on the wrong side of the road, yet still navigate towards them while staying on the correct side of the road.
4. **State-of-the-art CARLA performance:** Our method substantially outperforms MBRL, a custom IL method, and all five prior CARLA IL methods known to us. It learned near-perfect driving through dynamic and static CARLA environments from expert observations alone.

6.2 Deep Imitative Models

We begin by formalizing assumptions and notation. We model continuous-state, discrete-time, partially-observed Markov processes. Our agent's state at time t is $\mathbf{s}_t \in \mathbb{R}^D$; $t = 0$ refers to the current time step, and ϕ is the agent's observations. Variables are bolded. Random variables are capitalized. Absent subscripts denote *all* future time steps, e.g. $\mathbf{S} \doteq \mathbf{S}_{1:T} \in \mathbb{R}^{T \times D}$. We denote a probability density function of a random variable $\mathbf{S}$ as $p(\mathbf{S})$, and its value as $p(\mathbf{s}) \doteq p(\mathbf{S}{=}\mathbf{s})$.

To learn agent dynamics that are possible and preferred, we construct a model of expert behavior. We fit an "Imitative Model" $q(\mathbf{S}_{1:T}|\phi) = \prod_{t=1}^{T} q(\mathbf{S}_t|\mathbf{S}_{1:t-1}, \phi)$ to a dataset of expert trajectories $\mathcal{D} = \{(s^i, \phi^i)\}_{i=1}^N$ drawn from a (unknown) distribution of expert behavior $s^i \sim p(\mathbf{S}|\phi^i)$. By training $q(\mathbf{S}|\phi)$ to forecast expert trajectories with high likelihood, we model the scene-conditioned expert dynamics, which can score trajectories by how likely they are to come from the expert.

6.2.1 Imitative Planning to Goals

After training, $q(\mathbf{S}|\phi)$ can generate trajectories that resemble those that the expert might generate – e.g. trajectories that navigate roads with expert-like maneuvers. However, these maneuvers will not have a specific goal. Beyond generating human-like behaviors, we wish to *direct* our agent to goals and have the agent automatically reason about the necessary mid-level details. We define general tasks by a set of goal variables $\mathcal{G}$. The probability of a plan $\mathbf{s}$ conditioned on the goal $\mathcal{G}$ is modelled by a posterior $p(\mathbf{s}|\mathcal{G}, \phi)$. This posterior is implemented with $q(\mathbf{s}|\phi)$ as a *learned* imitation prior and $p(\mathcal{G}|\mathbf{s}, \phi)$ as a *test-time* goal likelihood. We give examples of $p(\mathcal{G}|\mathbf{s}, \phi)$ after deriving a maximum a posteriori inference procedure to generate expert-like plans that achieve abstract goals:

$$\begin{aligned}\mathbf{s}^* \doteq \arg\max_{\mathbf{s}} \log p(\mathbf{s}|\mathcal{G}, \phi) &= \arg\max_{\mathbf{s}} \log q(\mathbf{s}|\phi) + \log p(\mathcal{G}|\mathbf{s}, \phi) - \log p(\mathcal{G}|\phi) \\ &= \arg\max_{\mathbf{s}} \log \underbrace{q(\mathbf{s}|\phi)}_{\text{imitation prior}} + \log \underbrace{p(\mathcal{G}|\mathbf{s}, \phi)}_{\text{goal likelihood}} . \end{aligned} \tag{6.1}$$

We perform gradient-based optimization of Eq. 6.1, and defer this discussion to Section 6.2.6. Next, we discuss several goal likelihoods, which direct the planning in different ways. They communicate *goals* they desire the agent to achieve, but not how to achieve them. The planning procedure determines how to achieve them by producing paths similar to those an expert would have taken to reach the given goal. In contrast to black-box one-step IL that predicts controls, our method produces interpretable *multi-step plans* accompanied by two scores. One estimates the plan's "expertness", the second estimates its probability to achieve the goal. Their sum communicates the plan's overall quality.

6.2.2 Constructing Goal Likelihoods

Constraint-based planning to goal sets (hyperparameter-free): Consider the setting where we have access to a set of desired final states, one of which the agent should achieve. We can model this by applying a Dirac-delta distribution on the final state, to ensure it lands in a goal set $\mathbb{G} \subset \mathbb{R}^D$:

$$p(\mathcal{G}|\mathbf{s}, \phi) \leftarrow \delta_{\mathbf{s}_T}(\mathbb{G}), \quad \delta_{\mathbf{s}_T}(\mathbb{G}) = 1 \text{ if } \mathbf{s}_T \in \mathbb{G}, \quad \delta_{\mathbf{s}_T}(\mathbb{G}) = 0 \text{ if } \mathbf{s}_T \notin \mathbb{G}. \tag{6.2}$$

$\delta_{\mathbf{s}_T}(\mathbb{G})$'s *partial support* of $\mathbf{s}_T \in \mathbb{G} \subset \mathbb{R}^D$ constrains $\mathbf{s}_T$ and introduces *no hyperparameters* into $p(\mathcal{G}|\mathbf{s}, \phi)$. For each choice of $\mathbb{G}$, we have a different way to provide high-level task information to the agent. The simplest choice for $\mathbb{G}$ is a *finite* set of points: a **(A) Final-State Indicator** likelihood. We applied (A) to a sequence of *waypoints* received from a standard A* planner (provided by the CARLA simulator), and outperformed all prior dynamic-world CARLA methods known to us. We can also consider providing an *infinite* number of points. Providing a set of line-segments as $\mathbb{G}$ yields a **(B) Line-Segment Final-State Indicator** likelihood, which encourages the final state to land along one of the segments. Finally, consider a **(C) Region Final-State Indicator** likelihood in which $\mathbb{G}$ is a polygon (see Figs. 6.2 and 6.5). Solving Eq. 6.1 with (C) amounts to *planning the most expert-like trajectory that ends inside a goal region*. We found these methods to work well when $\mathbb{G}$ contains "expert-like" final position(s), as the prior strongly penalizes plans ending in non-expert-like positions.

Unconstrained planning to goal sets (hyperparameter-based): Instead of *constraining* that the final state of the trajectory reach a goal, we can use a goal likelihood with *full support* ($\mathbf{s}_T \in \mathbb{R}^D$), centered at a desired final state. *This lets the goal likelihood encourage goals, rather than dictate them.* If there is a single desired goal ($\mathbb{G} = \{\mathbf{g}_T\}$), the **(D) Gaussian Final-State** likelihood $p(\mathcal{G}|\mathbf{s}, \phi) \leftarrow \mathcal{N}(\mathbf{g}_T; \mathbf{s}_T, \epsilon I)$ treats $\mathbf{g}_T$ as a *noisy* observation of a final future state, and encourages the plan to arrive at a final state. We can also plan to K successive states $\mathcal{G} = (\mathbf{g}_{T-K+1}, \ldots, \mathbf{g}_T)$ with a **(E) Gaussian State Sequence:** $p(\mathcal{G}|\mathbf{s}, \phi) \leftarrow \prod_{k=T-K+1}^{T} \mathcal{N}(\mathbf{g}_k; \mathbf{s}_k, \epsilon I)$ if a program wishes to specify a desired end velocity or acceleration when reaching the final state $\mathbf{g}_T$ (Fig. 6.3). Alternatively, a planner may propose a set of states with the intention that the agent should reach any one of them. This is possible by using a **(F) Gaussian Final-State Mixture:** $p(\mathcal{G}|\mathbf{s}, \phi) \leftarrow \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{g}_T^k; \mathbf{s}_T, \epsilon I)$ and is useful if some of those final states are not reachable with an expert-like plan. Unlike A–C, D–F introduce a hyperparameter "ϵ". However, they are useful when *no states in $\mathbb{G}$ correspond to observed expert behavior*, as they allow the imitation prior to be robust to poorly specified goals.

In order to tune the ϵ hyperparameter of the unconstrained likelihoods, we undertook the following binary-search procedure. When the prior frequently overwhelmed the posterior, we set $\epsilon \leftarrow 0.2\epsilon$, to yield tighter covariances, and thus *more* penalty for failing to satisfy the goals. When the posterior frequently overwhelmed the prior, we set $\epsilon \leftarrow 5\epsilon$, to yield looser covariances, and thus *less* penalty for failing to satisfy the goals. We executed this process three times: once for the "Gaussian Final-State Mixture" experiments (Section 6.4), once for the "Noise Robustness" Experiments (Section 6.4.6), and once for the pothole-planning experiments (Section 6.4.7). Note that for the Constrained-Goal Likelihoods introduced no hyperparameters to tune.

Costed planning: Our model has the additional flexibility to accept arbitrary user-specified costs c at test-time. For example, we may have updated knowledge of new hazards at test-time, such as a given map of potholes or a predicted cost map. Cost-based knowledge $c(\mathbf{s}_i|\phi)$ can be incorporated as an **(G) Energy-based** likelihood: $p(\mathcal{G}|\mathbf{s}, \phi) \propto \prod_{t=1}^{T} e^{-c(\mathbf{s}_t|\phi)}$ [116, 223]. This can be combined with other goal-seeking objectives by simply *multiplying* the likelihoods together. Examples of combining G (energy-based) with F (Gaussian mixture) were shown in Fig. 6.2 and are shown in Fig. 6.4. Next, we describe instantiating $q(\mathbf{S}|\phi)$ in CARLA [50].

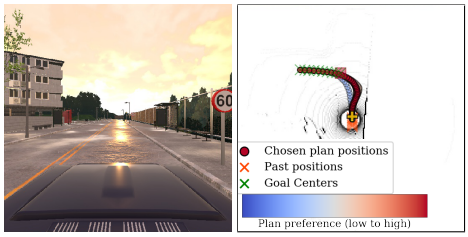


Figure 6.3: Imitative planning with the Gaussian State Sequence enables fine-grained control of the plans.

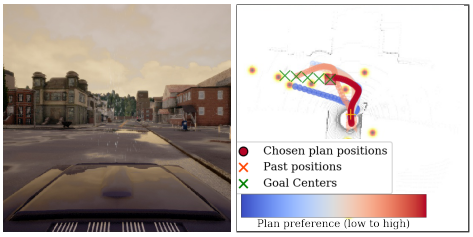


Figure 6.4: Costs can be assigned to "potholes" only seen at test-time. The planner prefers routes avoiding potholes.

Figure 6.5: Goal regions can be coarsely specified to give directions.

6.2.3 Applying Deep Imitative Models to Autonomous Driving

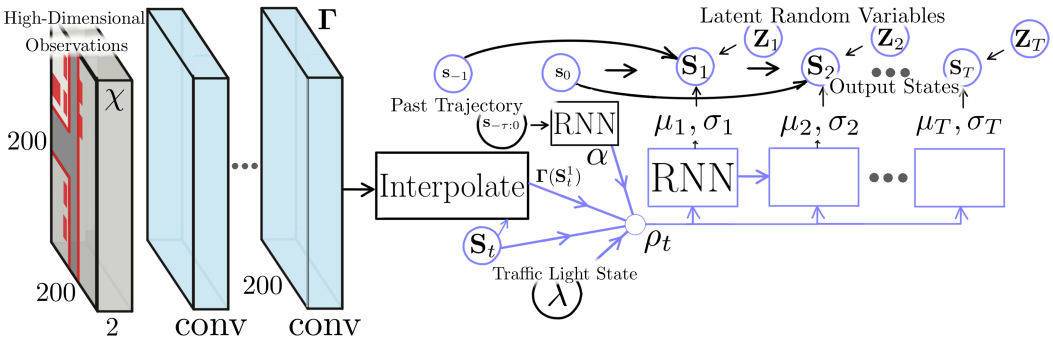


Figure 6.6: Architecture of m_θ and σ_θ, which parameterize $q_\theta(\mathbf{S}|\phi{=}\{\chi, \mathbf{s}_{-\tau:0}, \boldsymbol{\lambda}\})$. Inputs: LIDAR χ, past-states $\mathbf{s}_{-\tau:0}$, light-state $\boldsymbol{\lambda}$, and latent noise $\mathbf{Z}_{1:T}$. Output: trajectory $\mathbf{S}_{1:T}$. Details in Table 6.1.

Table 6.1: Detailed Architecture that implements $\mathbf{s}_{1:T} = f(\mathbf{z}_{1:T}, \phi)$. Typically, $T = 40$, $D = 2$,$H = W = 200$.

Component	Input [dimensionality]	Layer or Operation	Output [dimensionality]	Details
Static featurization of context: $\phi = \{\chi, \mathbf{s}_{-\tau:0}^{1:A}\}$.				
MapFeat	χ $[H, W, 2]$	2D Convolution	${}^1\chi$ $[H, W, 32]$	3×3 stride 1, ReLu
MapFeat	${}^{i-1}\chi$ $[H, W, 32]$	2D Convolution	${}^i\chi$ $[H, W, 32]$	3×3 stride 1, ReLu, $i \in [2, \ldots, 8]$
MapFeat	${}^8\chi$ $[H, W, 32]$	2D Convolution	$\boldsymbol{\Gamma}$ $[H, W, 8]$	3×3 stride 1, ReLu
PastRNN	$\mathbf{s}_{-\tau:0}$ $[\tau + 1, D]$	RNN	α $[32]$	GRU across time dimension
Dynamic generation via loop: for $t \in \{0, \ldots, T-1\}$.				
MapFeat	$\mathbf{s}_t$ $[D]$	Interpolate	$\gamma_t = \boldsymbol{\Gamma}(\mathbf{s}_t)$ $[8]$	Differentiable interpolation
JointFeat	$\gamma_t, \mathbf{s}_0, {}^2\eta, \alpha, \boldsymbol{\lambda}$	$\gamma_t \oplus \mathbf{s}_0 \oplus {}^2\eta \oplus \alpha \oplus \boldsymbol{\lambda}$	ρ_t $[D + 50 + 32 + 1]$	Concatenate ($\oplus$)
FutureRNN	ρ_t $[D + 50 + 32 + 1]$	RNN	${}^1\rho_t$ $[50]$	GRU
FutureMLP	${}^1\rho_t$ $[50]$	Affine (FC)	${}^2\rho_t$ $[200]$	Tanh activation
FutureMLP	${}^2\rho_t[200]$	Affine (FC)	m_t $[D]$, ξ_t $[D, D]$	Identity activation
MatrixExp	ξ_t $[D, D]$	$\text{expm}(\xi_t + \xi_t^{a,\text{transpose}})$	$\boldsymbol{\sigma}_t$ $[D, D]$	Differentiable Matrix Exponential [171]
VerletStep	$\mathbf{s}_t, \mathbf{s}_{t-1}, m_t, \boldsymbol{\sigma}_t, \mathbf{z}_t$	$2\mathbf{s}_t - \mathbf{s}_{t-1} + m_t + \boldsymbol{\sigma}_t \mathbf{z}_t$	$\mathbf{s}_{t+1}$ $[D]$	

In our autonomous driving application, we model the agent's state at time t as $\mathbf{s}_t \in \mathbb{R}^D$ with $D{=}2$; $\mathbf{s}_t$ represents our agent's location on the ground plane. The agent has access to environment perception $\phi \leftarrow \{\mathbf{s}_{-\tau:0}, \boldsymbol{\chi}, \boldsymbol{\lambda}\}$, where τ is the number of past positions we condition on, $\boldsymbol{\chi}$ is a high-dimensional observation of the scene, and $\boldsymbol{\lambda}$ is a low-dimensional traffic light signal. $\boldsymbol{\chi}$ could

represent either LIDAR or camera images (or both), and is the agent's observation of the world. In our setting, we featurize LIDAR to $\boldsymbol{\chi} = \mathbb{R}^{200\times200\times2}$, with $\boldsymbol{\chi}_{ij}$ representing a 2-bin histogram of points above and at ground level in a 0.5m^2 cell at position (i, j). CARLA provides ground-truth $\mathbf{s}_{-\tau:0}$ and $\boldsymbol{\lambda}$. Their availability is a realistic input assumption in perception-based autonomous driving pipelines.

Model requirements: A deep imitative model forecasts future expert behavior. It must be able to compute $q(\mathbf{s}|\phi)\forall\mathbf{s} \in \mathbb{R}^{T\times D}$. The ability to compute $\nabla_{\mathbf{s}} q(\mathbf{s}|\phi)$ enables gradient-based optimization for planning. [183] provide a recent survey on forecasting agent behavior. As many forecasting methods cannot compute trajectory probabilities, we must be judicious in choosing $q(\mathbf{S}|\phi)$. A model that can compute probabilities R2P2 [171] (presented in Chapter 4), a generative autoregressive flow [141, 166]. We extend R2P2 to instantiate the deep imitative model $q(\mathbf{S}|\phi)$. **R2P2 was previously used to forecast vehicle trajectories: it was not demonstrated or developed to plan or execute controls.** Although we used R2P2, other future-trajectory density estimation techniques could be used – designing $q(\mathbf{s}|\phi)$ is not the primary focus of this work. In R2P2, $q_\theta(\mathbf{S}|\phi)$ is induced by an invertible, differentiable function: $\mathbf{S} = f_\theta(\mathbf{Z};\phi) : \mathbb{R}^{T\times2} \mapsto \mathbb{R}^{T\times2}$; f_θ warps a latent sample from a base distribution $\mathbf{Z}\sim q_0 = \mathcal{N}(0, I)$ to $\mathbf{S}$. θ is trained to maximize $q_\theta(\mathbf{S}|\phi)$ of expert trajectories. f_θ is defined for $1..T$ as follows:

$$\mathbf{S}_t = f_t(\mathbf{Z}_{1:t}) = \mu_\theta(\mathbf{S}_{1:t-1}, \phi) + \sigma_\theta(\mathbf{S}_{1:t-1}, \phi)\mathbf{Z}_t, \tag{6.3}$$

where $\mu_\theta(\mathbf{S}_{1:t-1}, \phi) = 2\mathbf{S}_{t-1} - \mathbf{S}_{t-2} + m_\theta(\mathbf{S}_{1:t-1}, \phi)$ encodes a constant-velocity inductive bias. The $m_\theta \in \mathbb{R}^2$ and $\sigma_\theta \in \mathbb{R}^{2\times2}$ are computed by expressive neural networks. The resulting trajectory distribution is complex and multimodal (Section 6.2.4). Because traffic light state was not included in the ϕ of R2P2's "RNN" model, it could not react to traffic lights. We created a new model that includes $\boldsymbol{\lambda}$. It fixed cases where $q(\mathbf{S}|\phi)$ exhibited no forward-moving preference when the agent was already stopped, and improved $q(\mathbf{S}|\phi)$'s stopping preference at red lights. We used $T = 40$ trajectories at 10Hz (4 seconds), and $\tau = 3$. Fig. 6.6 depicts the architecture of μ_θ and σ_θ.

6.2.4 Prior Visualization and Statistics

We show examples of the priors multimodality in Fig. 6.7

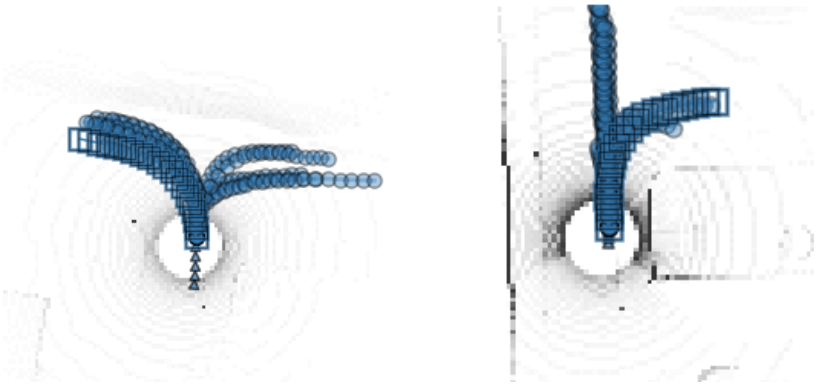


Figure 6.7: *Left*: Samples from the prior, $q(\mathbf{S}|\phi)$, go left or right. *Right:* Samples go forward or right.

6.2.4.1 Statistics of Prior and Goal Likelihoods

Following are the values of the planning criterion on $N \approx 8\cdot10^3$ rounds from applying the "Gaussian Final-State Mixture" to Town01 Dynamic. Mean of $\log q(\mathbf{s}^*|\phi) \approx 104$. Mean of $\log p(\mathcal{G}|\mathbf{s}^*, \phi) = -4$.

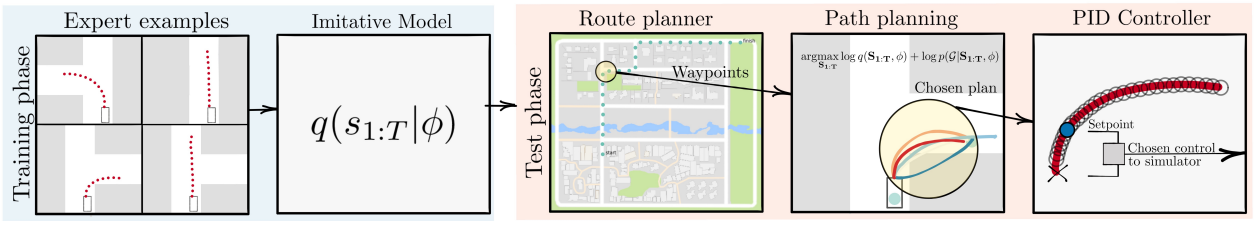


Figure 6.8: Illustration of our method applied to autonomous driving. Our method trains an imitative model from a dataset of expert examples. After training, the model is repurposed as an imitative planner. At test-time, a route planner provides waypoints to the imitative planner, which computes expert-like paths to each goal. The best plan is chosen according to the planning objective and provided to a low-level PID-controller in order to produce steering and throttle actions. This procedure is also described with pseudocode in Section 6.2.6.

This illustrates that while the prior's value mostly dominates the values of the final plans, the Gaussian Final-State Goal Mixture likelihood has a moderate amount of influence on the value of the final plan.

6.2.5 Imitative Driving

We now instantiate a complete autonomous driving framework based on imitative models to study in our experiments, seen in Fig. 6.8. We use three layers of spatial abstraction to plan to a faraway destination, common to autonomous vehicle setups: coarse route planning over a road map, path planning within the observable space, and feedback control to follow the planned path [145, 197]. For instance, a route planner based on a conventional GPS-based navigation system might output waypoints roughly in the lanes of the desired direction of travel, but not accounting for environmental factors such as the positions of other vehicles. This roughly communicates *possibilities* of where the vehicle could go, but not *when* or *how* it could get to them, or any environmental factors like other vehicles. A goal likelihood from Sec. 6.2.2 is formed from the route and passed to the planner, which generates a state-space plan according to the optimization in Eq. 6.1. The resulting plan is fed to a simple PID controller on steering, throttle, and braking. In Fig. 6.8 we illustrate how we use our model in our application.

6.2.6 Algorithms

Algorithm 4 IMITATIVEDRIVING(ROUTEPLAN, IMITATIVEPLAN, PIDCONTROLLER, q_θ, H)

$\phi \leftarrow$ ENVIRONMENT($\emptyset$) {Initialize the robot}
while not at destination **do**
  $\mathcal{G} \leftarrow$ ROUTEPLAN(ϕ) {Generate goals from a route}
  $\mathbf{s}^{\mathcal{G}}_{1:T} \leftarrow$ IMITATIVEPLANR2P2($q_\theta, \mathcal{G}, \phi$) {Plan path}
  for $h = 0$ to H **do**
    $u \leftarrow$ PIDCONTROLLER($\phi, \mathbf{s}^{\mathcal{G}}_{1:T}, h, H$)
    $\phi \leftarrow$ ENVIRONMENT(u) {Execute control}
  end for
end while

In Algorithm 4, we provide pseudocode for receding-horizon control via our imitative model. In Algorithm 5 we provide pesudocode that describes how we plan in the latent space of the trajectory. Since $\mathbf{s}_{1:T} = f(\mathbf{z}_{1:T})$ in our implementation, and f is differentiable, we can perform gradient descent of the same objective in terms of $\mathbf{z}_{1:T}$. Since q is trained with $\mathbf{z}_{1:T} \sim \mathcal{N}(0, I)$, the latent space is likelier to be better numerically conditioned than the space of $\mathbf{s}_{1:T}$, although we did not compare the two approaches formally.

Algorithm 5 IMITATIVEPLANR2P2($q_\theta, \mathcal{G}, \phi, f$)

Define MAP objective $\mathcal{L}$ with q_θ according to Eq. 6.1 {Incorporate the Imitative Model}
Initialize $\mathbf{z}_{1:T} \sim q_0$
while not converged **do**
  $\mathbf{z}_{1:T} \leftarrow \mathbf{z}_{1:T} + \nabla_{\mathbf{z}_{1:T}} \mathcal{L}(\mathbf{s}_{1:T} = f(\mathbf{z}_{1:T}), \mathcal{G}, \phi)$
end while
return $\mathbf{s}_{1:T} = f(\mathbf{z}_{1:T})$

In Algorithm 6, we detail the speed-based throttle and position-based steering PID controllers.

Algorithm 6 PIDCONTROLLER($\phi = \{\mathbf{s}_0, \mathbf{s}_{-1}, \dots\}, \mathbf{s}^{\mathcal{G}}_{1:T}, h, H; K^{\dot{s}}_p, K^{\alpha}_p$)

$i \leftarrow T - H + h$ {Compute the index of the target position}
$\dot{s}_{\text{process-speed}} \leftarrow (\mathbf{s}_{0,x} - \mathbf{s}_{-1,x})$ {Compute the current forward speed from the observations}
$s_{\text{setpoint-position}} \leftarrow \mathbf{s}^{\mathcal{G}}_{i,x}$ {Retrieve the target position x-coordinate from the plan}
$\dot{s}_{\text{setpoint-speed}} \leftarrow {}^{s_{\text{setpoint-position}}}/_{i}$ {Compute the forward target speed}
$e_{\dot{s}} \leftarrow \dot{s}_{\text{setpoint-speed}} - \dot{s}_{\text{process-speed}}$ {Compute the forward speed error}
$u_{\dot{s}} \leftarrow K^{\dot{s}}_p e_{\dot{s}}$ {Compute the accelerator control with a nonzero proportional term}
throttle $\leftarrow \mathbb{1}(e > 0) \cdot u + \mathbb{1}(e \leq 0) \cdot 0$ {Use the control as throttle if the speed error is positive}
brake $\leftarrow \mathbb{1}(e > 0) \cdot 0 + \mathbb{1}(e \leq 0) \cdot u$ {Use the control as brake if the speed error is negative}
$\alpha_{\text{process}} \leftarrow \arctan(\mathbf{s}_{0,y} - \mathbf{s}_{-1,y}, \mathbf{s}_{0,x} - \mathbf{s}_{-1,x})$ {Compute current heading}
$\alpha_{\text{setpoint}} \leftarrow \arctan(\mathbf{s}^{\mathcal{G}}_{i,y} - \mathbf{s}_{0,y}, |\mathbf{s}^{\mathcal{G}}_{i,x} - \mathbf{s}_{0,x}|)$ {Compute target forward heading}
$e_\alpha \leftarrow \alpha_{\text{setpoint}} - \alpha_{\text{process}}$ {Compute the heading error}
steering $\leftarrow K^{\alpha}_p e_\alpha$ {Compute the steering with a nonzero proportional term}
$u \leftarrow \big[\text{throttle}, \text{steering}, \text{brake}\big]$
return u

6.2.7 Optimizing Goal Likelihoods with Set Constraints

We now derive an approach to optimize our main objective with set constraints. Although we could apply a constrained optimizer, we find that we are able to exploit properties of the model and constraints to derive differentiable objectives that enable approximate optimization of the corresponding closed-form optimization problems. These enable us to use the same straightforward gradient-descent-based optimization approach described in Algorithm 5.

Shorthand notation: In this section we omit dependencies on ϕ for brevity, and use short hand $\mu_t \doteq \mu_\theta(\mathbf{s}_{1:t-1})$ and $\Sigma_t \doteq \Sigma_\theta(\mathbf{s}_{1:t-1})$. For example, $q(\mathbf{s}_t|\mathbf{s}_{1:t-1}) = \mathcal{N}(\mathbf{s}_t; \mu_t, \Sigma_t)$.

Let us begin by defining a useful delta function:

$$\delta_{\mathbf{s}_T}(\mathbb{G}) \doteq \begin{cases} 1 & \text{if } \mathbf{s}_T \in \mathbb{G} \\ 0 & \text{if } \mathbf{s}_T \notin \mathbb{G}, \end{cases} \tag{6.4}$$

which serves as our goal likelihood when using goal with set constraints: $p(\mathcal{G}|\mathbf{s}_{1:T}) \leftarrow \delta_{S_T}(\mathbb{G})$. We now derive the corresponding maximum a posteriori optimization problem:

$$\begin{aligned}
\mathbf{s}^*_{1:T} &\doteq \underset{\mathbf{s}_{1:T}\in\mathbb{R}^{2T}}{\arg\max}\ p(\mathbf{s}_{1:T}|\mathcal{G}) \\
&= \underset{\mathbf{s}_{1:T}\in\mathbb{R}^{2T}}{\arg\max}\ p(\mathcal{G}|\mathbf{s}_{1:T}) \cdot q(\mathbf{s}_{1:T}) \cdot p^{-1}(\mathcal{G}) \\
&= \underset{\mathbf{s}_{1:T}\in\mathbb{R}^{2T}}{\arg\max}\ \underbrace{p(\mathcal{G}|\mathbf{s}_{1:T})}_{\text{goal likelihood}} \cdot \underbrace{q(\mathbf{s}_{1:T})}_{\text{imitation prior}} \\
&= \underset{\mathbf{s}_{1:T}\in\mathbb{R}^{2T}}{\arg\max}\ \underbrace{\delta_{S_T}(\mathbb{G})}_{\text{set constraint}} \cdot \underbrace{q(\mathbf{s}_{1:T})}_{\text{imitation prior}} \\
&= \underset{\mathbf{s}_{1:T}\in\mathbb{R}^{2T}}{\arg\max} \begin{cases} q(\mathbf{s}_{1:T}) & \text{if } \mathbf{s}_T \in \mathbb{G} \\ 0 & \text{if } \mathbf{s}_T \notin \mathbb{G} \end{cases} \\
&= \underset{\mathbf{s}_{1:T-1}\in\mathbb{R}^{2(T-1)}, \mathbf{s}_T\in\mathbb{G}}{\arg\max}\ q(\mathbf{s}_{1:T}) \\
&= \underset{\mathbf{s}_{1:T-1}\in\mathbb{R}^{2(T-1)}}{\arg\max}\ \underset{\mathbf{s}_T\in\mathcal{G}}{\arg\max}\ q(\mathbf{s}_T|\mathbf{s}_{1:T-1}) \prod_{t=1}^{T-1} q(\mathbf{s}_t|\mathbf{s}_{1:t-1}) \\
&= \underset{\mathbf{s}_{1:T-1}\in\mathbb{R}^{2(T-1)}}{\arg\max}\ \underset{\mathbf{s}_T\in\mathcal{G}}{\arg\max}\ \mathcal{N}(\mathbf{s}_T; \mu_T, \Sigma_T) \prod_{t=1}^{T-1} \mathcal{N}(\mathbf{s}_t; \mu_t, \Sigma_t). \qquad (6.5)
\end{aligned}$$

By exploiting the fact that $q(\mathbf{s}_T|\mathbf{s}_{1:T-1}) = \mathcal{N}(\mathbf{s}_T; \mu_T, \Sigma_T)$, we can derive closed-form solutions for

$$\mathbf{s}^*_T = \underset{\mathbf{s}_T\in\mathbb{G}}{\arg\max}\ \mathcal{N}(\mathbf{s}_T; \mu_T, \Sigma_T) \qquad (6.6)$$

when $\mathbb{G}$ has special structure, which *enables us to apply gradient descent to solve this constrained-optimization problem* (examples below). With a closed form solution to equation 6.6, we can easily compute equation 6.5 using *unconstrained*-optimization as follows:

$$\mathbf{s}^*_{1:T} = \underset{\mathbf{s}_{1:T-1}\in\mathbb{R}^{2(T-1)}}{\arg\max}\ \underset{\mathbf{s}_T\in\mathbb{G}_{\text{line-segment}}}{\arg\max}\ q(\mathbf{s}_T|\mathbf{s}_{1:T-1}) \prod_{t=1}^{T-1} q(\mathbf{s}_t|\mathbf{s}_{1:t-1}) \qquad (6.7)$$

$$\mathbf{s}^*_{1:T-1} = \underbrace{\underset{\mathbf{s}_{1:T-1}\in\mathbb{R}^{2(T-1)}}{\arg\max}}_{\text{unconstrained optimization}}\ \underbrace{q(\mathbf{s}^*_T|\mathbf{s}_{1:t-1}) \prod_{t=1}^{T-1} q(\mathbf{s}_t|\mathbf{s}_{1:t-1})}_{\text{objective function of } \mathbf{s}_{1:T-1}}. \qquad (6.8)$$

Note that equation 6.8 only helps solve equation 6.5 if equation 6.6 has a closed-form solution. We detail example of goal sets with such closed-form solutions in the following subsections.

6.2.7.1 Point Goal Set

The solution to equation 6.6 in the case of a single desired goal $g \in \mathbb{R}^D$ is simply:

$$\mathbb{G}_{\text{point}} \doteq \{\mathbf{g}_T\}, \qquad (6.9)$$

$$\begin{aligned}
\mathbf{s}^*_{T,\text{point}} &\doteq \underset{\mathbf{s}_T\in\mathbb{G}_{\text{point}}}{\arg\max}\ \mathcal{N}(\mathbf{s}_T; \mu_T, \Sigma_T) \\
&= \mathbf{g}_T. \qquad (6.10)
\end{aligned}$$

More generally, multiple point goals help define *optional* end points for planning: where the agent only need reach one valid end point (see Fig. 6.9 for examples), formulated as:

$$\mathbb{G}_{\text{points}} \doteq \{\mathbf{g}_T^k\}_{k=1}^K, \tag{6.11}$$

$$\mathbf{s}_{T,\text{points}}^* \doteq \underset{\mathbf{g}_T^k \in \mathbb{G}_{\text{points}}}{\arg\max}\, \mathcal{N}\left(\mathbf{g}_T^k; \mu_T, \Sigma_T\right). \tag{6.12}$$

6.2.7.2 Line Segment Goal Set

We can form a goal set as a finite-length line segment, connecting point $\mathbf{a} \in \mathbb{R}^D$ to point $\mathbf{b} \in \mathbb{R}^D$:

$$g_{\text{line}}(u) \doteq \mathbf{a} + u \cdot (\mathbf{b} - \mathbf{a}),\ u \in \mathbb{R}, \tag{6.13}$$

$$\mathbb{G}_{\text{line-segment}}^{\mathbf{a}\to\mathbf{b}} \doteq \{g_{\text{line}}(u) : u \in [0,1]\}. \tag{6.14}$$

The solution to equation 6.6 in the case of line-segment goals is:

$$\mathbf{s}_{T,\text{line-segment}}^* \doteq \underset{\mathbf{s}_T \in \mathbb{G}_{\text{line-segment}}^{\mathbf{a}\to\mathbf{b}}}{\arg\max}\ \mathcal{N}\left(\mathbf{s}_T; \mu_T, \Sigma_T\right) \tag{6.15}$$

$$= \mathbf{a} + \min\left(1,\ \max\left(0,\ \frac{(\mathbf{b}-\mathbf{a})^\top \Sigma_T^{-1}(\mu_T - \mathbf{a})}{(\mathbf{b}-\mathbf{a})^\top \Sigma_T^{-1}(\mathbf{b}-\mathbf{a})}\right)\right) \cdot (\mathbf{b} - \mathbf{a}). \tag{6.16}$$

Proof:

To solve equation 6.15 is to find which point along the line $g_{\text{line}}(u)$ maximizes $\mathcal{N}\left(\cdot; \mu_T, \Sigma_T\right)$ subject to the constraint $0 \leq u \leq 1$:

$$u^* \doteq \underset{u\in[0,1]}{\arg\max}\ \mathcal{N}\left(g_{\text{line}}(u); \mu_T, \Sigma_T\right))$$

$$= \underset{u\in[0,1]}{\arg\min}\ \underbrace{(g_{\text{line}}(u) - \mu_T)^\top \Sigma_T^{-1}(g_{\text{line}}(u) - \mu_T)}_{\mathcal{L}_u(u)}. \tag{6.17}$$

Since $\mathcal{L}_u$ is convex, the optimal value u^* is value closest to the unconstrained $\arg\max$ of $\mathcal{L}_u(u)$, subject to $0 \leq u \leq 1$:

$$u_{\mathbb{R}}^* \doteq \underset{u\in\mathbb{R}}{\arg\max}\ \mathcal{L}_u(u), \tag{6.18}$$

$$u^* = \underset{u\in[0,1]}{\arg\min}\ \mathcal{L}_u(u)$$

$$= \min\left(1,\ \max\left(0,\ u_{\mathbb{R}}^*\right)\right). \tag{6.19}$$

We now solve for $u_{\mathbb{R}}^*$:

$$u_{\mathbb{R}}^* = u : 0 = \frac{\mathrm{d}\mathcal{L}(u)}{\mathrm{d}u} = \frac{\mathrm{d}\left((g_{\text{line}}(u) - \mu_T)^\top \Sigma_T^{-1}(g_{\text{line}}(u) - \mu_T)\right)}{\mathrm{d}u}$$

$$= 2 \cdot \frac{\mathrm{d}(g_{\text{line}}(u) - \mu_T)^\top}{\mathrm{d}u} \Sigma_T^{-1}(g_{\text{line}}(u) - \mu_T)$$

$$= 2 \cdot \frac{\mathrm{d}(\mathbf{a} + u \cdot (\mathbf{b}-\mathbf{a}) - \mu_T)^\top}{\mathrm{d}u} \Sigma_T^{-1}(\mathbf{a} + u \cdot (\mathbf{b}-\mathbf{a}) - \mu_T)$$

$$= 2 \cdot (\mathbf{b}-\mathbf{a})^\top \Sigma_T^{-1}(\mathbf{a} + u \cdot (\mathbf{b}-\mathbf{a}) - \mu_T),$$

$$u_{\mathbb{R}}^* = \frac{(\mathbf{b}-\mathbf{a})^\top \Sigma_T^{-1}(\mu_T - \mathbf{a})}{(\mathbf{b}-\mathbf{a})^\top \Sigma_T^{-1}(\mathbf{b}-\mathbf{a})}, \tag{6.20}$$

which gives us:

$$\begin{aligned}
\mathbf{s}^*_{T,\text{line-segment}} &= g_{\text{line}}(u^*) \\
&= \mathbf{a} + u^* \cdot (\mathbf{b} - \mathbf{a}) \\
&= \mathbf{a} + \min\left(1,\ \max\left(0,\ u^*_{\mathbb{R}}\right)\right) \cdot (\mathbf{b} - \mathbf{a}) \\
&= \mathbf{a} + \min\left(1,\ \max\left(0,\ \frac{(\mathbf{b} - \mathbf{a})^\top \Sigma_T^{-1}(\mu_T - \mathbf{a})}{(\mathbf{b} - \mathbf{a})^\top \Sigma_T^{-1}(\mathbf{b} - \mathbf{a})}\right)\right) \cdot (\mathbf{b} - \mathbf{a}). \qquad (6.21)
\end{aligned}$$

6.2.7.3 Multiple Line Segment Goal Set:

More generally, we can combine multiple line-segments to form piecewise linear "paths" we wish to follow. By defining a path that connects points $(\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_N)$, we can evaluate $\mathcal{L}^i_u$ for each $\mathbb{G}^{\mathbf{p}_i \to \mathbf{p}_{i+1}}_{\text{line-segment}}$, select the optimal segment $i^* = \arg\max_i \mathcal{L}^i_u$, and use the segment i^*'s solution to u^* to compute s^*_T. Examples shown in Fig. 6.10.

6.2.7.4 Polygon Goal Set

Instead of a route or path, a user (or program) may wish to provide a general *region* the agent should go to, and state within that region being equally valid. Polygon regions (including both boundary and interior) offer closed form solution to equation 6.6 and are simple to specify. A polygon can be specified by an ordered sequence of vertices $(\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_N) \in \mathbb{R}^{N \times 2}$. Edges are then defined as the sequence of line-segments between successive vertices (and a final edge between first and last vertex): $\big((\mathbf{p}_0, \mathbf{p}_1), ..., (\mathbf{p}_{N-1}, \mathbf{p}_N), (\mathbf{p}_N, \mathbf{p}_0)\big)$. Examples shown in Fig. 6.11 and 6.12.

Solving equation 6.6 with a polygon has two cases: depending whether μ_T is *inside* the polygon, or *outside*. If μ_T lies inside the polygon, then the optimal value for $\mathbf{s}^*_T$ that maximizes $\mathcal{N}(\mathbf{s}^*_T; \mu_T, \Sigma_T)$ is simply μ_T: the mode of the Gaussian distribution. Otherwise, if μ_T lies outside the polygon, then the optimal value $\mathbf{s}^*_T$ will lie on one of the polygon's edges, solved using 6.2.7.3. Having introduced and discussed the goal likelihoods, we next turn to describing related work.

6.3 Related Work

A body of previous work has explored offline IL (Behavior Cloning – BC) in the CARLA simulator [41, 42, 122, 123, 191]. These BC approaches condition on goals drawn from a small discrete set of directives. Despite BC's theoretical drift shortcomings [181], these methods still perform empirically well. *These approaches and ours share the same high-level routing algorithm: an A* planner on route nodes that generates waypoints*. In contrast to our approach, these approaches use the waypoints in a *Waypoint Classifier*, which reasons about the map and the geometry of the route to classify the waypoints into one of several directives: {Turn left, Turn right, Follow Lane, Go Straight}. One of the original motivations for these type of controls was to enable *a human* to direct the robot [41]. **However, in scenarios where there is no human in the loop (i.e. autonomous driving), we advocate for approaches to make use of the detailed spatial information inherent in these waypoints**. Our approach and several others we designed make use of this spatial information. One of these is CIL-States (CILS): whereas the approach in [41] uses images to directly generate controls, *CILS uses identical inputs and PID controllers as our method*. With respect to prior conditional IL methods, our main approach has more flexibility to handle more complex directives post-training, the ability to learn without goal labels, and the ability to generate interpretable planned and unplanned trajectories. These contrasting capabilities are illustrated in Table 6.2.

Table 6.2: Desirable attributes of each approach. A green check denotes that a method has a desirable attribute, whereas a red cross denotes the opposite. A "†" indicates an approach we implemented.

Approach	Flexible to New Goals	Trains without goal labels	Outputs Plans	Trains Offline	Has Expert P.D.F.
CIRL* [123]	✗	✗	✗	✗	✗
CAL* [191]	✗	✗	✗	✓	✗
MT* [122]	✗	✗	✗	✓	✗
CIL* [41]	✗	✗	✗	✓	✗
CILRS* [42]	✗	✗	✗	✓	✗
CILS†	✗	✓	✗	✓	✗
MBRL†	✓	✓	✓	✗	✗
Imitative Models (*Ours*)†	✓	✓	✓	✓	✓

Table 6.3: Algorithmic components of each approach. A "†" indicates an approach we implemented.

Approach	Control Algorithm	← Learning Algorithm	← Goal-Generation Algorithm	← Routing Algorithm	High-Dim. Obs.
CIRL* [123]	Policy	Behavior Cloning+RL	Waypoint Classifier	A* Waypointer	Image
CAL* [191]	PID	Affordance Learning	Waypoint Classifier	A* Waypointer	Image
MT* [122]	Policy	Behavior Cloning	Waypoint Classifier	A* Waypointer	Image
CIL* [41]	Policy	Behavior Cloning	Waypoint Classifier	A* Waypointer	Image
CILRS* [42]	Policy	Behavior Cloning	Waypoint Classifier	A* Waypointer	Image
CILS†	PID	Trajectory Regressor	Waypoint Classifier	A* Waypointer	LIDAR
MBRL†	Reachability Tree	State Regressor	Waypoint Selector	A* Waypointer	LIDAR
Imitative Models (*Ours*)†	Imitative Plan+PID	Traj. Density Est.	Goal Likelihoods	A* Waypointer	LIDAR

Our approach is also related to MBRL. MBRL can also plan, but with a one-step predictive model of *possible* dynamics. The task of evoking expert-like behavior is offloaded to the reward function, which can be difficult and time-consuming to craft properly. We know of no MBRL approach previously applied to CARLA, so we devised one for comparison. *This MBRL approach also uses identical inputs to our method*, instead to plan a reachability tree [109] over an dynamic obstacle-based reward function.

Several prior works [11, 208, 220] used imitation learning to train policies that contain planning-like modules as part of the model architecture. While our work also combines planning and imitation learning, ours captures a distribution over possible trajectories, and then plan trajectories at test-time that accomplish a variety of given goals with high probability under this distribution. Our approach is suited to offline-learning settings where interactively collecting data is costly (time-consuming or dangerous). However, there exists online IL approaches that seek to be *safe* [132, 211, 249].

6.4 Experiments

We evaluate our method using the CARLA driving simulator [50]. We seek to answer four primary questions: **(1) Can we generate interpretable, expert-like plans with offline learning and no reward engineering**? Neither IL nor MBRL can do so. It is straightforward to *interpret* the trajectories by visualizing them on the ground plane; we thus seek to validate whether these plans are *expert-like* by equating expert-like behavior with high performance on the CARLA benchmark. **(2) Can we achieve state-of-the-art CARLA performance using resources commonly available in real autonomous vehicle settings?** *There are several differences between the approaches, as discussed in Sec 6.3 and shown in Tables 6.2 and 6.3. Our approach uses the CARLA toolkit's resources that are commonly available in real autonomous vehicle settings: waypoint-based routes (all prior approaches use these) and LIDAR (CARLA-provided, but only the approaches we implemented use it). Furthermore, the two additional methods of comparison we implemented (CILS and MBRL) use the exact same inputs as our algorithm. These reasons justify an overall performance comparison to answer (2): whether we can achieve state-of-the-art performance using commonly available resources. We advocate that other approaches also make use of such*

resources. **(3) How flexible is our approach to new tasks?** We investigate (3) by applying each of the goal likelihoods we derived and observing the resulting performance. **(4) How robust is our approach to error in the provided goals?** We do so by injecting two different types of error into the waypoints and observing the resulting performance.

6.4.1 Instantiating Goal Likelihoods for Autonomous Driving

The waypointer uses the CARLA planner's provided route to generate waypoints. In the constrained-based planning goal likelihoods, we use this route to generate waypoints without interpolating between them. In the relaxed goal likelihoods, we interpolate this route to every 2 meters, and use the first 20 waypoints.

Given the in-lane waypoints generated by CARLA's route planner, we use these to create Point goal sets, Line-Segment goal sets, and Polygon goal sets, which respectively correspond to the (A) Final-State Indicator, (B) Line-Segment Final-State Indicator, and (C) Final-State Region Indicator described in Section 6.2.2. For (A), we simply feed the waypoints directly into the Final-State Indicator, which results in a constrained optimization to ensure that $S_T \in \mathbb{G} = \{g_T^k\}_{k=1}^K$. We also included the vehicle's current position in the goal set, in order to allow it to stop. The gradient-descent based optimization is then formed from combining Eq. 6.8 with Eq. 6.12. The gradient to the nearest goal of the final state of the partially-optimized plan encourage the optimization to move the plan closer to that goal. We used $K = 10$. We applied the same procedure to generate the goal set for the (B) Line Segment indicator, as the waypoints returned by the planner are ordered. Finally, for the (C) Final-State Region Indicator (polygon), we used the ordered waypoints as the "skeleton" of a polygon that surrounds. It was created by adding a two vertices for each point $\mathbf{v}_t$ in the skeleton at a distance 1 meter from $\mathbf{v}_t$ perpendicular to the segment connecting the surrounding points $(\mathbf{v}_{t-1}, \mathbf{v}_{t+1})$. This resulted in a goal set $\mathbb{G}_{\text{polygon}} \supset \mathbb{G}_{\text{line-segment}}$, as it surrounds the line segments. The (F) Gaussian Final-State Mixture goal set was constructed in the same way as (A), and also used when the pothole costs were added.

For the methods we implemented, the task is to drive the *furthest* road location from the vehicle's initial position. Note that this protocol more difficult than the one used in prior work [41, 42, 122, 123, 191], which has no distance guarantees between start positions and goals, and often results in shorter paths.

6.4.2 Planning Visualizations

Visualizations of examples of our method deployed with different goal likelihoods are shown in Fig. 6.9, Fig. 6.10, Fig. 6.11, and Fig. 6.12.

6.4.3 Dataset and Metrics

Before training $q(\mathbf{S}|\phi)$, we ran CARLA's expert in the dynamic world setting of Town01 to collect a dataset of examples. We ran the autopilot in Town01 for over 900 episodes of 100 seconds each in the presence of 100 other vehicles, and recorded the trajectory of every vehicle and the autopilot's LIDAR observation. We randomized episodes to either train, validation, or test sets. We created sets of 60,701 train, 7586 validation, and 7567 test scenes, each with 2 seconds of past and 4 seconds of future position information at 10Hz. The dataset also includes 100 episodes obtained by following the same procedure in Town02. We then train $q(\mathbf{S}|\phi)$ on this dataset. Following existing protocol, each test episode begins with the vehicle randomly positioned on a road in the `Town01` or `Town02` maps in one of two settings: static-world (no other vehicles) or dynamic-world (with other vehicles). We construct the goal set $\mathbb{G}$ for the Final-State Indicator (A) directly from the route output

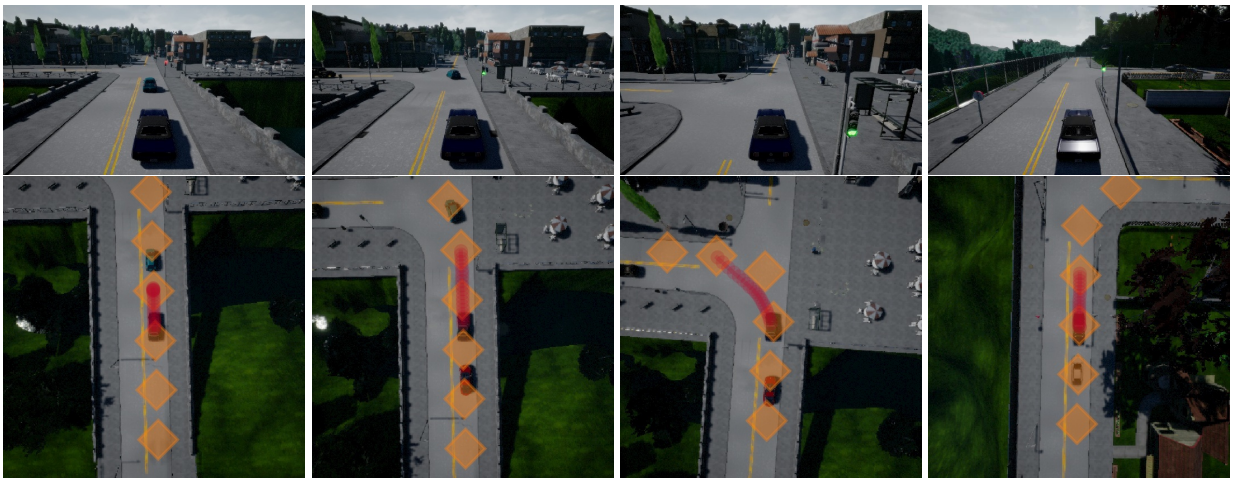


Figure 6.9: Planning with the Final State Indicator yields plans that end at one of the provided locations. The orange diamonds indicate the locations in the goal set. The red circles indicate the chosen plan.

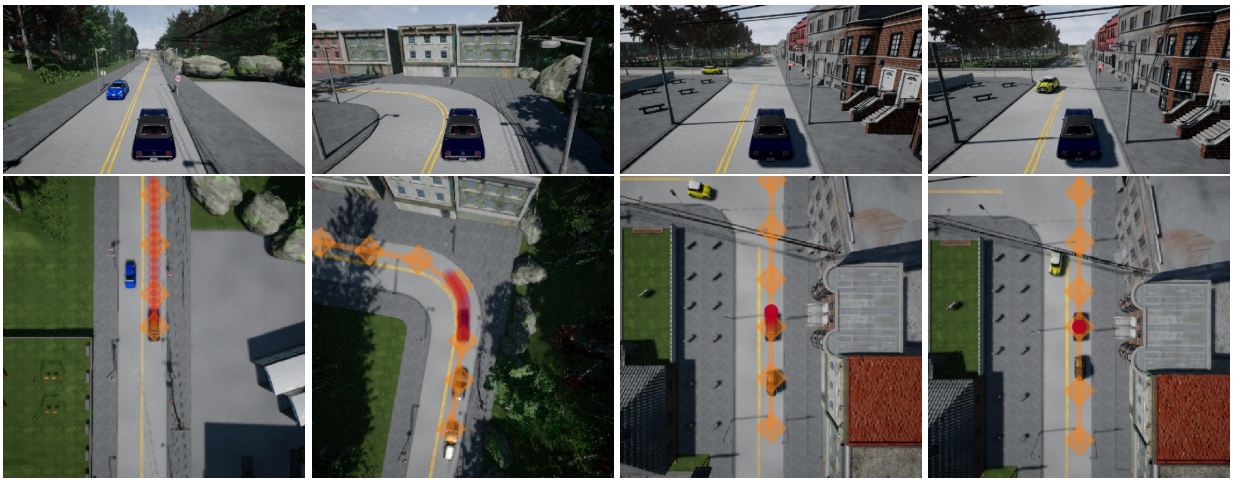


Figure 6.10: Planning with the Line Segment Final State Indicator yields plans that end along one of the segments. The orange diamonds indicate the endpoints of each line segment. The red circles indicate the chosen plan.

Figure 6.11: Planning with the Region Final State Indicator yields plans that end inside the region. The orange polygon indicates the region. The red circles indicate the chosen plan.

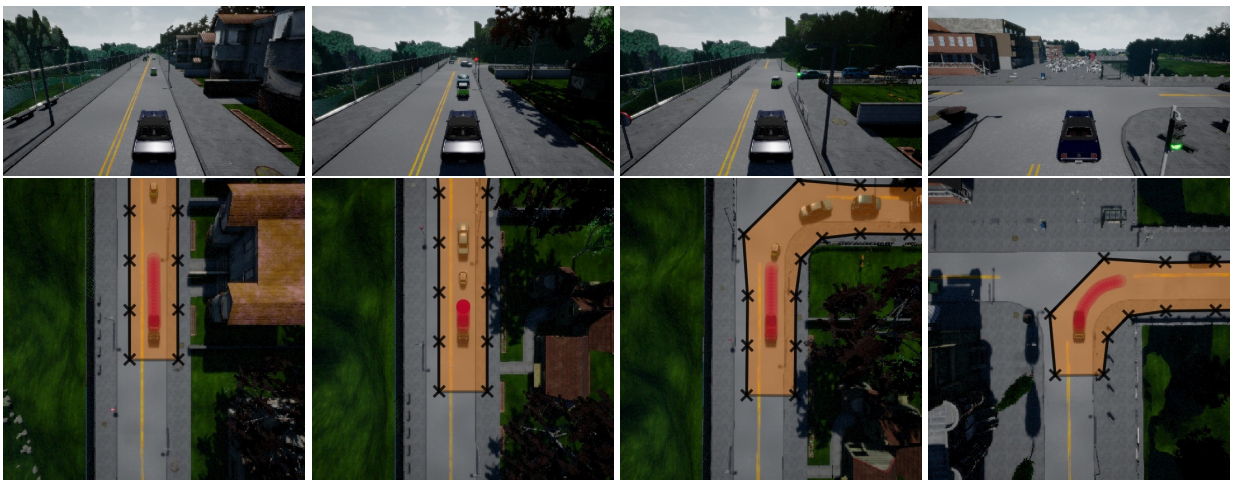


Figure 6.12: Planning with the Region Final State Indicator yields plans that end inside the region. The orange polygon indicates the region. The red circles indicate the chosen plan. Note even with a wider goal region than Fig. 6.11, the vehicle remains in its lane, due to the imitation prior. Despite their coarseness, these wide goal regions still provide useful guidance to the vehicle.

by CARLA's waypointer. B's line segments are formed by connecting the waypoints to form a piecewise linear set of segments. C's regions are created a *polygonal goal region* around the segments of (B). Each represents an increasing level of coarseness of direction. Coarser directions are easier to specify when there is ambiguity in positions (both the position of the vehicle and the position of the goals). We use three metrics: (a) success rate in driving to the destination without any collisions (which all prior work reports); (b) red-light violations; and (c) proportion of time spent driving in the wrong lane and off road. With the exception of metric (a), lower numbers are better.

6.4.4 Baseline Details

Conditional Imitation Learning of States (CILS): We designed a conditional imitation learning baselines that predicts the setpoint for the PID-controller. Each receives the same scene observations (LIDAR) and is trained with the same set of trajectories as our main method. It uses nearly the same architecture as that of the original CIL, except it outputs setpoints instead of controls, and also observes the traffic light information. We found it very effective for stable control on straightaways. When the model encounters corners, however, prediction is more difficult, as in order to successfully avoid the curbs, the model must implicitly plan a safe path. We found that using the traffic light information allowed it to stop more frequently. **Model-Based Reinforcement Learning:** To compare against a purely model-based reinforcement learning algorithm, we propose a model-based reinforcement learning baseline. This baseline first learns a forwards dynamics model $\mathbf{s}_{t+1} = f(\mathbf{s}_{t-3:t}, \mathbf{a}_t)$ given observed expert data (a_t are recorded vehicle actions). We use an MLP with two hidden layers, each 100 units. Note that our forwards dynamics model does not imitate the expert preferred actions, but only models what is physically possible. Together with the same LIDAR map $\boldsymbol{\chi}$ our method uses to locate obstacles, this baseline uses its dynamics model to plan a reachability tree [109] through the free-space to the waypoint while avoiding obstacles. The planner opts for the lowest-cost path that ends near the goal $C(\mathbf{s}_{1:T}; \mathbf{g}_T) = ||\mathbf{s}_T - \mathbf{g}_T||_2 + \sum_{t=1}^{T} c(\mathbf{s}_t)$, where cost of a position is determined by $c(\mathbf{s}_t) = 1.5\mathbb{1}(\mathbf{s}_t < 1$ meters from any obstacle$) + 0.75\mathbb{1}(1 <= \mathbf{s}_t < 2$ meters from any obstacle$) + \dddot{\mathbf{s}}_t$.

We plan forwards over 20 time steps using a breadth-first search search over CARLA steering angle $\{-0.3, -0.1, 0., 0.1, 0.3\}$, noting valid steering angles are normalized to $[-1, 1]$, with constant throttle at 0.5, noting the valid throttle range is $[0, 1]$. Our search expands each state node by the available actions and retains the 50 closest nodes to the waypoint. The planned trajectory efficiently reaches the waypoint, and can successfully plan around perceived obstacles to avoid getting stuck. To convert the LIDAR images into obstacle maps, we expanded all obstacles by the approximate radius of the car, 1.5 meters.

For Dynamic-MBRL, we use the same setup as the Static-MBRL method, except we add a discrete temporal dimension to the search space (one $\mathbb{R}^2$ spatial dimension per T time steps into the future). All static obstacles remain static, however all LIDAR points that were known to collide with a vehicle are now removed: and replaced at every time step using a constant velocity model of that vehicle. We found that the main failure mode was due to both to inaccuracy in constant velocity prediction as well as the model's inability to perceive lanes in the LIDAR. The vehicle would sometimes wander into the opposing traffic's lane, having failed to anticipate an oncoming vehicle blocking its path.

6.4.5 CARLA Benchmark Results

: Towards questions (1) and (3) (expert-like plans and flexibility), we apply our approach with a variety of goal likelihoods to the CARLA simulator. Towards question (2), we compare our methods against CILS, MBRL, and prior work. These results are shown in Table 6.4. The metrics for the

methods we did not implement are from the aggregation reported in [42]. We observe our method to outperform all other approaches in all settings: static world, dynamic world, training conditions, and test conditions. We observe the *Goal Indicator methods are able to perform well, despite having no hyperparameters to tune*. We found that we could further improve our approach's performance if we use the light state to define different goal sets, which defines a "smart" waypointer. This waypointer simply removes nearby waypoints closer than 5 meters from the vehicle when a green light is observed in the measurements provided by CARLA, to encourage the agent to move forward, and removes far waypoints beyond 5 meters from the vehicle when a red light is observed in the measurements provided by CARLA. The settings where we use this are suffixed with "S." in the Tables. We observed the planner prefers *closer* goals when obstructed, when the vehicle was already stopped, and when a red light was detected; we observed the planner prefers *farther* goals when unobstructed and when green lights or no lights were observed. Examples of these and other interesting behaviors are best seen in the videos on the website (https://sites.google.com/view/imitative-models). These behaviors follow from the method leveraging $q(\mathbf{S}|\phi)$'s internalization of aspects of expert behavior in order to reproduce them in new situations. Altogether, these results provide affirmative answers to questions (1) and (2). Towards question (3), these results show that our approach is flexible to different directions defined by these goal likelihoods.

Table 6.4: We evaluate different autonomous driving methods on CARLA's *Dynamic Navigation* task. A "†" indicates methods we have implemented (each observes the same waypoints and LIDAR as input). A "*" indicates results reported in [42]. A "–" indicates an unreported statistic. A "‡" indicates an optimistic estimate in transferring a result from the static setting to the dynamic setting. "S." denotes a "smart" waypointer reactive to light state.

		Town01 (training conditions)				Town02 (test conditions)			
Dynamic Nav. Method		Success	Ran Red Light	Wrong lane	Off road	Success	Ran Red Light	Wrong lane	Off road
CIRL*	[123]	82%	–	–	–	41%	–	–	–
CAL*	[191]	83%	–	–	–	64%	–	–	–
MT*	[122]	81%	–	–	–	53%	–	–	–
CIL*	[41]	83%	83%‡	–	–	38%	82%‡	–	–
CILRS*	[42]	92%	27%‡	–	–	66%	64%‡	–	–
CILS, Waypoint Input†		17%	**0.0%**	0.20%	12.1%	36%	**0.0%**	1.11%	11.70%
MBRL, Waypoint Input†		64%	72%	11.1%	2.96%	48%	54%	20.6%	13.3 %
Our method, Final-State Indicator†		92%	26%	0.05%	0.012%	84%	35%	0.13%	0.38%
Our method, Line Segment Final-St. Indicator†		84%	42%	**0.03%**	0.295%	88%	33%	0.12%	0.14%
Our method, Region Final-St. Indicator†		84%	56%	**0.03%**	0.131%	88%	54%	0.13%	0.22%
Our method, Gaussian Final-St. Mix.†		92%	6.3%	0.04%	**0.005%**	**100%**	12%	0.11%	**0.04%**
Our method, Region Final-St. Indicator S.†		92%	2.8%	0.021%	0.099%	92%	4.0%	0.11%	1.85%
Our method, Gaussian Final-St. Mix. S.†		**100%**	1.7%	**0.03%**	**0.005%**	92%	**0.0%**	**0.05%**	0.15%

		Town01 (training conditions)				Town02 (test conditions)			
Static Nav. Method		Success	Ran Red Light	Wrong lane	Off road	Success	Ran Red Light	Wrong lane	Off road
CIRL*	[123]	93%	–	–	–	68%	–	–	–
CAL*	[191]	92%	–	–	–	68%	–	–	–
MT*	[122]	81%	–	–	–	78%	–	–	–
CIL*	[41]	86%	83%	–	–	44%	82%	–	–
CILRS*	[42]	95%	27%	–	–	90%	64%	–	–
CILS, Waypoint Input†		28%	**0.0%**	0.38%	10.23%	36%	**0.0%**	1.69%	16.82%
MBRL, Waypoint Input†		96%	78%	14.3%	1.94%	96%	73%	19.6 %	0.75%
Our method, Final-State Indicator†		**100%**	48%	0.05%	**0.002%**	**100%**	52%	0.10%	**0.13%**
Our method, Gaussian Final-St. Mixture†		96%	0.83%	**0.01%**	0.08%	96%	**0.0%**	**0.03%**	0.14%
Our method, Gaussian Final-St. Mix. S.†		96%	**0.0%**	0.04%	0.07%	92%	**0.0%**	0.18%	0.27%

6.4.6 Robustness to Errors in Goal-Specification

Towards questions (3) (flexibility) and (4) (noise-robustness), we analyze the performance of our method when the path planner is heavily degraded, to understand its stability and reliability. We use the *Gaussian Final-State Mixture* goal likelihood.

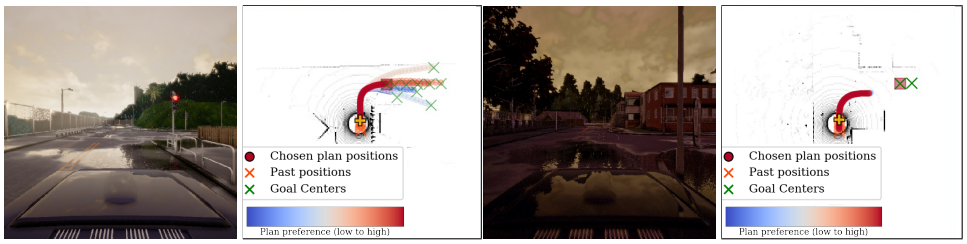


Figure 6.13: Tolerating bad goals. The planner prefers goals in the distribution of expert behavior (on the road at a reasonable distance). *Left*: Planning with $1/2$ decoy goals. *Right*: Planning with all goals on the wrong side of the road.

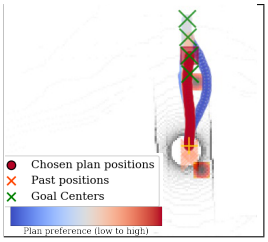


Figure 6.14: Test-time plans steering around potholes.

Navigating with high-variance waypoints. As a test of our model's capability to stay in the distribution of demonstrated behavior, we designed a "decoy waypoints" experiment, in which *half* of the waypoints are highly perturbed versions of the other half, serving as distractions for our Gaussian Final-State Mixture imitative planner. The perturbation distribution is $\mathcal{N}(0, \sigma = 8m)$; each waypoint is perturbed with a standard deviation of 8 meters. We observed surprising robustness to decoy waypoints. Examples of this robustness are shown in Fig. 6.13. In Table 6.5, we report the success rate and the mean number of planning rounds for failed episodes in the "$1/2$ distractors" row. These numbers indicate our method can execute dozens of planning rounds without decoy waypoints causing a catastrophic failure, and often it can execute the hundreds necessary to achieve the goal. One failure mode of this approach is when decoy waypoints lie on a valid off-route path at intersections, which temporarily confuses the planner.

Navigating with waypoints on the wrong side of the road. We also designed an experiment to test our method under systemic bias in the route planner. Our method is provided waypoints on the wrong side of the road (in CARLA, the left side), and tasked with following the directions of these waypoints while staying on the *correct* side of the road (the right side). In order for the value of $q(\mathbf{s}|\phi)$ to outweigh the influence of these waypoints, we increased the ϵ hyperparameter. We found our method to still be very effective at navigating, and report results in Table 6.5. We also investigated providing very coarse 8-meter wide regions to the Region Final-State likelihood; these always include space in the wrong lane and off-road (Fig. 6.12). Nonetheless, on Town01 Dynamic, this approach still achieved an overall success rate of 48%. Taken together towards question (4), our results indicate that our method is *fairly robust to errors in goal-specification*.

6.4.7 Producing Unobserved Behaviors to Avoid Novel Obstacles

To further investigate our model's flexibility to test-time objectives (question 3), we designed a pothole avoidance experiment. We simulated potholes in the environment by randomly inserting them in the cost map near waypoints. We ran our method with a test-time-only cost map of the simulated potholes by combining goal likelihoods (F) and (G), and compared to our method that did not incorporate the cost map (using (F) only, and thus had no incentive to avoid potholes). We recorded the number of collisions with potholes. In Table 6.5, our method with cost incorporated avoided most potholes while avoiding collisions with the environment. To do so, it drove closer

Table 6.5: **Robustness to waypoint noise and test-time pothole adaptation**. Our method is robust to waypoints on the wrong side of the road and fairly robust to decoy waypoints. Our method is flexible enough to safely produce behavior not demonstrated (pothole avoidance) by incorporating a test-time cost. Ten episodes are collected in each Town.

		Town01 (training conditions)			Town02 (test conditions)		
Waypointer	Extra Cost	Success	Wrong lane	Potholes hit	Success	Wrong lane	Potholes hit
Noiseless waypointer		100%	0.00%	177/230	100%	0.41%	82/154
Waypoints wrong lane		100%	0.34%	–	70%	3.16%	–
$1/2$ waypoints distracting		70%	–		50%	–	–
Noiseless waypointer	Pothole	90%	1.53%	10/230	70%	1.53%	35/154

to the centerline, and occasionally entered the opposite lane. Our model internalized obstacle avoidance by staying on the road and demonstrated its flexibility to obstacles not observed during training. Fig. 6.14 shows an example of this behavior.

Pothole details: We simulated potholes in the environment by randomly inserting them in the cost map near each waypoint i with offsets distributed $\mathcal{N}_i(\mu{=}[-15\text{m}, 0\text{m}], \Sigma = \text{diag}([1, 0.01]))$, (i.e. mean-centered on the right side of the lane 15m before each waypoint). We inserted pixels of root cost $-1e3$ in the cost map at a single sample of each $\mathcal{N}_i$, binary-dilated the cost map by $1/3$ of the lane-width (spreading the cost to neighboring pixels), and then blurred the cost map by convolving with a normalized truncated Gaussian filter of $\sigma = 1$ and truncation width 1.

6.4.8 Plan Reliability Estimation

Besides using our model to make a best-effort attempt to reach a user-specified goal, the fact that our model produces explicit likelihoods can also be leveraged to test the *reliability* of a plan by evaluating whether reaching particular waypoints will result in human-like behavior or not. This capability can be quite important for real-world safety-critical applications, such as autonomous driving, and can be used to build a degree of fault tolerance into the system. We designed a classification experiment to evaluate how well our model can recognize safe and unsafe plans. We planned our model to known good waypoints (where the expert actually went) and known bad waypoints (off-road) on 1650 held-out test scenes. We used the planning criterion to classify these as good and bad plans and found that we can detect these bad plans with 97.5% recall and 90.2% precision. This result indicates imitative models could be effective in estimating the reliability of plans.

We determined a threshold on the planning criterion by single-goal planning to the expert's final location on offline validation data and setting it to the criterion's mean minus one stddev. Although a more intelligent calibration could be performed by analyzing the information retrieval statistics on the offline validation, we found this simple calibration to yield reasonably good performance. We used 1650 test scenes to perform classification of plans to three different types of waypoints 1) where the expert actually arrived at time T (89.4% reliable), 2) waypoints 20m ahead along the waypointer-provided route, which are often near where the expert arrives (73.8% reliable) 3) the same waypoints from 2), shifted 2.5m off of the road (2.5% reliable). This shows that our learned model exhibits a strong preference for valid waypoints. Therefore, a waypointer that provides expert waypoints via 1) half of the time, and slightly out-of-distribution waypoints via 3) in the other half, an "unreliable" plan classifier achieves 97.5% recall and 90.2% precision.

6.5 Discussion

We proposed "Imitative Models" to combine the benefits of IL and MBRL. Imitative Models are probabilistic predictive models able to plan interpretable expert-like trajectories to achieve new goals. Inference with an Imitative Model resembles trajectory optimization in MBRL, enabling it to both *incorporate new goals* and *plan to them* at test-time, which IL cannot. Learning an Imitative Model resembles offline IL, enabling it to circumvent the difficult reward-engineering and costly online data collection necessities of MBRL. We derived families of flexible goal objectives and showed our model can successfully incorporate them without additional training. Our method substantially outperformed six IL approaches and an MBRL approach in a dynamic simulated autonomous driving task. We showed our approach is robust to poorly specified goals, such as goals on the wrong side of the road. We believe our method is broadly applicable in settings where expert demonstrations are available, flexibility to new situations is demanded, and safety is paramount.

Chapter 7

Forecasting Multi-Agent Motion Trajectories for Explicitly-Planned Interactions

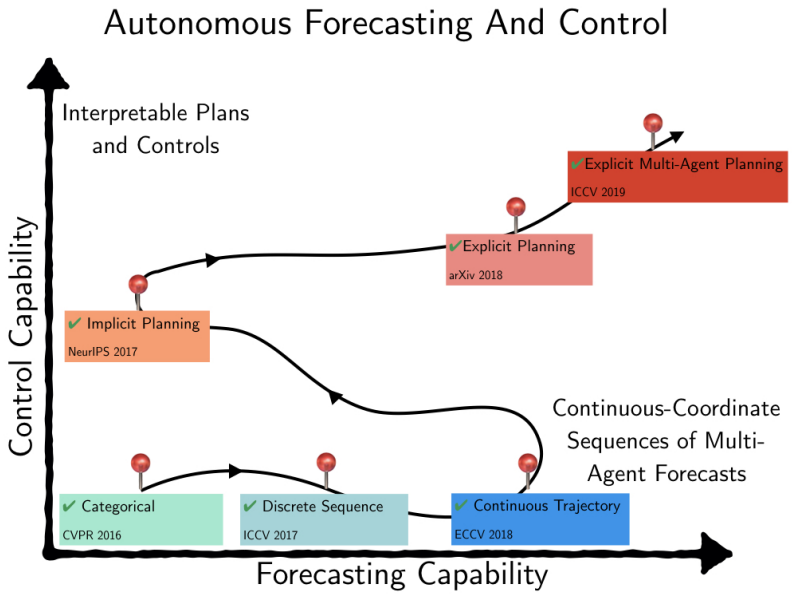


Figure 7.1: This chapter focuses on building a joint forecasting and control approach capable of explicitly planning and interpretably forecasting *multiple agents*.

7.1 Introduction

This Chapter's situation within the Thesis is depicted in Fig. 7.1. In many situations, the behaviors of other *uncontrolled* agents is the dominant source of uncertainty in the environment. We build upon our prior work on explicitly forecasting and explicitly planning by expanding the space of both to *reason over multiple agents*. This will enable approaches to jointly reason over the future behaviors of other agents in order to explicitly plan controls.

For autonomous vehicles (AVs) to behave appropriately on roads populated by human-driven vehicles, they must be able to reason about the uncertain intentions and decisions of other drivers

from rich perceptual information. Towards these capabilities, we present a probabilistic forecasting model of future interactions of a variable number of multiple agents. We perform both standard forecasting and the novel task of conditional forecasting, which reasons about how all agents will likely respond to the goal of a controlled agent (here, the AV). We train models on real and simulated data to forecast vehicle trajectories given past positions and LIDAR. Our evaluation shows that our model is substantially more accurate in multi-agent driving scenarios compared to existing state-of-the-art. Beyond its general ability to perform conditional forecasting queries, we show that our model's predictions of all agents improve when conditioned on knowledge of the AV's goal, further illustrating its capability to model agent interactions.

Autonomous driving requires reasoning about the future behaviors of agents in a variety of situations: at stop signs, roundabouts, crosswalks, when parking, when merging etc. In multi-agent settings, each agent's behavior affects the behavior of others. Motivated by people's ability to reason in these settings, we present a method to forecast multi-agent interactions from perceptual data, such as images and LIDAR. Beyond forecasting the behavior of all agents, we want our model to *conditionally forecast* how other agents are likely to respond to different decisions each agent could make. We want to forecast what other agents would likely do in response to a robot's intention to achieve a goal. This reasoning is essential for agents to make good decisions in multi-agent environments: they must reason how their future decisions could affect the multi-agent system around them. Examples of forecasting and conditioning forecasts on robot goals are shown in Fig. 7.2 and Fig. 7.3. Videos of the outputs of our approach are available at https://sites.google.com/view/precog.

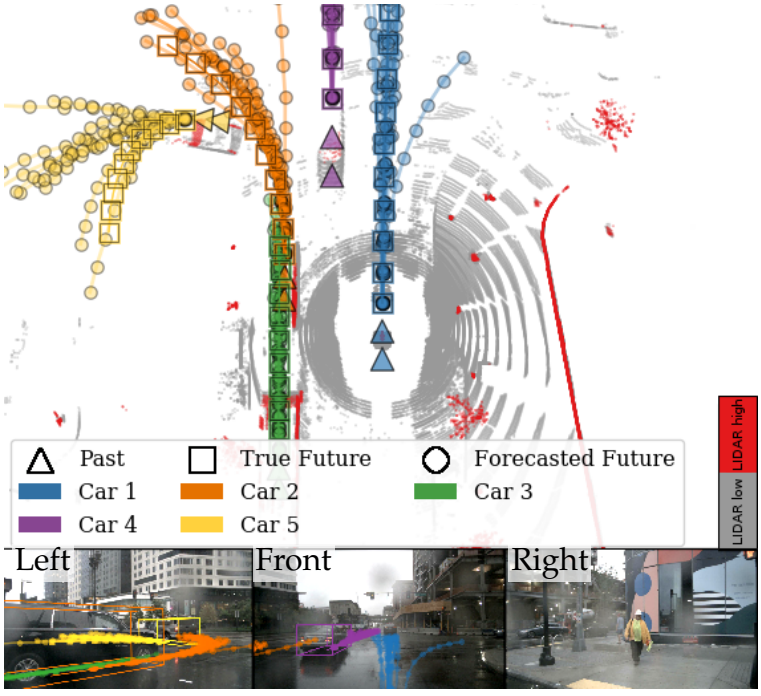


Figure 7.2: Forecasting on nuScenes [32]. The input to our model is a high-dimensional LIDAR observation, which informs a distribution over all agents' future trajectories.

Throughout the paper, we use *goal* to mean a future states that an agent desires. *Planning* means the algorithmic process of producing a sequence of future decisions (in our model, choices of latent values) likely to satisfy a goal. *Forecasting* means the prediction of a sequence of likely future states; forecasts can either be single-agent or multi-agent. Finally, *conditional forecasting* means *forecasting*

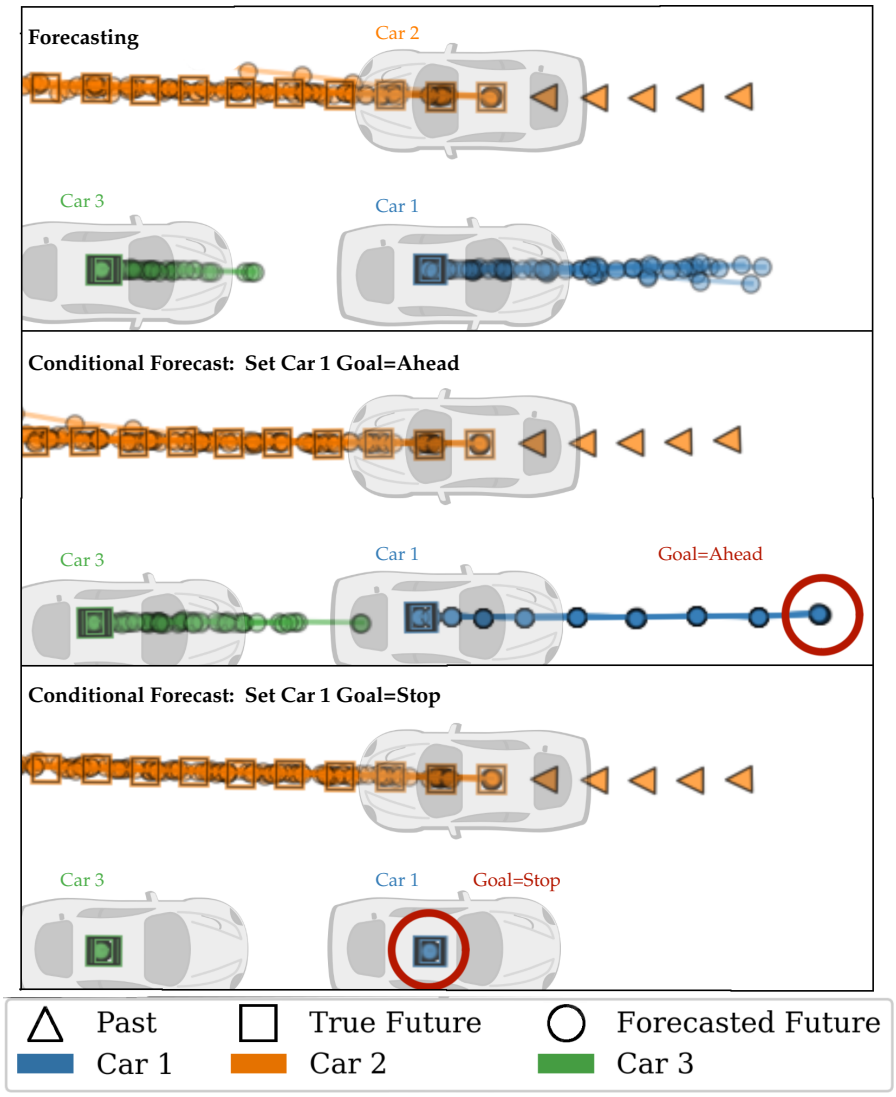


Figure 7.3: Conditioning the model on different Car 1 goals produces different predictions: here it forecasts Car 3 to move if Car 1 yields space, or stay stopped if Car 1 stays stopped.

by conditioning on one or more agent goals. By *planning* an agent's decisions to a *goal* and sampling from the other agents's stochastic decisions, we perform *multi-agent conditional forecasting*. Although we plan future decisions in order to perform conditional forecasting, executing these plans on the robot is outside the scope of this work.

Towards conditional forecasting, we propose a factorized flow-based generative model that forecasts the joint state of all agents. Our model reasons probabilistically about plausible future interactions between agents given rich observations of their environment. It uses latent variables to capture the uncertainty in other agents' decisions. Our key idea is the use of *factorized* latent variables to model decoupled agent decisions even though agent dynamics are coupled. Factorization across agents and time enable us to *query* the effects of changing an arbitrary agent's decision at an arbitrary time step. Our contributions are:

1. **State-of-the-art multi-agent vehicle forecasting:** We develop a multi-agent forecasting model called Estimating Social-forecast Probabilities (ESP) that uses *exact* likelihood inference (unlike

VAEs or GANs) to outperform three state-of-the-art methods on real and simulated vehicle datasets [32, 50].

2. **Goal-conditioned multi-agent forecasting:** We present the *first* generative multi-agent forecasting method able to condition on agent goals, called PREdiction Conditioned on Goals (PRECOG). After modelling agent interactions, conditioning on one agent's goal alters the predictions of other agents.
3. **Multi-agent imitative planning objective:** We derive a data-driven objective for motion planning in multi-agent environments. It balances the likelihood of reaching a goal with the probability that expert demonstrators would execute the same plan. We use this objective for offline planning to known goals, which improves forecasting performance.

7.2 Related Work

Multi-agent modeling and forecasting is a challenging problem for control applications in which agents react to each other concurrently. Safe control requires faithful models of reality to anticipate dangerous situations before they occur. Modeling dependencies between agents is especially critical in tightly-coupled scenarios such as intersections.

Game-theoretic planning: Traditionally, multi-agent planning and game theory approaches explicitly model multiple agents' policies or internal states, usually by generalizing Markov decision processes (MDPs) to multiple decisions makers [39, 221]. These frameworks facilitate reasoning about collaboration strategies, but suffer from "state space explosion" intractability except when interactions are known to be sparse [131] or hierarchically decomposable [58].

Multi-agent forecasting: Data-driven approaches have been applied to forecast complex interactions between multiple pedestrians [8, 20, 55, 77, 128], vehicles [46, 113, 148], and athletes [54, 110, 114, 210, 248, 250]. These methods attempt to generalize from previously observed interactions to predict multi-agent behavior in new situations. Forecasting is related to Imitation Learning [144], which learns a model to mimic demonstrated behavior. In contrast to some Imitation Learning methods, e.g. behavior cloning [157], behavior forecasting models are not executed in the environment of the observed agent – they are instead predictive models of the agent. In this sense, forecasting can be considered non-interactive Imitation Learning without execution.

Forecasting for control and planning: Generative models for multi-agent forecasting and control have been proposed. In terms of multi-agent forecasting, our work is related to [193] which uses a conditional VAE [99] encoding of the joint states of multiple agents together with recurrent cells to predict future human actions. However, our work differs in three crucial ways. First, we model continual co-influence between agents, versus "robot-only" influence, where an agent's responses to the human are not modeled. Second, our method uses contextual visual information useful for generalization to many new scenes. Third, we model interactions between more than two vehicles *jointly*. While [91] assumes conditional independencies for computational reasons, we do not, as they impose minimal overhead.

We consider scenarios in which the model may control one of the agents (a "robot"). In terms of planned control, our method generalizes imitative models [176]. In [176], single-agent forecasting models are used for deterministic single-agent planning. Our work instead considers multi-agent forecasting, and therefore must plan over a distribution of possible paths: from our robot's perspective, the future actions of other human drivers are uncertain. By modeling co-influence, our robot's trajectory are conditional on the (uncertain) future human trajectories, and therefore future robots states are necessarily uncertain. Thus, our work proposes a nontrivial extension for imitative models: we consider future path planning uncertainty induced by the uncertain actions of other

agents in a multi-agent setting. While [176] could implicitly model other agents through its visual conditioning, we show explicit modeling of other agents yields better forecasting results, in addition to giving us the tools to predict responses to agent's plans.

7.3 Deep Multi-Agent Forecasting

In this section, we will describe our likelihood-based model for multi-agent forecasting, and then describe how we use it to perform planning and multi-agent conditional forecasting. First, we define our notation and terminology. We treat our multi-agent system as a continuous-space, discrete-time, partially-observed Markov process, composed of A agents (vehicles) that interact over T time steps. We model all agent positions at time t as $\mathbf{S}_t \in \mathbb{R}^{A\times D}$, where $D=2$. $\mathbf{S}_t^a$ represents agent a's (x, y) coordinates on the ground plane. We assume there is one "robot agent" (e.g. the AV) and $A-1$ "human agents" (e.g. human drivers that our model cannot control). We define $\mathbf{S}_t^r \doteq \mathbf{S}_t^1 \in \mathbb{R}^D$ to index the robot state, and $\mathbf{S}_t^h \doteq \mathbf{S}_t^{2:A} \in \mathbb{R}^{(A-1)\times D}$ to index the human states. Bold font distinguishes variables from functions. Capital English letters denote random variables. We define $t=0$ to be the current time. Subscript absence denotes *all* future time steps, and superscript absence denotes *all* agents, e.g. $\mathbf{S} \doteq \mathbf{S}_{1:T}^{1:A} \in \mathbb{R}^{T\times A\times D}$.

Each agent has access to environment perception $\phi \doteq \{\mathbf{s}_{-\tau:0}, \boldsymbol{\chi}\}$, where τ is the number of past multi-agent positions we condition on and $\boldsymbol{\chi}$ is a high-dimensional observation of the scene. $\boldsymbol{\chi}$ might represent LIDAR or camera images, and is the robot's observation of the world. In our setting, LIDAR is provided as $\boldsymbol{\chi} = \mathbb{R}^{200\times 200\times 2}$, with $\boldsymbol{\chi}_{ij}$ representing a 2-bin histogram of points above and at ground level in $0.5\mathrm{m}^2$ cells. Although our perception is robot-centric, each agent is modeled to have access to $\boldsymbol{\chi}$.

7.3.1 Estimating Social-forecast Probability (ESP)

We propose a data-driven likelihood-based generative model of multi-agent interaction to probabilistically predict T-step dynamics of a multi-agent system: $\mathbf{S} \sim q(\mathbf{S}|\phi;\mathcal{D})$, where $\mathcal{D}$ is training data of observed multi-agent state trajectories. Our model learns to map latent variables $\mathbf{Z}$ via an invertible function f to multi-agent trajectories $\mathbf{S}$ conditioned on ϕ. f's invertibility induces $q(\mathbf{S}|\phi)$, a *pushforward distribution* [130], also known as an *invertible generative model* [48, 67, 73, 100, 171]. Invertible generative models can efficiently and exactly compute probabilities of samples. Here, it means we can compute the probability of joint multi-agent trajectories, critical to our goal of *planning* with the model. We name the model "Estimating Social-forecast Probabilities" (ESP). $\mathbf{S}$ is sampled from q as follows:

$$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}); \quad \mathbf{S} = f(\mathbf{Z}; \phi); \quad \mathbf{S}, \mathbf{Z} \in \mathbb{R}^{T\times A\times D}. \tag{7.1}$$

Our latent variables $\mathbf{Z} \doteq \mathbf{Z}_{1:T}^{1:A}$ factorize across agents and time, which allows us to *decide* agent a's reaction at time t by setting $\mathbf{Z}_t^a \leftarrow \mathbf{z}_t^a$, discussed later. Our model is related to the R2P2 single-agent generative model [171], which constructs a deep likelihood-based generative model for single-agent vehicle forecasting. For multi-step prediction, we generalize R2P2's autoregressive one-step single-agent prediction for the multi-agent setting, and assume a one-step time delay for agents to react to each other:

$$\mathbf{S}_t^a \;=\; \mu_\theta^a(\mathbf{S}_{1:t-1}, \phi) + \sigma_\theta^a(\mathbf{S}_{1:t-1}, \phi) \cdot \mathbf{Z}_t^a \;\in \mathbb{R}^D, \tag{7.2}$$

where $\mu_\theta^a(\cdot)$ and $\sigma_\theta^a(\cdot)$ are neural network functions (with trainable weights θ) outputting a one-step mean prediction $\boldsymbol{\mu}_t^a \in \mathbb{R}^D$ and standard-deviation matrix $\boldsymbol{\sigma}_t^a \in \mathbb{R}^{D\times D}$ of agent a, defining the

system's transition function q as

$$q(\mathbf{S}_t|\mathbf{S}_{1:t-1},\phi) = \prod_{a=1}^{A} \mathcal{N}(\mathbf{S}_t^a; \boldsymbol{\mu}_t^a, \boldsymbol{\Sigma}_t^a), \tag{7.3}$$

where $\boldsymbol{\Sigma}_t^a = \boldsymbol{\sigma}_t^a \boldsymbol{\sigma}_t^{a\top}$. Note that equation 7.2 predicts the a^{th} agent's state $\mathbf{S}_t^a$ given the previous *multi-agent* states $\mathbf{S}_{1:t-1}$. We can see that given $\mathbf{S}_{1:t-1}$, the one-step prediction in equation 7.2 is unimodal Gaussian. However, multi-step predictions are generally multimodal given the recursive nonlinear conditioning of neural network outputs $\boldsymbol{\mu}_t^a$ and $\boldsymbol{\sigma}_t^a$ on previous predictions. The final joint of this model can be written as

$$q(\mathbf{S}|\phi) = \prod_{t=1}^{T} q(\mathbf{S}_t|\mathbf{S}_{1:t-1},\phi). \tag{7.4}$$

7.3.2 Model Implementation

To implement our model $q(\mathbf{S}|\phi)$, we design neural networks that output $\boldsymbol{\mu}_t^a$ and $\boldsymbol{\sigma}_t^a$. Similar to [171], we expand $\mu_\theta^a(\cdot)$ to represent a "Verlet" step, which predicts a constant-velocity mean when $\mathbf{m}_t^a = m_\theta^a(\mathbf{S}_{1:t-1},\phi) = 0$:

$$\mathbf{S}_t^a = \underbrace{2\mathbf{S}_{t-1}^a - \mathbf{S}_{t-2}^a + m_\theta^a(\mathbf{S}_{1:t-1},\phi)}_{\boldsymbol{\mu}_t^a} + \underbrace{\sigma_\theta^a(\mathbf{S}_{1:t-1},\phi)}_{\boldsymbol{\sigma}_t^a} \cdot \mathbf{Z}_t^a. \tag{7.5}$$

A high-level diagram of our implementation shown in Fig. 6.6. Recall $\phi = \{\mathbf{s}_{-\tau:0}, \boldsymbol{\chi}\}$: the context contains the past positions of all agents, $\mathbf{s}_{-\tau:0}$, and a feature map $\boldsymbol{\chi}$, implemented as LIDAR observed by the robot. We encode $\mathbf{s}_{-\tau:0}$ with a GRU. A CNN processes $\boldsymbol{\chi}$ to $\boldsymbol{\Gamma}$ at the same spatial resolution as $\boldsymbol{\chi}$. Features for each agent's predicted position $\mathbf{S}_t^a$ are computed by interpolating into $\boldsymbol{\Gamma}$ as $\boldsymbol{\Gamma}(\mathbf{S}_t^a)$. Positional "social features" for agent a are computed: $\mathbf{S}_t^a - \mathbf{S}_t^b \; \forall \; b \in A \setminus \{a\}$, as well as visual "social features" $\gamma_t^a = \boldsymbol{\Gamma}(\mathbf{s}_t^1) \oplus \cdots \oplus \boldsymbol{\Gamma}(\mathbf{s}_t^A)$. The social features, past encoding, and CNN features are fed to a per-agent GRU, which produces $\mathbf{m}_t^a$ and $\boldsymbol{\sigma}_t^a$ in equation 7.5. We train with observations of expert multi-agent interaction $\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)$ by maximizing likelihood with respect to our model parameters θ. We use shared parameters to produce $\boldsymbol{\Gamma}$ and the past encoding.

Flexible-count implementation: While the implementation described so far is limited to predict for a *fixed-count* of agents in a scene, we also implemented a *flexible-count* version. There are two flavors of a model that is flexible in practice. (1) A fully-flexible model applicable to any scene with agent count $A_{\text{test}} \in \mathbb{N}$. (2) A partially-flexible model applicable to any scene with agent count $A_{\text{test}} \in \{1..A_{\text{train}}\}$, controlled by a hyperparameter upper-bound A_{train} set at training time. To implement (1), the count of model parameters must be *independent* of A_{test} in order for the same architecture to apply to scenes with different counts of agents. To implement (2), "missing agents" must not affect the *joint distribution over the existing agents*, equivalent to ensuring $\partial \mathbf{S}^{\text{existing}} / \partial \mathbf{Z}^{\text{missing}} = 0$ in our framework. We implemented (2) by using a mask $M \in \{0,1\}^{A_{\text{train}}}$ to mask features of missing agents. In this model, we shared parameters across agents, and trained it on data with varying counts of agents.

Both past and future trajectories for each agent are represented in each agent's own *local* coordinate frame at $t = 0$, with agent's forward axis pointing along the agent's yaw at $t = 0$. Each agent a observes positions of the other agents in the coordinate frame of agent a. We use a 9-layer fully-convolutional network with stride 1 and 32 channels per layer, and kernel sizes of 3×3, to process χ into a feature grid $\boldsymbol{\Gamma}$ at the same spatial resolution as χ. The LIDAR is mounted on the first agent, thus it is generally more informative about nearby agents. This enables the prediction to be learned *relative* to the agent, with global context obtained by feature map interpolation. At each time step, each agent's predicted future position $\mathbf{s}_t^a$ is bilinearly-interpolated into $\boldsymbol{\Gamma}$: $\boldsymbol{\Gamma}(\mathbf{s}_t^a)$, which

ensures $^{\mathrm{d}\boldsymbol{\Gamma}(\mathbf{s}_t^a)}/_{\mathrm{d}\mathbf{s}_t^a}$ exists. The "SocialMapFeat" component performs this interpolation by converting the positions (in meters) to feature grid coordinates (in $0.5\ ^{\mathrm{meters}}/_{\mathrm{cell}}$), and bilinearly-interpolating each into the feature map $\boldsymbol{\Gamma}$. The interpolation is performed by retrieving the features at the corners of the nearest unit square to the current continuous position.

We also employed an additional featurization scheme, termed "whiskers". Instead of interpolating only at $\mathbf{s}_t^a$, we interpolated at nearby positions subsampled from arcs relative to $\mathbf{s}_t^a$ at various radii. By letting $\mathbf{s}_t^a - \mathbf{s}_{t-1}^a$ define the predicted orientation, the arcs were generated by evenly sampling 7 points along arcs of length $^{5\pi}/_4$ at radii $[1, 2, 4, 8, 16, 32]$ meters, which loosely simulates the forecasted agent's future field-of-view. The midpoint of each arc lied along the ray from $\mathbf{s}_{t-1}^a$ through $\mathbf{s}_t^a$. After interpolating at points $\{\omega_n\}_{n=1}^{42}$, the resulting feature is of size $8 \cdot 7 \cdot 6$ (8 is the size of the last dimension of Γ, 7 is the number of points per arc, and 6 is the number of arcs). We found this approach to yield superior performance and employed it in the R2P2-MA baseline, as well as all of our methods. The full details of the architecture are provided in Tab. 7.1.

Finally, we performed additional featurization in the nuScenes setting by replacing χ with a *signed-distance transform*, similar to [171]. It provides a spatially-smoother input to the convolutional network, which we found augmented performance. The signed distance transform (SDT) of $\chi_c \in \mathbb{R}^{H \times W}$ can be computed by first binarizing to $\chi_c \in \{0, 1\}^{H \times W}$ and using the Euclidean distance transform (DT), which is commonly provided (*e.g.* in `scipy`). We compute it by binarizing with threshold τ: $\mathrm{SDT}(\chi_c, \tau) = \mathrm{DT}(\chi_c \geq \tau) - \mathrm{DT}(\chi_c < \tau)$, then clipping the result to $[-10, 1]$, and finally normalizing to $[0, 1]$. For LIDAR channels, we use $\tau = 5$. When we use the already-binarized road prior, binarization is unnecessary.

We trained our model with stochastic gradient descent using the Adam optimizer with learning rate $1 \cdot 10^{-4}$, with minibatch size 10, until validation-set performance (of $\hat{e}$, as discussed in the main paper) showed no improvement for a period of 10 epochs.

Table 7.1: Detailed ESP Architecture that implements $\mathbf{s}_{1:T}^{1:A} = f(\mathbf{z}_{1:T}^{1:A}, \phi)$. Typically, $T = 20$, $A = 5$, $D = 2$. In CARLA, $H = W = 100$. In nuScenes, $H = W = 200$. An asterisk $(^*)$ denotes a component whose output is masked in the flexible-count version of ESP by using the agent-presence mask $M \in \{0, 1\}^{A_{\text{train}}}$, discussed in Sec. 7.3.2.

Component	Input [dimensionality]	Layer or Operation	Output [dimensionality]	Details
Static featurization of context: $\phi = \{\chi, \mathbf{s}_{-\tau:0}^{1:A}\}$. *Shared parameters for each agent.*				
MapFeat	$\chi\ [H, W, 2]$	2D Convolution	$^1\chi\ [H, W, 32]$	3×3 stride 1, ReLu
MapFeat	$^{i-1}\chi\ [H, W, 32]$	2D Convolution	$^i\chi\ [H, W, 32]$	3×3 stride 1, ReLu, $i \in [2, \ldots, 8]$
MapFeat	$^8\chi\ [H, W, 32]$	2D Convolution	$\boldsymbol{\Gamma}\ [H, W, 8]$	3×3 stride 1, ReLu
PastRNN*	$\mathbf{s}_{-\tau:0}^{1:A}\ [\tau + 1, A, D]$	RNN	$^1\alpha^{1:A}\ [A, 128]$	GRU across time dimension
PastRNN	$\alpha\ [A, 128]$	$^1\alpha^a \oplus \sum_{b \in \{1..A\} \setminus a} {}^1\alpha^b$	$^2\alpha^a\ [256]$	Index, Concat, & Sum for agent-a context
Dynamic generation via double loop: for $t \in \{0, \ldots, T-1\}$, for $a \in \{1, \ldots, A\}$. *Shared or separate parameters for each agent.*				
SocialFeat*	$\mathbf{s}_t^{1:A}\ [AD]$	$\mathbf{s}_t^a - \mathbf{s}_t^b, b \in \{1..A\} \setminus a$	$^0\eta_t^a\ [AD - D]$	Agent displacements
SocialFeatMLP	$^0\eta_t^a\ [AD - D]$	Affine (FC)	$^1\eta_t^a\ [200]$	Tanh activation
SocialFeatMLP	$^1\eta_t^a\ [200]$	Affine (FC)	$^2\eta_t^a\ [50]$	Identity activation
WhiskerMapFeat	$\omega_1 \ldots \omega_N\ [42, D]$	Interpolate	$w_t^a = \boldsymbol{\Gamma}(\omega_1) \oplus \cdots \oplus \boldsymbol{\Gamma}(\omega_N)\ [8 \cdot 42]$	Interpolate ahead of the sample's P.O.V.
SocialMapFeat*	$\mathbf{s}_t^{1:A}\ [AD]$	Interpolate	$\gamma_t^a = \boldsymbol{\Gamma}(\mathbf{s}_t^1) \oplus \cdots \oplus \boldsymbol{\Gamma}(\mathbf{s}_t^A)\ [8A]$	Differentiable interpolation, concat. $(\oplus)$
JointFeat	$\gamma_t^a, \mathbf{s}_0^{1:A}, {}^2\eta_a, {}^2\alpha^a, w_t^a$	$\gamma_t^a \oplus \mathbf{s}_0^{1:A} \oplus {}^2\eta_a \oplus {}^2\alpha^a \oplus w_t^a$	$\rho_t^a\ [8A + AD + 50 + 256 + 336]$	Concatenate $(\oplus)$
FutureRNN	$\rho_t^a\ [8A + AD + 50 + 256]$	RNN	$^1\rho_t^a\ [50]$	GRU
FutureMLP	$^1\rho_t^a\ [50]$	Affine (FC)	$^2\rho_t^a\ [200]$	Tanh activation
FutureMLP	$^2\rho_t^a [200]$	Affine (FC)	$m_t^a\ [D],\ \xi_t^a\ [D, D]$	Identity activation
MatrixExp	$\xi_t^a\ [D, D]$	$\exp(\xi_t^a + \xi_t^{a,\mathrm{transpose}})$	$\boldsymbol{\sigma}_t^a\ [D, D]$	Differentiable Matrix Exponential [171]
VerletStep	$\mathbf{s}_t, \mathbf{s}_{t-1}, m_t^a, \boldsymbol{\sigma}_t^a, \mathbf{z}_t^a$	$2\mathbf{s}_t - \mathbf{s}_{t-1} + m_t^a + \boldsymbol{\sigma}_t^a \mathbf{z}_t^a$	$\mathbf{s}_{t+1}^a\ [D]$	(Eq. 7.5)

7.3.3 Alternate Joint PDF forms

The original joint can be expanded over each agent:

$$q(\mathbf{S}|\phi) \;=\; \prod_{t=1}^{T} q(\mathbf{S}_t|\mathbf{S}_{1:t-1},\phi) \;=\; \prod_{t=1}^{T}\prod_{a=1}^{A} \mathcal{N}(\mathbf{S}_t^a;\boldsymbol{\mu}_t^a,\boldsymbol{\Sigma}_t^a).$$

Additionally, the change-of-variables rule yields an equivalent density [48, 67, 100, 171]:

$$q(\mathbf{S}|\phi) = \mathcal{N}(f^{-1}(\mathbf{S};\phi);0,I)\left|\det\frac{\mathrm{d}f}{\mathrm{d}\mathbf{Z}}|_{\mathbf{Z}=f^{-1}(\mathbf{S};\phi)}\right|^{-1},$$

We can derive expressions via the rollout equation equation 7.5, reproduced here as equation 7.6, which implicitly defines f:

$$\mathbf{S}_t^a = \underbrace{2\mathbf{S}_{t-1}^a - \mathbf{S}_{t-2}^a + m_\theta^a(\mathbf{S}_{1:t-1},\phi)}_{\boldsymbol{\mu}_t^a} + \underbrace{\sigma_\theta^a(\mathbf{S}_{1:t-1},\phi)}_{\boldsymbol{\sigma}_t^a}\cdot\mathbf{Z}_t^a. \tag{7.6}$$

The full Jacobian is given as:

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{Z}} = \begin{bmatrix} \frac{\partial\mathbf{S}_1}{\partial\mathbf{Z}_1} & 0 & \dots & 0 \\ \frac{\partial\mathbf{S}_2}{\partial\mathbf{Z}_1} & \frac{\partial\mathbf{S}_2}{\partial\mathbf{Z}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \frac{\partial\mathbf{S}_T}{\partial\mathbf{Z}_1} & \frac{\partial\mathbf{S}_T}{\partial\mathbf{Z}_2} & \dots & \frac{\partial\mathbf{S}_T}{\partial\mathbf{Z}_T} \end{bmatrix},$$

where

$$\frac{\partial\mathbf{S}_t}{\partial\mathbf{Z}_t} = \begin{bmatrix} \boldsymbol{\sigma}_t^1 & 0 & \dots & 0 \\ \frac{\partial\mathbf{S}_t^2}{\partial\mathbf{Z}_t^1} & \boldsymbol{\sigma}_t^2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \frac{\partial\mathbf{S}_t^A}{\partial\mathbf{Z}_t^1} & \frac{\partial\mathbf{S}_t^A}{\partial\mathbf{Z}_t^2} & \dots & \boldsymbol{\sigma}_t^A \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma}_t^1 & 0 & \dots & 0 \\ 0 & \boldsymbol{\sigma}_t^2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \boldsymbol{\sigma}_t^A \end{bmatrix}$$

Due to the block triangular nature of the Jacobian and applying Laplace expansion along the diagonal:

$$\det\frac{\mathrm{d}f}{\mathrm{d}\mathbf{Z}} = \prod_t \det\frac{\partial\mathbf{S}_t}{\partial\mathbf{Z}_t} = \prod_{t=1}^{T}\prod_{a=1}^{A}\det\,\sigma_t^a(\mathbf{S}_{1:t-1},\phi).$$

$\mathbf{Z}=f^{-1}(\mathbf{S};\phi)$ is given by computing each $\mathbf{Z}_t^a = \left(\sigma_t^a(\mathbf{S}_{1:t-1},\phi)\right)^{-1}\left(\mathbf{S}_t^a - \mu_t^a(\mathbf{S}_{1:t-1},\phi)\right)$. Algorithmically, the functions f and f^{-1} are implemented separately, each with a *double for-loop over agents and time*. Note that since we use RNNs to produce $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$, the forward f and its inverse must be computed in the same direction by stepping the RNN's forward in time over the input $\mathbf{S}$. To aid implementation, we use the following checks to ensure f is a bijection: $||\mathbf{Z} - f^{-1}(f(\mathbf{Z},\phi),\phi)||_\infty < \epsilon$, $||\mathbf{S} - f(f^{-1}(\mathbf{S},\phi),\phi)||_\infty < \epsilon$.

7.3.4 PREdiction Conditioned On Goals (PRECOG)

A distinguishing feature of our generative model for multi-step, multi-agent prediction is its latent variables $\mathbf{Z} \doteq \mathbf{Z}_{1:T}^{1:A}$ that factorizes over agents and time. Factorization makes it possible to use the model for highly flexible conditional forecasts. Conditional forecasts predict how other agents would likely respond to different robot decisions at different moments in time. Since robots are not merely passive observers, but one of potentially many agents, the ability to anticipate how they affect others is critical to their ability to plan useful, safe, and effective actions, critical to their utility within a planning and control framework [129].

Human drivers can appear to take highly stochastic actions in part because we cannot observe their goals. In our model, the source of this uncertainty comes from the latent variables $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In practical scenarios, the robot knows its own goals, can choose its own actions, and can plan a course of action to achieve a desired goal. Recall from equation 7.2 that *one-step* agent predictions are conditionally independent from each other give the previous multi-agent states. Therefore, certainty in the latent state $\mathbf{Z}_t^a$ corresponds to certainty of the a^{th} agent's *reaction* to the multi-agent system at time t. Different values of $\mathbf{Z}_t^a$ correspond to different ways of reacting to the same information. Deciding values of $\mathbf{Z}_t^a$ corresponds to controlling the agent a. We can therefore implement control of the robot via assigning values to its latent variables $\mathbf{Z}^r \leftarrow \mathbf{z}^r$. In contrast, human reactions $\mathbf{Z}_t^h$ cannot be decided by the robot, and so remain uncertain from the robot's perspective and can only be influenced by their conditioning on the robot's previous states in $\mathbf{S}_{1:t-1}$, as seen Fig. 7.4b. Therefore, to generate conditional-forecasts, we decide $\mathbf{z}^r$, sample $\mathbf{Z}^h$, concat. $\mathbf{Z} = \mathbf{z}^r \oplus \mathbf{Z}^h$, and warp $\mathbf{S} = f(\mathbf{Z}, \phi)$. This factorization of latent variables easily facilitates conditional forecasting. To forecast $\mathbf{S}$, we can fix $\mathbf{z}^r$ while sampling the human agents' reactions from their distribution $p(\mathbf{Z}^h) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, which are warped via equation 7.1.

7.3.5 Multi-Agent Planning

We discussed how forecasting can condition on some value of $\mathbf{z}^r$, but not yet how to find *desirable* values of $\mathbf{z}^r$, e.g. values that would safely direct the robot towards its goal location. We perform multi-agent planning by optimizing an objective $\mathcal{L}$ w.r.t.the control variables $\mathbf{z}^r$, which allows us to produce the "best" forecasts under $\mathcal{L}$.

While many valid objectives can be adopted, we use imitative models (IM), which estimate the likeliest state trajectory an expert driver "would have taken" to satisfy a goal, based on prior expert demonstrations [176]. IM modeled single-agent environments where robot trajectories are planned without consideration of other agents. Multi-agent planning is different, because future robot states are uncertain (states $\mathbf{S}_{t>1}^r$ in Fig. 7.4b), even when conditioned on control variables $\mathbf{z}^r$, because of the uncertainty in surrounding human drivers $\mathbf{Z}^h$.

We generalize IM to multi-agent environments, and plan w.r.t.the uncertainty of human drivers close by. First, we chose a "goal likelihood" function that represents the likelihood that a robot reaches its goal $\mathcal{G}$ given state trajectory $\mathbf{S}$. For instance, the likelihood could be a waypoint $\mathbf{w} \in \mathbb{R}^D$ the robot should approach: $p(\mathcal{G}|\mathbf{S}, \phi) = \mathcal{N}(\mathbf{w}; \mathbf{S}_T^r, \epsilon\mathbf{I})$. Second, we combine the goal likelihood with a "prior probability" model of safe multi-agent state trajectories $q(\mathbf{S}|\phi)$, learned from expert demonstrations. Note that unlike many other generative multi-agent models, we can compute the probability of generating $\mathbf{S}$ from $q(\mathbf{S}|\phi)$ exactly, which is critical to our planning approach. This results in a "posterior" $p(\mathbf{S}|\mathcal{G}, \phi)$. Finally, we plan a goal-seeking path in the learned distribution of

demonstrated multi-agent behavior under the log-posterior probability derived as:

$$\log \mathbb{E}_{\mathbf{Z}^h}[p(\mathbf{S}|\mathcal{G},\phi)] \;\geq\; \mathbb{E}_{\mathbf{Z}^h}[\log p(\mathbf{S}|\mathcal{G},\phi)] \tag{7.7}$$

$$= \mathbb{E}_{\mathbf{Z}^h}[\log q(\mathbf{S}|\phi)\cdot p(\mathcal{G}|\mathbf{S},\phi)] - \log p(\mathcal{G}|\phi) \tag{7.8}$$

$$\mathcal{L}(\mathbf{z}^r,\mathcal{G}) = \mathbb{E}_{\mathbf{Z}^h}[\log \underbrace{q(f(\mathbf{Z})|\phi)}_{\text{multi-agent prior}} + \log \underbrace{p(\mathcal{G}|f(\mathbf{Z}),\phi)}_{\text{goal likelihood}}], \tag{7.9}$$

where equation 7.7 follows by Jensen's inequality, which we use to avoid the numerical issue of a single sampled $\mathbf{Z}^h$ dominating the batch. equation 7.8 follows from Bayes' rule and uses our learned model q as the prior. In equation 7.9, we drop $p(\mathcal{G}|\phi)$ because it is constant w.r.t.$\mathbf{z}^r$. Recall $\mathbf{Z} = \mathbf{z}^r \oplus \mathbf{Z}^h$ is the concatenation of robot and human control variables. The robot can plan using our ESP model by optimizing equation 7.9:

$$\mathbf{z}^{r*} = \arg\max_{\mathbf{z}^r} \mathcal{L}(\mathbf{z}^r,\mathcal{G}). \tag{7.10}$$

Other objectives might be used instead, e.g. maximizing the posterior probability of the robot trajectories only. This may place human agents in unusual, precarious driving situations, outside the prior distribution of "usual driving interaction". equation 7.10 encourages the robot to avoid actions likely to put the joint system in an unexpected situation.

7.3.6 Planning and Forecasting Algorithms

To execute planning, we perform gradient ascent to approximately solve the optimization problem equation 7.10. Recall the latent joint behavior is $\mathbf{Z} \doteq \mathbf{Z}^{1:A}_{1:T}$, the latent human behavior is $\mathbf{Z}^h \doteq \mathbf{Z}^{2:A}_{1:T}$, and the robot behavior is $\mathbf{z}^r \doteq \mathbf{z}^1_{1:T}$. We approximate the expectation in equation 7.9 with a sample expectation over K samples from $p(\mathbf{Z}^h) = \mathcal{N}(\mathbf{0},\mathbf{I})$, denoted ${}^{1:K}\mathbf{z}^h$, where the k^{th} sample is ${}^k\mathbf{z}^h$. Each of these samples for the latent human behavior is combined with the single latent robot plan, each denoted ${}^k\mathbf{z} = [\mathbf{z}^r, {}^k\mathbf{z}^h]$. This batch is denoted ${}^{1:K}\mathbf{z}$. The approximation of equation 7.9 is then given as

$$\hat{L}({}^{1:K}\mathbf{z},\mathcal{G},\phi) = \frac{1}{K}\sum_{k=1}^{K} \log(q(f({}^k\mathbf{z})|\phi)p(\mathcal{G}|f({}^k\mathbf{z}),\phi)), \tag{7.11}$$

with $\hat{L}$ parameterized by (q,f,p), and f's dependence on ϕ dropped for notational brevity. The ${}^{1:K}\mathbf{z}^h$ is redrawn before each gradient ascent step on equation 7.11. This procedure is illustrated in Alg. 7.

Algorithm 7 MULTIIMITATIVEPLAN(q,f,p,ϕ,K)

Define $\hat{L}$ with q,f,p
Initialize $\mathbf{z}^r_{1:T} \sim \mathcal{N}(0,I)$
while not converged **do**
  ${}^{1:K}\mathbf{z}^h \overset{\text{iid}}{\sim} N(0,I)$
  $\mathbf{z}^r_{1:T} \leftarrow \mathbf{z}^r_{1:T} + \nabla_{\mathbf{z}^r_{1:T}} \hat{L}({}^{1:K}\mathbf{z},\mathcal{G},\phi)$
end while
return $\mathbf{z}^r_{1:T}$

In our implementation, we use $K = 12$, track the $\mathbf{z}^r_{1:T}$ that achieved the best $\hat{L}$ score, terminate the ascent if the best $\mathbf{z}^r_{1:T}$ has not improved in 10 steps, and return the corresponding best $\mathbf{z}^r_{1:T}$. To

initialize $\mathbf{z}^r_{1:T}$ more robustly, we sample a full ${}^{1:K}\mathbf{z}$ multiple times (15), and use the $\mathbf{z}^r$ corresponding to the best $\hat{L}$. We can also run the planning over multiple initial samples of $\mathbf{z}^r_{1:T}$.

Now, we further detail how we can use this planning to perform goal-conditioned forecasting. As described in Sec. 7.4.4, we model goals in our experiments by defining our goal-likelihood $p(\mathcal{G}|\mathbf{S}_{1:T}, \phi) = \mathcal{N}(\mathbf{S}^r_T; \mathbf{S}^{*r}_T, 0.1 \cdot \mathbf{I})$, *i.e.* a normal distribution at the controlled agent's last true future position, $\mathbf{S}^{*r}_T$. In general, we can pass any final desired robot position, $\mathbf{s}^{\dagger r}_T$ as the mean of this distribution. Then, we perform goal-conditioned forecasting on a specific scene ϕ to a specific robot goal $\mathbf{s}^{\dagger r}_T$, with our trained multi-agent density q, defined by f. This forecasting is performed by first planning $\mathbf{z}^r$ according to Alg. 7, then sampling $\mathbf{Z}^h$ again to generate stochastic joint outcomes, conditioned on the robot's plan. This procedure is illustrated in Algs. 8 and 9.

Algorithm 8 PRECOG$(q, f, p, \mathbf{s}^{\dagger r}_T, \phi, K)$

1: $\mathbf{z}^r \leftarrow$ MULTIIMITATIVEPLAN(q, f, p, ϕ, K)
2: Sample ${}^{1:K}\mathbf{z}^h_{1:T} \overset{\text{iid}}{\sim} N(0, I)$
3: Forecast ${}^{1:K}\mathbf{s}^{1:A}_{1:T} \leftarrow f({}^{1:K}\mathbf{z}^{1:A}_{1:T}, \phi)$
4: **return** ${}^{1:K}\mathbf{s}^{1:A}_{1:T}$

Algorithm 9 POSPRECOG$(q, f, \mathbf{s}^{\dagger r}_T, \phi, K)$

1: Define $p(\mathcal{G}|\mathbf{S}_{1:T}, \phi) = \mathcal{N}(\mathbf{S}^r_T; \mathbf{s}^{\dagger r}_T, 0.1 \cdot I)$
2: **return** PRECOG$(q, f, p, \mathbf{s}^{\dagger r}_T, \phi, K)$

In Fig. 7.5, we illustrate various forms of the probabilistic graphical models corresponding to our main model (ESP), a baseline model (R2P2-MA), and how the assignment of latent variables ($\mathbf{Z}$) in these models affects the production of the states ($\mathbf{S}$). In Fig. 7.6, we illustrate the graphical models of ESP and PRECOG for $A = 3$.

7.4 Experiments

We first compare our forecasting model against existing state-of-the-art multi-agent forecasting methods, including SocialGAN [77], DESIRE [113]. We also include a baseline model: R2P2-MA (adapted from R2P2 [171] to instead handle multiple agent inputs), which does not model how agents will react to each others' future decisions. Second, we investigate the novel problem of *conditional* forecasting. To quantify forecasting performance, we study scenarios where we have pairs of the robot's true goal and the sequence of joint states. Knowledge of goals should enable our model to better predict what the robot and each agent could do. Third, we ablate the high-dimensional contextual input $\boldsymbol{\chi}$ from our model to determine its relevance to forecasting.

nuScenes dataset: We used the recently-released full nuScenes dataset [32], a real-world dataset for multi-agent trajectory forecasting, in which 850 episodes of 20 seconds of driving were recorded and labelled at 2Hz with the positions of all agents, and synced with many sensors, including LIDAR. We processed each of the examples to train, val, and test splits. Each example has 2 seconds of past and 4 seconds of future positions at 5Hz and is accompanied by a LIDAR map composited from 1 second of previous scans. We also experimented concatenating a binary road mask to χ, indicated as "Road" in our evaluation.

CARLA dataset: We generated a realistic dataset for multi-agent trajectory forecasting and planning with the CARLA simulator [50]. We ran the autopilot in `Town01` for over 900 episodes of 100 seconds

each in the presence of 100 other vehicles, and recorded the trajectory of every vehicle and the autopilot's LIDAR observation. We randomized episodes to either train, validation, or test sets. We created sets of 60,701 train, 7586 validation, and 7567 test examples, each with 2 seconds of past and 2 seconds of future positions at 10Hz. See https://sites.google.com/view/precog for data.

The dataset also includes 100 episodes obtained by following the same procedure in `Town02`. We used this data to further evaluate our ESP model. We applied our saved models (the same models used to report results in the paper) to this data. We generated the CARLA data using version 0.8.4. We used the default vehicle. We used a LIDAR position of $(0.0, 0.0, 2.5)$, with 32 channels, a range of 50, 100,000 points per second, a rotation frequency of 10, an upper FOV limit of 10, and a lower FOV limit of -30. We will release the 100GB of collected data.

7.4.1 Metrics

Log-likelihood: As our models can perform exact likelihood inference (unlike GANs or VAEs), we can precisely evaluate how likely held-out samples are under each model. Test log-likelihood is given by the forward cross-entropy $H(p, q) = -\mathbb{E}_{\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)} \log q(\mathbf{S}^*|\phi)$, which is unbounded for general p and q. However, by perturbing samples from $p(\mathbf{S}^*|\phi)$ with noise drawn from a known distribution η (e.g. a Gaussian) to produce a perturbed distribution p', we can enforce a lower bound [171]. The lower bound is given by $H(p', q) \geq H(p') \geq H(\eta)$. We use $\eta = \mathcal{N}(\mathbf{0}, 0.01 \cdot \mathbf{I})$ (n.b. $H(\eta)$ is known analytically). Our likelihood statistic is:

$$\hat{e} \doteq \big[H(p', q) - H(\eta)\big]/_{(TAD)} \geq 0, \tag{7.12}$$

which has $^{\text{nats}}/_{\text{dim.}}$ units. We call $\hat{e}$ "extra nats" because it represents the (normalized) extra nats above the lower bound of 0. Normalization enables comparison across models of different dimensionalities.

Sample quality: For sample metrics, we must take care not to penalize the distribution when it generates plausible samples different than the expert trajectory. We extend the "minMSD" metric [113, 148, 171] to measure quality of *joint trajectory samples*. The "minMSD" metric samples a model and computes the error of the best sample in terms of MSD. In contrast to the commonly-used average displacement error (ADE) and final displacement error (FDE) metrics that computes the mean Euclidean error from a batch of samples to a *single* ground-truth sample [8, 46, 55, 77, 152], minMSD has the desirable property of not penalizing plausible samples that correspond to decisions the agents could have made, but did not. *This prevents erroneously penalizing models that make diverse behavior predictions*. We hope other multimodal prediction methods will also measure the quality of joint samples with minMSD, given by:

$$\hat{m}_K \doteq \mathbb{E}_{\mathbf{S}^*} \min_{k \in \{1..K\}} ||\mathbf{S}^* - \mathbf{S}^{(k)}||^2/_{(TA)}, \tag{7.13}$$

where $\mathbf{S}^* \sim p(\mathbf{S}^*|\phi), \mathbf{S}^{(k)} \overset{\text{iid}}{\sim} q(\mathbf{S}|\phi)$. We denote the per-agent error of the best *joint* trajectory with

$$\begin{aligned} \hat{m}^a_K &\doteq \mathbb{E}_{\mathbf{S}^* \sim p(\mathbf{S}^*|\phi)} ||\mathbf{S}^{*a} - \mathbf{S}^{a,(k^\dagger)}||^2/_T, \\ k^\dagger &\doteq \arg\min_{k \in \{1..K\}} ||\mathbf{S}^* - \mathbf{S}^{(k)}||^2. \end{aligned} \tag{7.14}$$

7.4.2 Baselines

KDE [149, 179] serves as a useful performance bound on all methods; it can compute both $\hat{m}$ and $\hat{e}$. We selected a bandwidth using the validation data. Note KDE ignores ϕ.

Table 7.2: CARLA and nuScenes multi-agent forecasting evaluation. All CARLA-trained models use `Town01 Train` only, and are tested on `Town02 Test`. No training data is collected from `Town02`. Means and their standard errors are reported. The en-dash (–) indicates an approach unable to compute $\hat{e}$. The R2P2-MA model generalizes [171] to multi-agent. Variants of our ESP method (gray) outperform prior work.

Approach	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Test $\hat{m}_{K=12}$	Test $\hat{e}$
CARLA Town02 Test	2 agents		3 agents		4 agents		5 agents	
KDE	4.488 ± 0.145	8.179 ± 1.523	5.964 ± 0.099	6.029 ± 0.394	7.846 ± 0.087	5.181 ± 0.172	9.610 ± 0.078	5.116 ± 0.097
DESIRE [113]	1.159 ± 0.027	–	1.099 ± 0.018	–	1.410 ± 0.018	–	1.697 ± 0.017	–
SocialGAN [77]	0.902 ± 0.022	–	0.756 ± 0.015	–	0.932 ± 0.014	–	0.979 ± 0.015	–
R2P2-MA [171]	0.454 ± 0.014	0.577 ± 0.004	0.516 ± 0.012	0.640 ± 0.022	0.575 ± 0.011	0.598 ± 0.010	0.632 ± 0.011	0.620 ± 0.010
Ours: ESP, no LIDAR	0.633 ± 0.017	0.579 ± 0.006	0.582 ± 0.014	0.620 ± 0.013	0.655 ± 0.013	0.591 ± 0.006	0.784 ± 0.013	0.584 ± 0.004
Ours: ESP	$\mathbf{0.393 \pm 0.014}$	0.550 ± 0.004	$\mathbf{0.377 \pm 0.011}$	0.529 ± 0.004	0.438 ± 0.010	0.540 ± 0.004	0.565 ± 0.009	0.592 ± 0.004
Ours: ESP, flex. count	0.488 ± 0.017	$\mathbf{0.537 \pm 0.002}$	0.412 ± 0.012	$\mathbf{0.508 \pm 0.001}$	$\mathbf{0.398 \pm 0.010}$	$\mathbf{0.499 \pm 0.001}$	$\mathbf{0.435 \pm 0.011}$	$\mathbf{0.496 \pm 0.001}$
nuScenes Test	2 agents		3 agents		4 agents		5 agents	
KDE	19.375 ± 0.798	3.760 ± 0.015	31.663 ± 0.894	4.102 ± 0.023	41.289 ± 1.170	4.369 ± 0.026	52.071 ± 1.449	4.615 ± 0.028
DESIRE [113]	3.473 ± 0.102	–	4.421 ± 0.130	–	5.957 ± 0.162	–	6.575 ± 0.198	–
SocialGAN [77]	2.119 ± 0.087	–	3.033 ± 0.110	–	3.484 ± 0.129	–	3.871 ± 0.148	–
R2P2-MA [171]	1.336 ± 0.062	0.951 ± 0.007	2.055 ± 0.093	0.989 ± 0.008	2.695 ± 0.100	1.020 ± 0.011	3.311 ± 0.166	1.050 ± 0.012
Ours: ESP, no LIDAR	1.496 ± 0.069	$\mathbf{0.920 \pm 0.008}$	2.240 ± 0.084	$\mathbf{0.955 \pm 0.008}$	3.201 ± 0.113	1.033 ± 0.012	3.442 ± 0.139	1.107 ± 0.018
Ours: ESP	1.325 ± 0.065	0.933 ± 0.008	1.705 ± 0.089	1.018 ± 0.011	2.547 ± 0.095	1.053 ± 0.015	3.266 ± 0.155	1.082 ± 0.013
Ours: ESP, Road	$\mathbf{1.081 \pm 0.053}$	0.929 ± 0.008	$\mathbf{1.505 \pm 0.070}$	1.016 ± 0.011	$\mathbf{2.360 \pm 0.093}$	$\mathbf{1.013 \pm 0.012}$	$\mathbf{2.892 \pm 0.162}$	1.114 ± 0.024
Ours: ESP, Road, flex.	1.464 ± 0.067	0.980 ± 0.003	2.029 ± 0.079	1.001 ± 0.003	2.525 ± 0.099	1.015 ± 0.002	2.933 ± 0.129	$\mathbf{1.029 \pm 0.002}$

DESIRE [113] proposed a conditional VAE model that observes past trajectories and visual context. We re-implemented DESIRE following the authors' description in the paper and supplement, as there is no open-source version available. In our domain, χ is purely LIDAR-based, whereas their model combines image-based semantic segmentation features into the same coordinate frame. We found most provided parameters to work well, except those related to the re-ranking component. The re-ranking often did not improve the trajectories. The best results were obtained with 1 re-ranking step. Whereas DESIRE is trained with a single-agent evidence lower bound (ELBO), our model jointly models multiple agents with an exact likelihood. As DESIRE does not compute multi-agent likelihoods, we cannot compute its $\hat{e}$, nor use it for planning in a multi-agent setting.

SocialGAN [77] proposed a conditional GAN multi-agent forecasting model that observes the past trajectories of all modeled agents, but not $\boldsymbol{\chi}$. We used the authors' public implementation `https://github.com/agrimgupta92/sgan`. We found the default `train.py` parameters yielded poor performance. We achieved significantly better SocialGAN performance by using the network parameters in the `run_traj.sh` script. In contrast to SocialGAN, we model joint trajectories, and can compute likelihoods for planning (and for $\hat{e}$).

R2P2 [171] proposed a likelihood-based conditional generative forecasting model for single-agents. We extend R2P2 to the multi-agent setting and use it as our R2P2-MA model; R2P2 does not jointly model agents. We re-implemented R2P2 following the authors' description in the paper and supplement, as there is no open-source version currently available. We trained it and our model with the forward-cross entropy loss. R2P2-MA's likelihood is given by $q(\mathbf{S}|\phi) = \prod_{a=1}^{A} q^a(\mathbf{S}^a|\phi)$. We extended R2P2 to the multi-agent setting and use it as our R2P2-MA model; R2P2 does not jointly model agents. We can compute R2P2's likelihood, and therefore $\hat{e}$, by assuming independence across agents: $q(\mathbf{S}|\phi) = \prod_{a=1}^{A} q^a(\mathbf{S}^a|\phi)$. Note that since this joint likelihood does not model agent's future actions to influence each other, R2P2 cannot be used for planning in a multi-agent setting. Fig. 7.5 compares the R2P2 baseline to our ESP model.

7.4.3 Multi-Agent Forecasting Experiments

Didactic Example: In the didactic example, a robot (blue) and a human (orange) both navigate in an intersection, the human has a stochastic goal: with 0.5 probability they will turn left, and otherwise they will drive straight. The human always travels straight for 4 time steps, and then reveals its intention by either going straight or left. The robot attempts to drive straight, but will acquiesce to the human if the human turns in front of the robot. We trained our models and evaluate them in Fig. 7.8. Each trajectory has length $T=20$. While both models closely match the training distribution in terms of likelihood, their sample qualities are significantly different. The R2P2-MA model generates samples that crash 50% of the time, because it does not condition future positions for the robot on future positions of the human, and vice-versa. In the ESP model, the robot is able to react to the human's decision during the generation process by choosing to turn when the human turns.

CARLA and nuScenes: We build 10 datasets from CARLA and nuScenes data, corresponding to modeling different numbers of agents $\{2..5\}$. Agents are sorted by their distances to the autopilot, at $t=0$. When 1 agent is included, only the autopilot is modeled; for A agents, the autopilot and the $A-1$ closest vehicles are modeled.

For each method, we report its best test-set score at the best val-set score. In R2P2 and our method, the val-set score is $\hat{e}$. In DESIRE and SocialGAN, the val-set score is $\hat{m}$, as they cannot compute $\hat{e}$. Tab. 7.2 shows the multi-agent forecasting results. **Across all** 10 **settings, our model achieves the best** $\hat{m}$ **and** $\hat{e}$ **scores**. We also ablated our model's access to $\boldsymbol{\chi}$ ("ESP, no LIDAR"), which puts it on equal footing with SocialGAN, in terms of model inputs. Visual context provides a uniform improvement in every case.

Qualitative examples of our forecasts are shown in Fig. 7.9. We observe three important types of multimodality: 1) multimodality in speed along a common specific direction, 2) the model properly predicts diverse plausible paths at intersections, and 3) when the agents are stopped, the model predicts sometimes the agents will stay still, and sometimes they will accelerate forward. The model also captures qualitative social behaviors, such as predicting that one car will wait for another before accelerating. See Sec. 7.4.6 for additional visualizations.

7.4.4 PRECOG Experiments

Now we perform our second set of evaluations. We investigate if our planning approach enables us to sample more plausible joint futures of all agents. Unlike the previous unconditional forecasting scenario, when the robot is using the ESP model for planning, it knows its own goal. We can simulate planning offline by assuming the goal was the state that the robot actually reached at $t=T$, and then planning a path from the current time step to this goal position. We can then evaluate the quality of the agent's path and the stochastic paths of other agents under this plan. While this does not test our model in a full control scenario, it does allow us to evaluate whether conditioning on the goal provides more accurate and higher-confidence predictions. We use our model's multi-agent prior equation 7.4 in the stochastic latent multi-agent planning objective equation 7.9, and define the goal-likelihood $p(\mathcal{G}|\mathbf{S},\phi)=\mathcal{N}(\mathbf{S}_T^r;\mathbf{S}_T^{*r},0.1\cdot\mathbf{I})$, *i.e.* a normal distribution at the controlled agent's last true future position, $\mathbf{S}_T^{*r}$. As discussed, this knowledge might be available in control scenarios where we are confident we can achieve this positional goal. Other goal-likelihoods could be applied to relax this assumption, but this setup allows us to easily measure the quality of the resulting joint samples. We use gradient-descent on equation 7.9 to approximate $\mathbf{z}^{r*}$. The resulting latent plan yields highly likely joint trajectories under the uncertainty of other agents and approximately maximizes the goal-likelihood. Note that since we planned in latent space, the resulting robot trajectory is not fully determined – it can evolve differently depending on the stochasticity of the

other agents. We next illustrate a scenario where joint modeling is critical to accurate forecasting and planning. Then, we conduct planning experiments on the CARLA and nuScenes datasets.

Table 7.4: CARLA multi-agent forecasting evaluation. All CARLA-trained models use `Town01 Train` only, and are tested on `Town01 Test`. Mean scores (and their standard errors) of sample quality $\hat{m}$ equation 7.13, and log likelihood $\hat{e}$ equation 7.12, are shown. The en-dash (–) indicates if an approach cannot compute likelihoods. The R2P2-MA generalizes the single-agent forecasting approach of [171]. Variants of our ESP method (highlighted gray) mostly outperform prior work in the multi-agent CARLA setting. For single agent evaluations, see Tab. 7.5.

Approach	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)
CARLA Town01 Test	2 agents		3 agents		4 agents		5 agents	
DESIRE [113]	1.943 ± 0.033	–	1.587 ± 0.020	–	2.234 ± 0.023	–	2.422 ± 0.017	–
SocialGAN [77]	0.977 ± 0.016	–	0.812 ± 0.013	–	1.098 ± 0.014	–	1.141 ± 0.015	–
R2P2-MA [171]	0.540 ± 0.009	0.625 ± 0.002	0.387 ± 0.008	0.645 ± 0.002	0.690 ± 0.009	0.621 ± 0.002	0.770 ± 0.008	0.618 ± 0.002
Ours: ESP, no LIDAR	0.724 ± 0.013	0.688 ± 0.003	0.719 ± 0.011	0.640 ± 0.002	0.919 ± 0.011	0.650 ± 0.002	1.102 ± 0.011	0.652 ± 0.002
Ours: ESP	$\mathbf{0.311 \pm 0.008}$	0.615 ± 0.002	$\mathbf{0.385 \pm 0.007}$	0.585 ± 0.002	0.509 ± 0.007	0.599 ± 0.002	0.675 ± 0.007	0.630 ± 0.001
Ours: ESP, flex. count	0.415 ± 0.014	$\mathbf{0.531 \pm 0.002}$	0.398 ± 0.011	$\mathbf{0.513 \pm 0.001}$	$\mathbf{0.411 \pm 0.010}$	$\mathbf{0.507 \pm 0.001}$	$\mathbf{0.447 \pm 0.009}$	$\mathbf{0.509 \pm 0.001}$

Table 7.5: Performance in CARLA $A = 1$ (N.B. here the model is identical to R2PA-MA (denoted by *)).

Approach	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)
CARLA Town01 Test	1 agent	
DESIRE [113]	1.067 ± 0.040	–
SocialGAN [77]	0.921 ± 0.031	–
R2P2-MA [171]	*	*
Ours: ESP, no LIDAR	0.496 ± 0.024	0.699 ± 0.006
Ours: ESP	$\mathbf{0.136 \pm 0.010}$	$\mathbf{0.634 \pm 0.006}$

7.4.5 Additional CARLA and nuScenes Evaluations.

We show additional evaluations on CARLA in Tab. 7.4. Table 7.4 shows the `Town01` of the models trained on `Town01` (on separate episodes). We show single-agent CARLA forecasting results in Tab. 7.5. We show histograms of $\hat{m}$ in Fig. 7.11, Fig. 7.12, and Fig. 7.15. We show a comparison to longer time-horizon forecasting in Tab. 7.6. We show a plot of means and their standard errors of $\hat{m}_K$ vs. K in Fig. 7.13.

7.4.5.1 CARLA and nuScenes PRECOG

DESIRE planning baseline: We developed a straightforward planning baseline by feeding an input goal state and past encoding to a two-layer 200-unit ReLU MLP trained to predict the latent state of the robot given training tuples $(x = (\mathcal{H}_X, \mathbf{S}^r_T \sim q_{\text{DESIRE}}(\mathbf{S}|\phi, \mathbf{z}^r)_T)), y = \mathbf{z}^r)$. The latents for the other agents are samples from their DESIRE priors.

Experiments: We use the trained ESP models to run PRECOG on the test-sets in CARLA and nuScenes. Here, we use both $\hat{m}_K$ and $\hat{m}^a_K$ to quantify joint sample quality in terms of all agents and each agent individually. In Tab. 7.3 and Fig. 7.10, we report results of our planning experiments. We observe that our planning approach significantly improves the quality of the joint trajectories. As expected, the forecasting performance improves the most for the planned agent ($\hat{m}^1_K$). Notably, the

Table 7.6: Performance on CARLA Town01 Test with $T = 40$ at 10Hz (4 seconds of future). This data has larger dimensionality than CARLA $T = 20$, 10Hz (2 seconds) data and the nuScenes $T = 20$, 5Hz (4 seconds) data.

Approach	Test $\hat{m}_{K=12}$ (minMSD)	Test $\hat{e}$ (extra nats)
Town01 Test, $T = 20$, 10Hz (2s)	5 agent	
ESP, flex. count	0.447 ± 0.009	0.509 ± 0.001
Town02 Test, $T = 20$, 10Hz (2s)	5 agent	
ESP, flex. count	0.435 ± 0.011	0.496 ± 0.001
Town01 Test, $T = 40$, 10Hz (4s)	5 agent	
ESP, flex. count	2.500 ± 0.077	0.492 ± 0.001
nuScenes Test, $T = 20$, 5Hz (4s)	5 agent	
ESP, flex. count	2.933 ± 0.129	1.029 ± 0.002

forecasting performance of the other agents improves across all datasets and all agents. We see the non-planned-agent improvements are usually greatest for Car 2 ($\hat{m}_K^2$). This result conforms to our intuitions: Car 2 is the *closest* agent to the planned agent, and thus, it the agent that Car 1 influences the most. Qualitative examples of this planning are shown in Fig. 7.10. We observe trends similar to the CARLA planning experiments – the forecasting performance improves the most for the planned agent, with the forecasting performance of the unplanned agent improving in response to the latent plans. See Sec. 7.4.6 for additional visualizations.

We report some remaining results (i.e. for $A = \{3, 4\}$) in Tab. 7.7. We observe similar trends in these results as in $A = 2$ and $A = 5$: PRECOG improves predictions of all agents' future trajectories, and that knowledge of the ego-agent's goal provides improves predictions for closer agents more than farther agents.

7.4.5.2 Robustness to Agent Localization Errors

In real-world data, there may be error in the localization of the other agents ($\mathbf{s}_{-\tau:0}$). We can simulate this error in our test-set by perturbing $\mathbf{s}^a_{-\tau:0}$ with a random vector $v^a \sim \mathcal{N}(\mathbf{0}, \epsilon I_{D\times D})$. We also train a model by injecting noise generated similarly. In Fig. 7.14 we compare nuScenes $A = 2$ ESP models trained without ($M_{\epsilon=0.0}$) and with ($M_{\epsilon=0.1}$) noise injection. We observe that $M_{\epsilon=0.0}$ is much more sensitive to test-time noise than $M_{\epsilon=0.1}$ at all perturbation scales, which shows noise injection is an effective strategy to mitigate the effects of localization error. We also note $M_{\epsilon=0.1}$ improves performance even when the test-data is not perturbed.

7.4.6 Additional Visualizations

We display additional visualization of our results in Figures 7.17, 7.18, 7.19, 7.20, and 7.23. In Fig. 7.17, we show additional forecasting results on the nuScenes dataset. In Fig. 7.18, we show additional forecasting results on the CARLA dataset. In Fig. 7.19, we show additional planning results on the CARLA dataset. In Fig. 7.20, we show additional planning results on the nuScenes dataset. In Fig. 7.21, we show qualitative results of high, medium, and low quality on the CARLA $A = 5$ dataset (ordered by $\hat{m}$), paired with their corresponding $\hat{m}$ scores. In Fig. 7.22, we show qualitative results of high, medium, and low quality on the nuScenes $A = 5$ dataset (ordered by $\hat{m}$), paired with their corresponding $\hat{m}$ scores. In Fig. 7.23, we visualize the planning criterion ($\hat{L}$) across

many different spatio-temporal goal positions in CARLA, which gives a qualitative interpretation of where the model prefers goal. In Fig. 7.24, we visualize the same posterior on nuScenes.

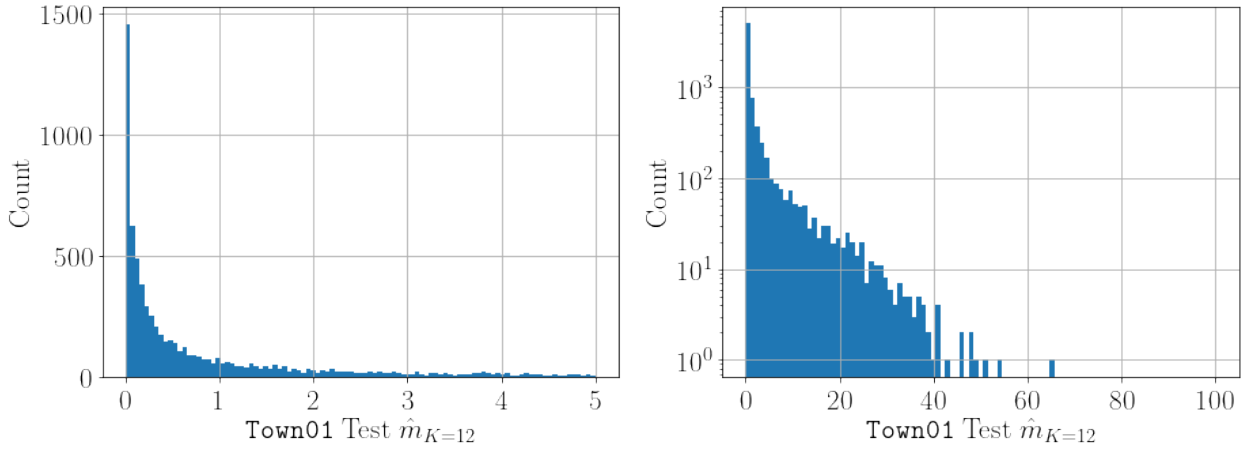
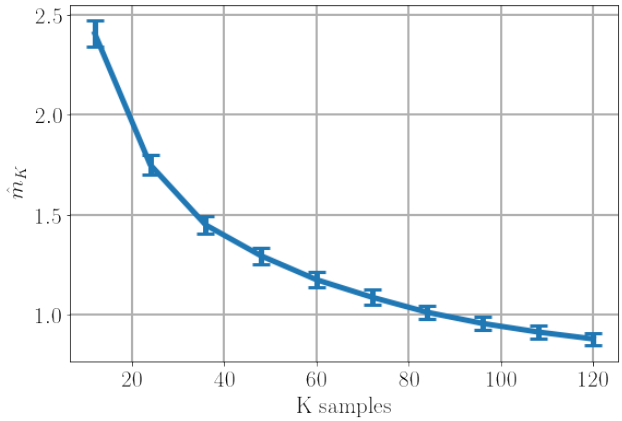
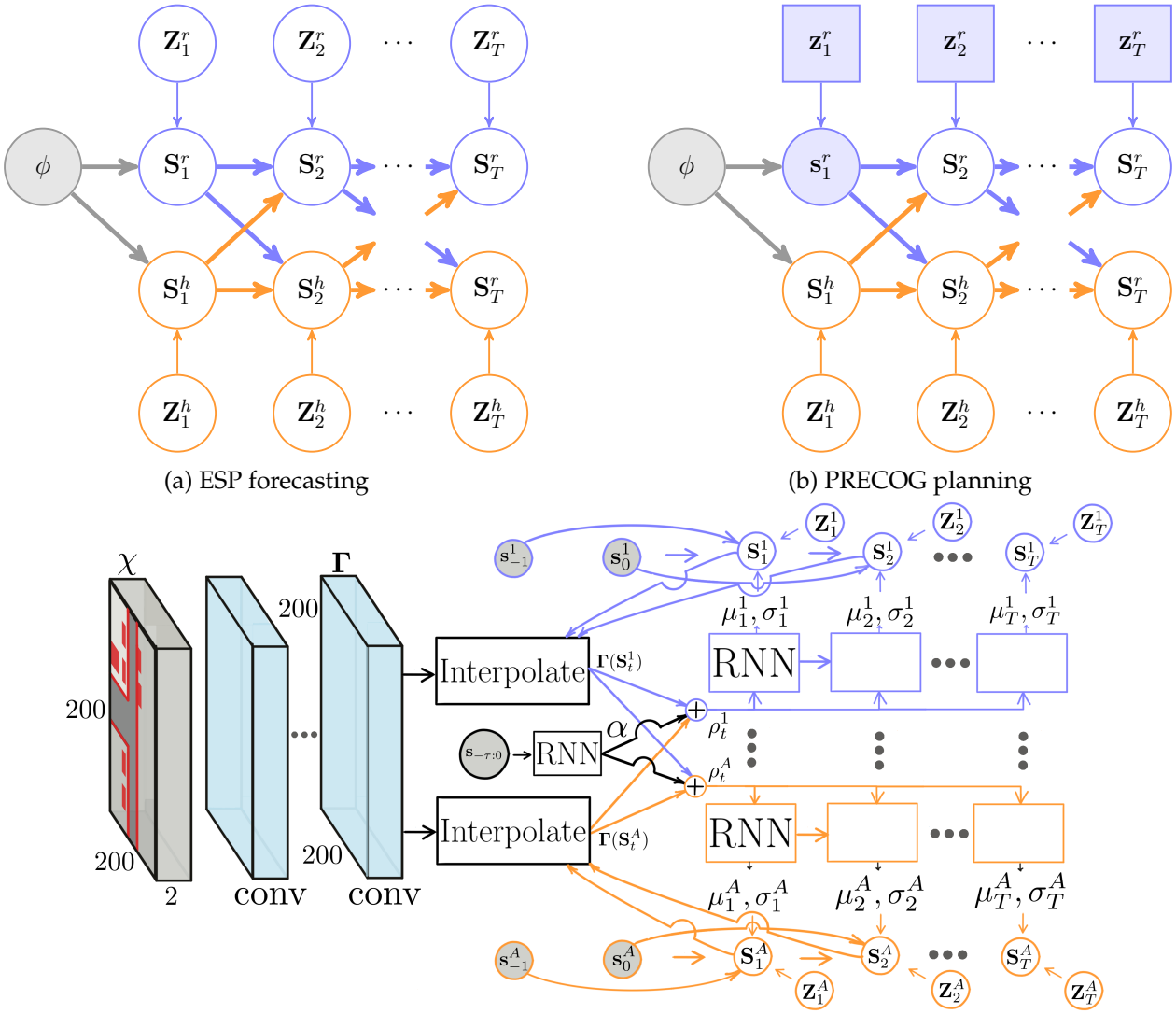


Figure 7.15: Histogram of $\hat{m}_{K=12}$ of forecasts made by the ESP flexible-count model on CARLA `Town01` Test $A = 5$, $T = 40$ at 10Hz (4 seconds of future). The median $\hat{m}_{K=12}$ is 0.38.



(a) Plot of $\hat{m}_K$ vs. K of the ESP flexible-count model on CARLA `Town01` Test $A = 5$, $T = 40$ at 10Hz (4s).

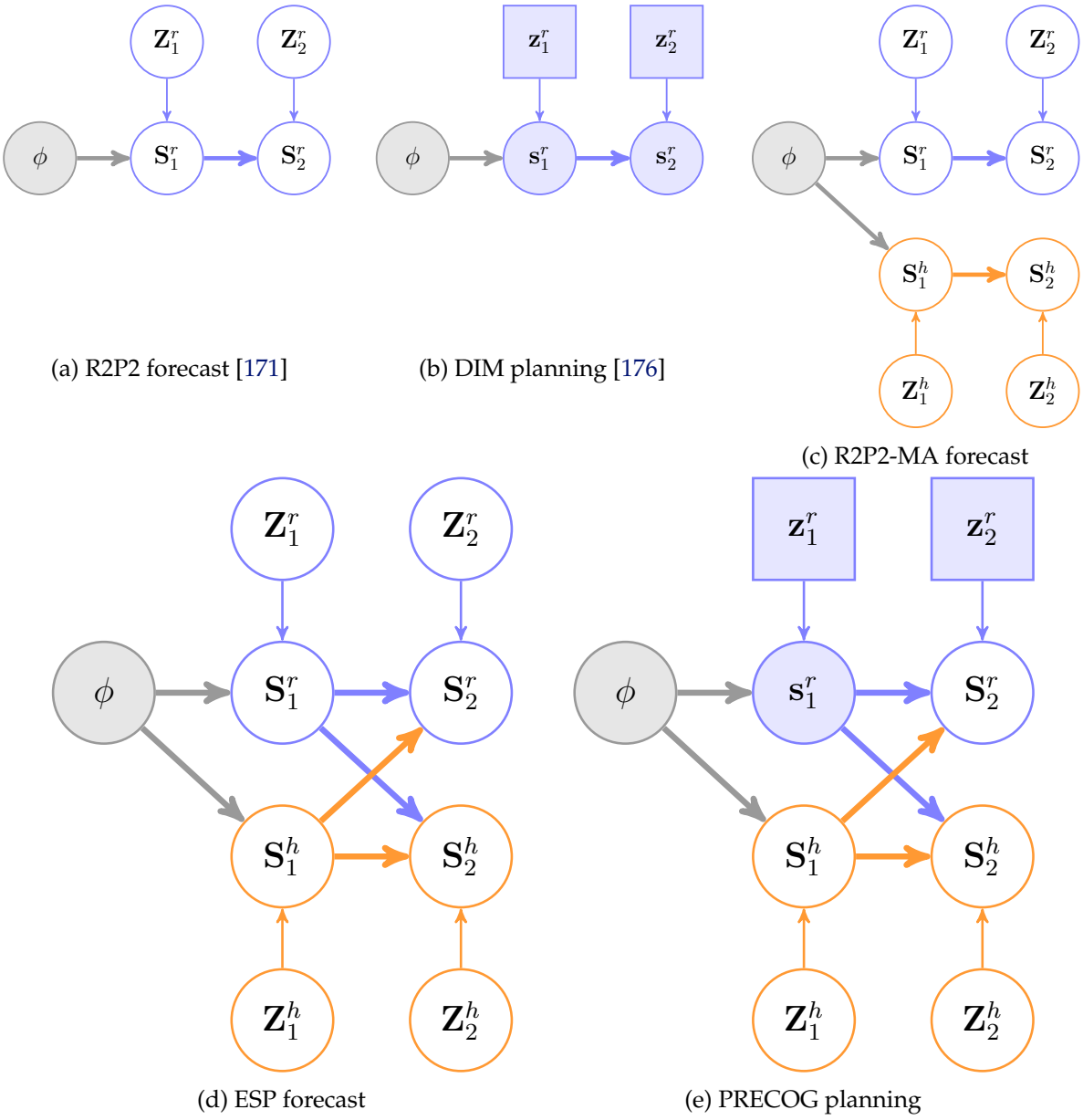


(a) ESP forecasting

(b) PRECOG planning

(c) ESP model implementation

Figure 7.4: Our factorized latent variable model of forecasting and planning shown for 2 agents. In Fig. 7.4a our model uses latent variable $\mathbf{Z}^a_{t+1}$ to represent variation in agent a's *plausible* scene-conditioned reactions to all agents $\mathbf{S}_t$, causing uncertainty in every agents' future states $\mathbf{S}$. Variation exists because of unknown driver goals and different driving styles observed in the training data. Beyond forecasting, our model admits *planning* robot decisions by *deciding* $\mathbf{Z}^r = \mathbf{z}^r$ (Fig. 7.4b). Shaded nodes represent observed or determined variables, and square nodes represent robot decisions [18]. Thick arrows represent grouped dependencies of *non-Makovian* $\mathbf{S}_t$ "carried forward" (a regular edge exists between any pair of nodes linked by a chain of thick edges). Note $\mathbf{Z}$ *factorizes* across agents, isolating the robot's reaction variable $\mathbf{z}^r$. Human reactions remain uncertain ($\mathbf{Z}^h$ is unobserved) and uncontrollable (the robot cannot decide $\mathbf{Z}^h$), and yet the robot's decisions $\mathbf{z}^r$ will still *influence* human drivers $\mathbf{S}^h_{2:T}$ (and vice-versa). Fig. 6.6 shows our implementation.



(a) R2P2 forecast [171]

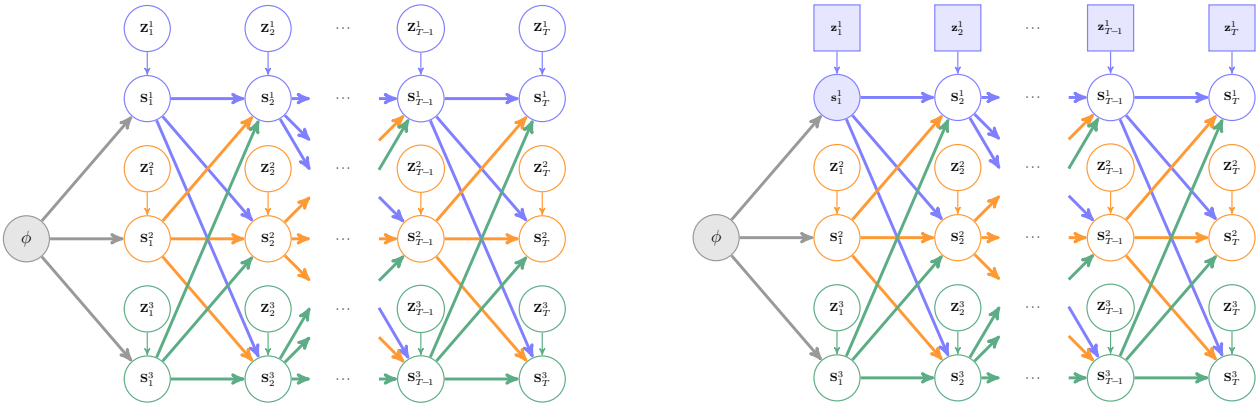
(b) DIM planning [176]

(c) R2P2-MA forecast

(d) ESP forecast

(e) PRECOG planning

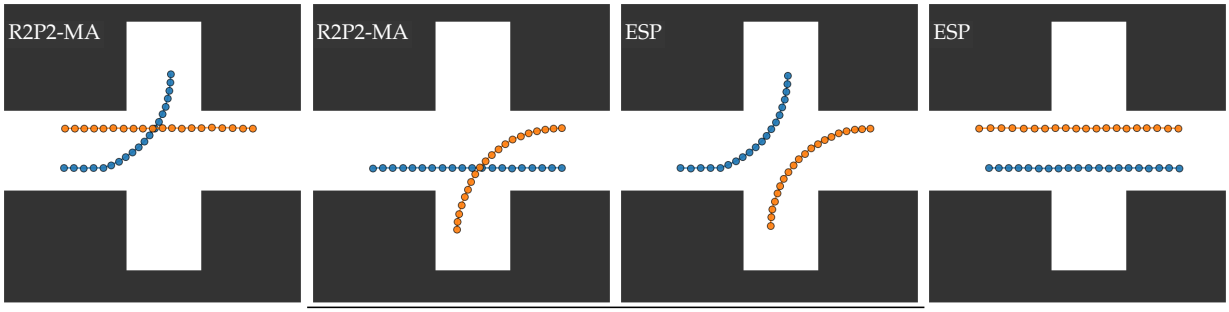
Figure 7.5: Graphical model comparison between prior work (Fig. 7.5a, Fig. 7.5b); a baseline we used (Fig. 7.5c); and our proposed methods (Fig. 7.5d, Fig. 7.5e). All figures show $A = 2$ and two steps of the true T-step horizon. Shaded nodes represent observed variables, and square nodes represent robot decisions. Thick arrows represent non-Makovian "carry-forward" dependencies (i.e. a state can depend on multiple previous states): add a thin arrow for every two nodes connected by a chain of thick arrows. Future reactions are always unknown in the case of the human drivers ("h" superscript), but can be decided in the case of robot ("r" superscipt) planning. How vehicles react affects—and induces uncertianty into—the multi-agent system state $\mathbf{S}$.



(a) ESP forecast (b) PRECOG planning

Figure 7.6: Graphical models of ESP and PRECOG for $A = 3$. See Fig. 7.5's caption for notation.

Figure 7.7: Images from the CARLA simulator [50]. *Left:* frontal view. *Right:* overhead view.



Model	Test $\hat{m}_{K=12}$	Test $\hat{e}$	Forecasting crashes	Planning crashes
R2P2-MA	0.331	0.085	50.8%	49.5%
ESP	**0.000**	**0.031**	**1.17%**	**0.00%**

Figure 7.8: Didactic evaluation. *Left plots:* R2P2-MA cannot model agent interaction, and generates joint behaviors not present in the data. *Right plots:* ESP allows agents to influence each other, and does not generate undesirable joint behaviors.

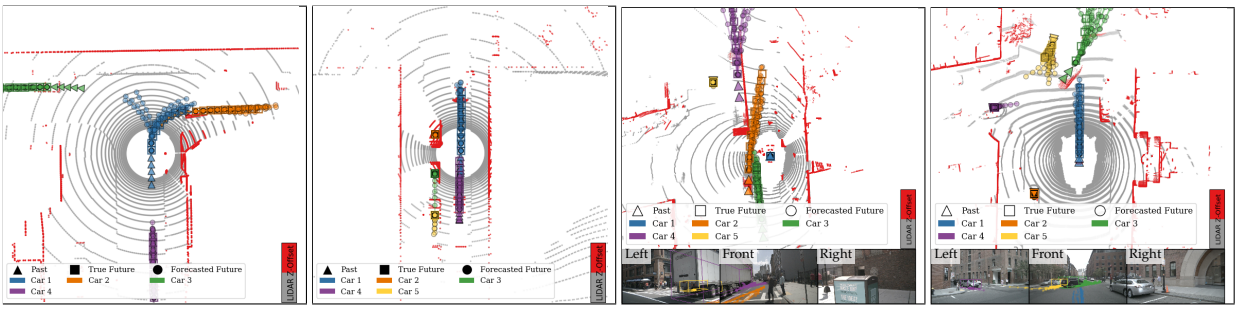
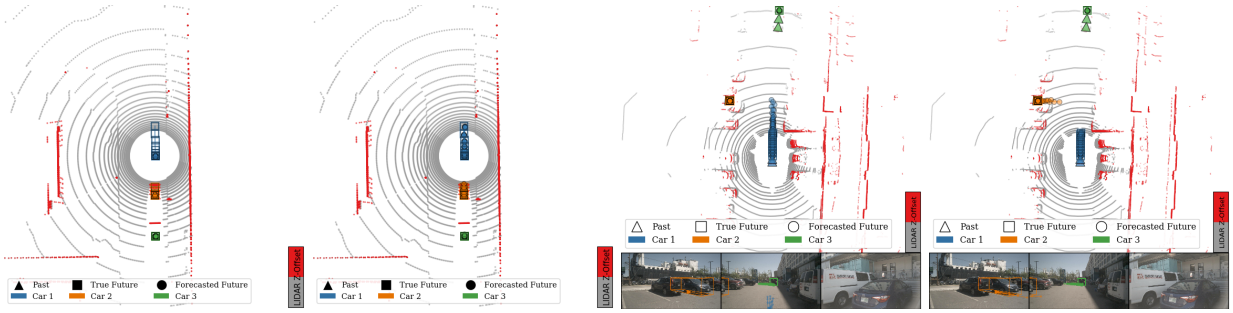


Figure 7.9: Examples of multi-agent forecasting with our learned ESP model. In each scene, 12 joint samples are shown, and LIDAR colors are discretized to near-ground and above-ground. *Left:* (CARLA) the model predicts Car 1 could either turn left or right, while the other agents' future maintain multimodality in their speeds. *Center-left:* The model predicts Car 2 will likely wait (it is blocked by Cars 3 and 5), and that Cars 3 and 5 sometimes move forward together, and sometimes stay stationary. *Center-right:* Car 2 is predicted to overtake Car 1, which itself is forecasted to continue to wait for pedestrians and Car 2. *Right:* Car 4 is predicted to wait for the other cars to clear the intersection, and Car 5 is predicted to either start turning or continue straight.

Table 7.3: Forecasting evaluation of our model on CARLA `Town01 Test` and nuScenes Test data. Planning the robot to a goal position (PRECOG) generates better predictions for all agents. Means and their standard errors are reported.

Data	Approach	Test $\hat{m}_{K=12}$	Test $\hat{m}^{a=1}_{K=12}$	Test $\hat{m}^{a=2}_{K=12}$	Test $\hat{m}^{a=3}_{K=12}$	Test $\hat{m}^{a=4}_{K=12}$	Test $\hat{m}^{a=5}_{K=12}$
CARLA 2	DESIRE [113]	1.837 ± 0.048	1.991 ± 0.066	1.683 ± 0.050	–	–	–
	DESIRE-plan	1.858 ± 0.046	0.918 ± 0.044	2.798 ± 0.073	–	–	–
	ESP	0.337 ± 0.013	0.196 ± 0.009	0.478 ± 0.024	–	–	–
	PRECOG	$\mathbf{0.241 \pm 0.012}$	$\mathbf{0.055 \pm 0.003}$	$\mathbf{0.426 \pm 0.024}$	–	–	–
CARLA 5	DESIRE [113]	2.622 ± 0.030	2.621 ± 0.045	2.422 ± 0.048	2.710 ± 0.066	2.969 ± 0.057	2.391 ± 0.049
	DESIRE-plan	2.329 ± 0.038	0.194 ± 0.004	2.239 ± 0.057	3.119 ± 0.098	3.332 ± 0.090	2.758 ± 0.083
	ESP	0.718 ± 0.012	0.340 ± 0.011	0.759 ± 0.024	0.809 ± 0.025	0.851 ± 0.023	0.828 ± 0.024
	PRECOG	$\mathbf{0.640 \pm 0.011}$	$\mathbf{0.066 \pm 0.003}$	$\mathbf{0.741 \pm 0.024}$	$\mathbf{0.790 \pm 0.024}$	$\mathbf{0.804 \pm 0.022}$	$\mathbf{0.801 \pm 0.024}$
nuScenes 2	DESIRE [113]	3.307 ± 0.093	3.002 ± 0.088	3.613 ± 0.140	–	–	–
	DESIRE-plan	4.528 ± 0.151	0.456 ± 0.015	8.600 ± 0.298	–	–	–
	ESP	1.094 ± 0.053	0.955 ± 0.057	1.233 ± 0.078	–	–	–
	PRECOG	$\mathbf{0.514 \pm 0.037}$	$\mathbf{0.158 \pm 0.016}$	$\mathbf{0.871 \pm 0.070}$	–	–	–
nuScenes 5	DESIRE [113]	6.830 ± 0.204	4.999 ± 0.219	6.415 ± 0.294	7.027 ± 0.360	7.418 ± 0.324	8.290 ± 0.532
	DESIRE-plan	6.562 ± 0.207	2.261 ± 0.100	6.644 ± 0.314	6.184 ± 0.325	9.203 ± 0.448	8.520 ± 0.514
	ESP	2.921 ± 0.175	1.861 ± 0.109	2.369 ± 0.188	2.812 ± 0.188	3.201 ± 0.254	4.363 ± 0.652
	PRECOG	$\mathbf{2.508 \pm 0.152}$	$\mathbf{0.149 \pm 0.021}$	$\mathbf{2.324 \pm 0.187}$	$\mathbf{2.654 \pm 0.190}$	$\mathbf{3.157 \pm 0.273}$	$\mathbf{4.254 \pm 0.586}$



(a) CARLA, ESP (b) CARLA, PRECOG (c) nuScenes, ESP (d) nuScenes, PRECOG

Figure 7.10: Examples of *planned* multi-agent forecasting (PRECOG) with our learned model in CARLA and nuScenes. By using our planning approach and conditioning the robot on its true final position, our predictions of the other agents change, our predictions for the robot become more accurate, and sometimes our predictions of the other agent become more accurate.

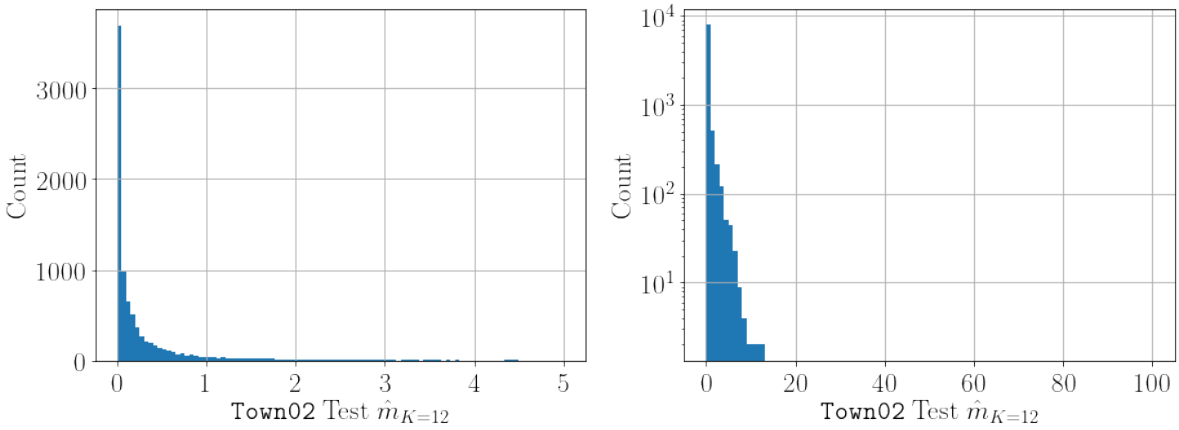


Figure 7.11: Histogram of $\hat{m}_{K=12}$ of forecasts made by the ESP flexible-count model on CARLA `Town02` Test $A = 5$, $T = 20$ at 10Hz (2 seconds of future). The median $\hat{m}_{K=12}$ is 0.09.

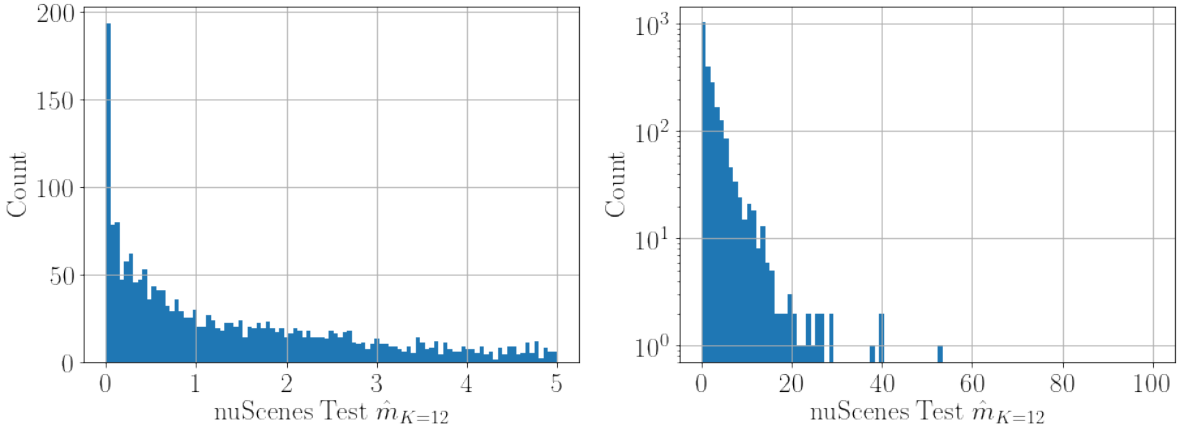
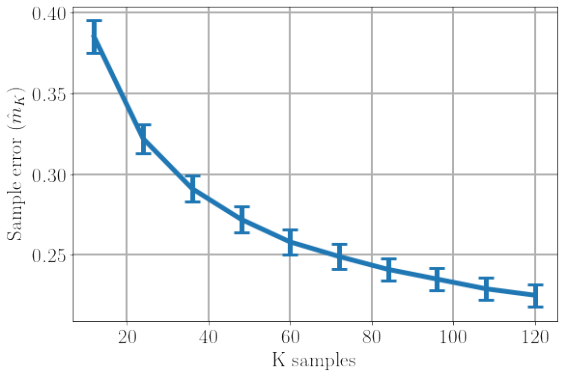
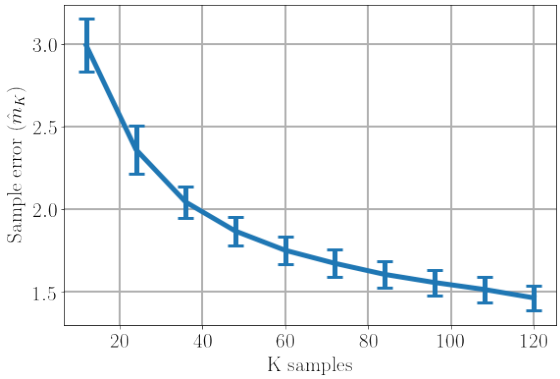


Figure 7.12: Histogram of $\hat{m}_{K=12}$ of forecasts made by the ESP flexible-count model on nuScenes Test $A = 5$, $T = 20$ at 5Hz (4 seconds of future). The median $\hat{m}_{K=12}$ is 1.31.



(a) Plot of $\hat{m}_K$ vs. K of the ESP flexible-count model on CARLA `Town02` Test $A = 5, T = 20$ at 10Hz (2s).



(b) Plot of $\hat{m}_K$ vs. K of the ESP flexible-count model on nuScenes Test $A = 5, T = 20$ at 5Hz (4s).

Figure 7.13: Mean $\hat{m}_K$ and its standard error vs. K in two settings.

Table 7.7: Forecasting evaluation of our model on CARLA `Town01 Test` and nuScenes Test data. Planning the robot to a goal position (PRECOG) generates better predictions for all agents. Means and their standard errors are reported. The en-dash (–) represents statistics of agents that are not present in a dataset.

Data	Approach	Test $\hat{m}_{K=12}$	Test $\hat{m}^{a=1}_{K=12}$	Test $\hat{m}^{a=2}_{K=12}$	Test $\hat{m}^{a=3}_{K=12}$	Test $\hat{m}^{a=4}_{K=12}$	Test $\hat{m}^{a=5}_{K=12}$
CARLA $A=2$	DESIRE	1.837 ± 0.048	1.991 ± 0.066	1.683 ± 0.050	–	–	–
	DESIRE-plan	1.858 ± 0.046	0.918 ± 0.044	2.798 ± 0.073	–	–	–
	ESP	0.337 ± 0.013	0.196 ± 0.009	0.478 ± 0.024	–	–	–
	PRECOG	$\mathbf{0.241 \pm 0.012}$	$\mathbf{0.055 \pm 0.003}$	$\mathbf{0.426 \pm 0.024}$	–	–	–
CARLA $A=3$	DESIRE	1.699 ± 0.032	1.570 ± 0.037	1.661 ± 0.047	1.865 ± 0.047	–	–
	DESIRE-plan	2.343 ± 0.047	0.232 ± 0.009	3.130 ± 0.078	3.667 ± 0.096	–	–
	ESP	0.426 ± 0.013	0.204 ± 0.009	0.556 ± 0.027	0.519 ± 0.021	–	–
	PRECOG	$\mathbf{0.355 \pm 0.012}$	$\mathbf{0.052 \pm 0.003}$	$\mathbf{0.519 \pm 0.025}$	$\mathbf{0.493 \pm 0.020}$	–	–
CARLA $A=4$	DESIRE	2.402 ± 0.038	2.422 ± 0.054	2.065 ± 0.044	2.531 ± 0.071	2.589 ± 0.064	–
	DESIRE-plan	1.828 ± 0.035	0.149 ± 0.004	2.480 ± 0.062	1.256 ± 0.047	3.426 ± 0.098	–
	ESP	0.537 ± 0.011	0.236 ± 0.009	0.615 ± 0.021	0.656 ± 0.023	0.643 ± 0.023	–
	PRECOG	$\mathbf{0.478 \pm 0.011}$	$\mathbf{0.054 \pm 0.003}$	$\mathbf{0.583 \pm 0.021}$	$\mathbf{0.637 \pm 0.022}$	$\mathbf{0.638 \pm 0.023}$	–
CARLA $A=5$	DESIRE	2.622 ± 0.030	2.621 ± 0.045	2.422 ± 0.048	2.710 ± 0.066	2.969 ± 0.057	2.391 ± 0.049
	DESIRE-plan	2.329 ± 0.038	0.194 ± 0.004	2.239 ± 0.057	3.119 ± 0.098	3.332 ± 0.090	2.758 ± 0.083
	ESP	0.718 ± 0.012	0.340 ± 0.011	0.759 ± 0.024	0.809 ± 0.025	0.851 ± 0.023	0.828 ± 0.024
	PRECOG	$\mathbf{0.640 \pm 0.011}$	$\mathbf{0.066 \pm 0.003}$	$\mathbf{0.741 \pm 0.024}$	$\mathbf{0.790 \pm 0.024}$	$\mathbf{0.804 \pm 0.022}$	$\mathbf{0.801 \pm 0.024}$
nuScenes $A=2$	DESIRE	3.307 ± 0.093	3.002 ± 0.088	3.613 ± 0.140	–	–	–
	DESIRE-plan	4.528 ± 0.151	0.456 ± 0.015	8.600 ± 0.298	–	–	–
	ESP	1.094 ± 0.053	0.955 ± 0.057	1.233 ± 0.078	–	–	–
	PRECOG	$\mathbf{0.514 \pm 0.037}$	$\mathbf{0.158 \pm 0.016}$	$\mathbf{0.871 \pm 0.070}$	–	–	–
nuScenes $A=3$	DESIRE	4.840 ± 0.135	3.931 ± 0.127	4.984 ± 0.207	5.606 ± 0.234	–	–
	DESIRE-plan	5.887 ± 0.187	0.409 ± 0.015	7.731 ± 0.337	9.521 ± 0.399	–	–
	ESP	1.511 ± 0.077	1.128 ± 0.061	1.543 ± 0.118	1.862 ± 0.147	–	–
	PRECOG	$\mathbf{1.016 \pm 0.062}$	$\mathbf{0.121 \pm 0.005}$	$\mathbf{1.320 \pm 0.105}$	$\mathbf{1.606 \pm 0.122}$	–	–
nuScenes $A=4$	DESIRE	5.771 ± 0.151	4.195 ± 0.159	5.854 ± 0.243	6.138 ± 0.280	6.896 ± 0.324	–
	DESIRE-plan	5.045 ± 0.158	0.471 ± 0.019	5.567 ± 0.245	5.492 ± 0.257	8.652 ± 0.407	–
	ESP	2.200 ± 0.090	1.604 ± 0.099	1.940 ± 0.123	2.405 ± 0.149	2.851 ± 0.213	–
	PRECOG	$\mathbf{1.755 \pm 0.083}$	$\mathbf{0.133 \pm 0.006}$	$\mathbf{1.804 \pm 0.126}$	$\mathbf{2.319 \pm 0.141}$	$\mathbf{2.764 \pm 0.231}$	–
nuScenes $A=5$	DESIRE	6.830 ± 0.204	4.999 ± 0.219	6.415 ± 0.294	7.027 ± 0.360	7.418 ± 0.324	8.290 ± 0.532
	DESIRE-plan	6.562 ± 0.207	2.261 ± 0.100	6.644 ± 0.314	6.184 ± 0.325	9.203 ± 0.448	8.520 ± 0.514
	ESP	2.921 ± 0.175	1.861 ± 0.109	2.369 ± 0.188	2.812 ± 0.188	3.201 ± 0.254	4.363 ± 0.652
	PRECOG	$\mathbf{2.508 \pm 0.152}$	$\mathbf{0.149 \pm 0.021}$	$\mathbf{2.324 \pm 0.187}$	$\mathbf{2.654 \pm 0.190}$	$\mathbf{3.157 \pm 0.273}$	$\mathbf{4.254 \pm 0.586}$

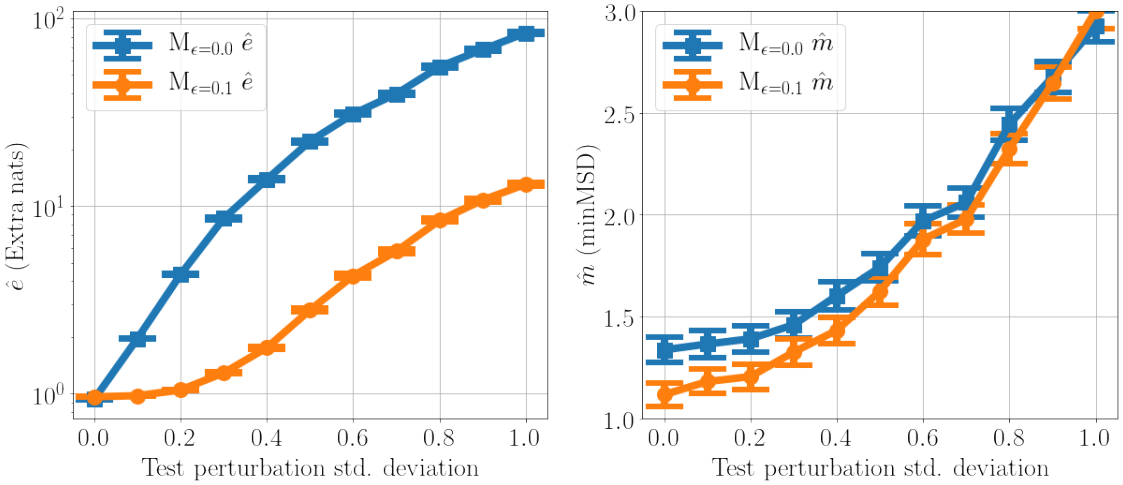


Figure 7.14: Evaluating the effects of noisy localization on nuScenes $A = 2$.

7.5 Conclusions

We presented a multi-agent forecasting method, ESP, that outperforms state-of-the-art multi-agent forecasting methods on real (nuScenes) and simulated (CARLA) driving data. We also developeded a novel algorithm, PRECOG, to condition forecasts on agent goals. We showed conditional forecasts improve joint-agent and per-agent predictions, compared to unconditional forecasts used in prior work. Conditional forecasting can be used for planning, which we demonstrated with a novel multi-agent imitative planning objective. Future directions include conditional forecasting w.r.t.multiple agent goals, useful for multi-AV coordination via communicated intent.

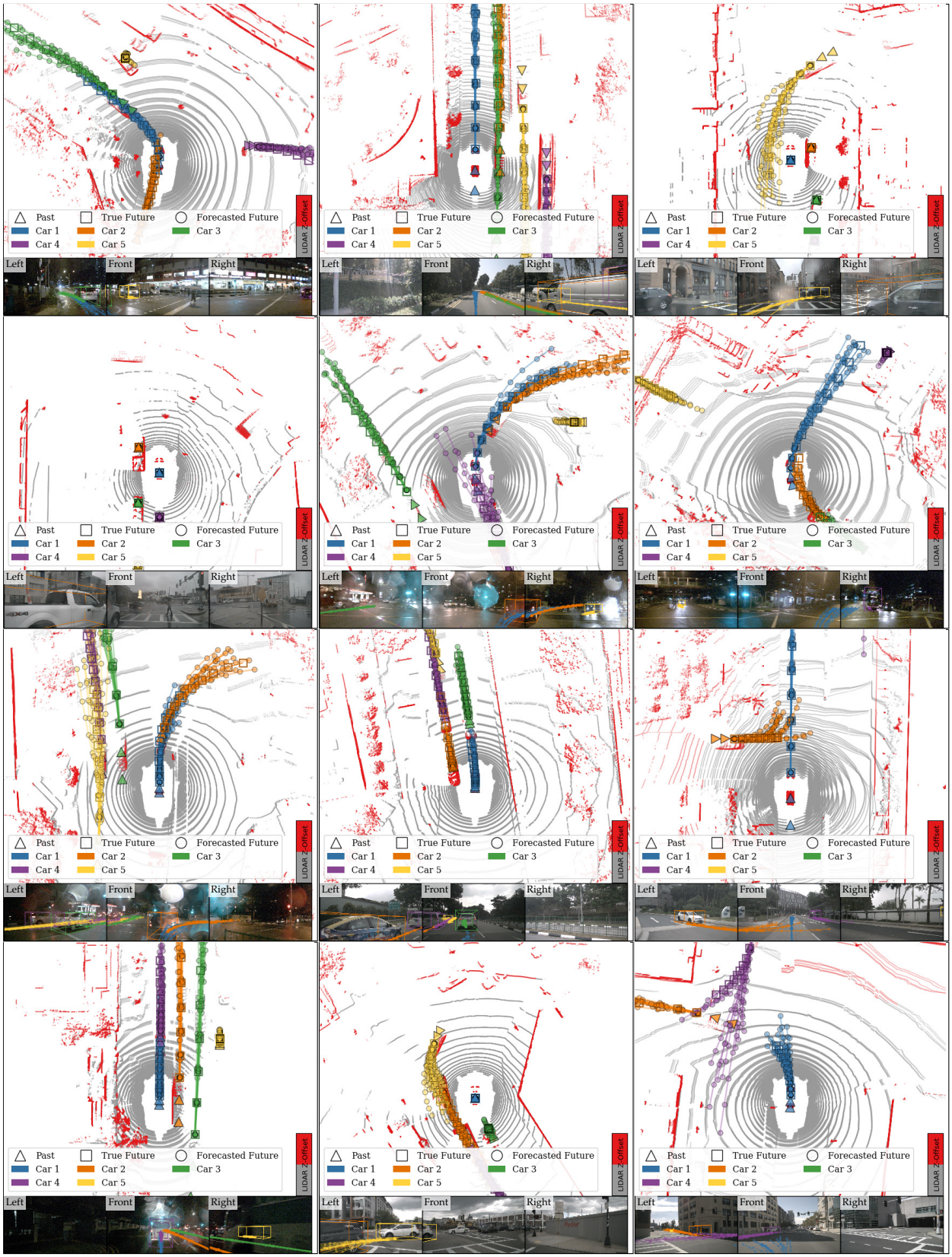


Figure 7.17: Example forecasting results on held-out nuScenes data with our learned ESP model. In each scene, 12 joint samples are shown, and LIDAR colors are discretized to near-ground and above-ground

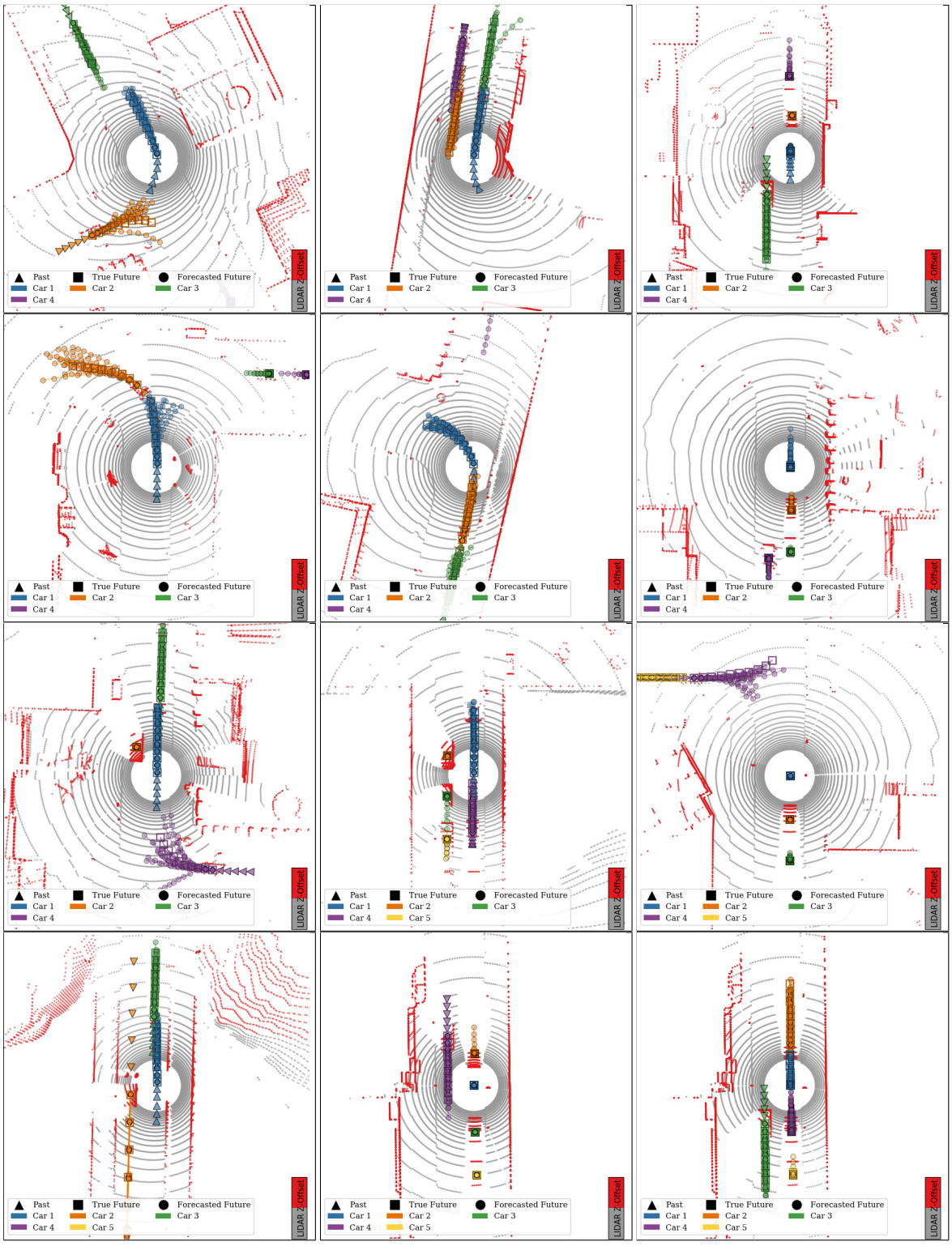
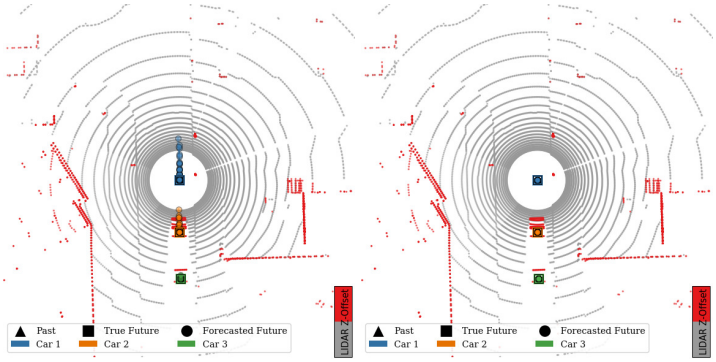
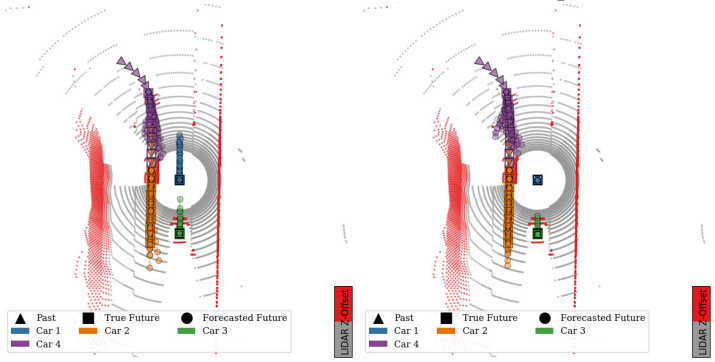


Figure 7.18: Examples of multi-agent forecasting with our learned ESP model. In each scene, 12 joint samples are shown, and LIDAR colors are discretized to near-ground and above-ground.



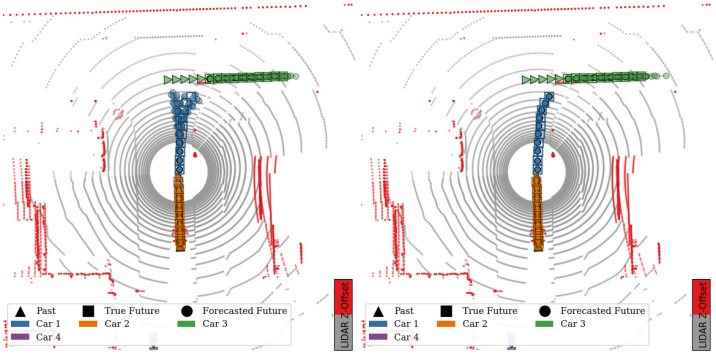
(a) Scene 1, forecasted

(b) Scene 1, planned



(c) Scene 2, forecasted

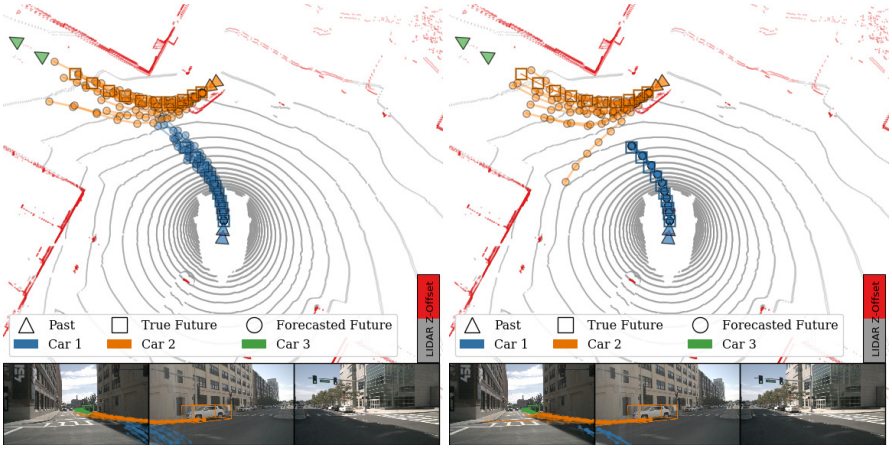
(d) Scene 2, planned



(e) Scene 3, forecasted

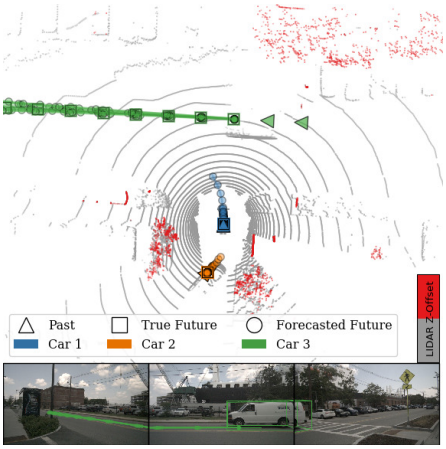
(f) Scene 3, planned

Figure 7.19: Additional examples of *planned* multi-agent forecasting (PRECOG) with our learned model in CARLA. By using our planning approach and conditioning the robot on its true final position, our predictions for the robot become more accurate, and often our predictions of the other agent become more accurate.

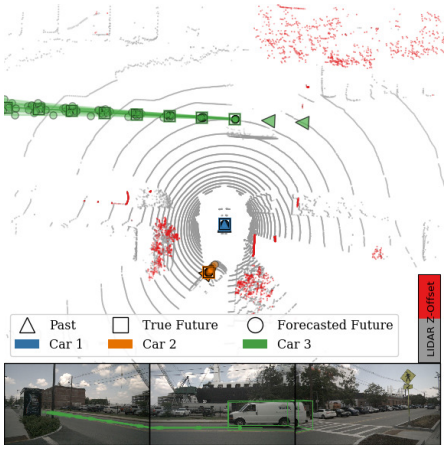


(a) Scene 1, forecasted

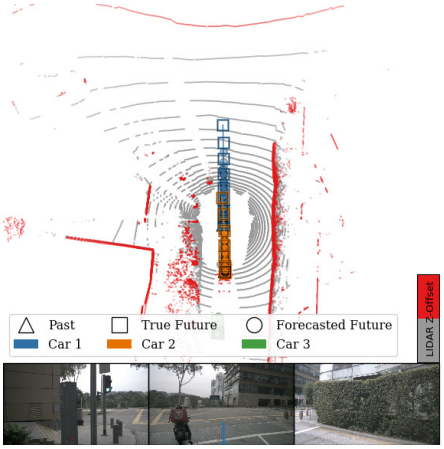
(b) Scene 1, planned



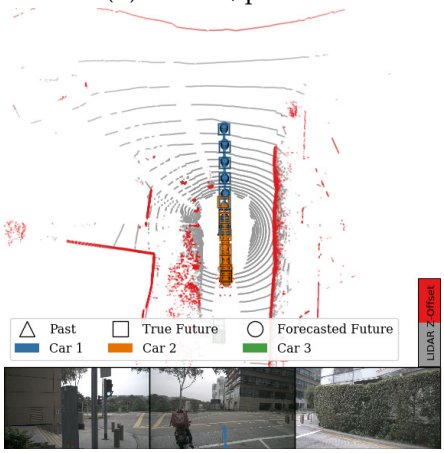
(c) Scene 2, forecasted



(d) Scene 2, planned

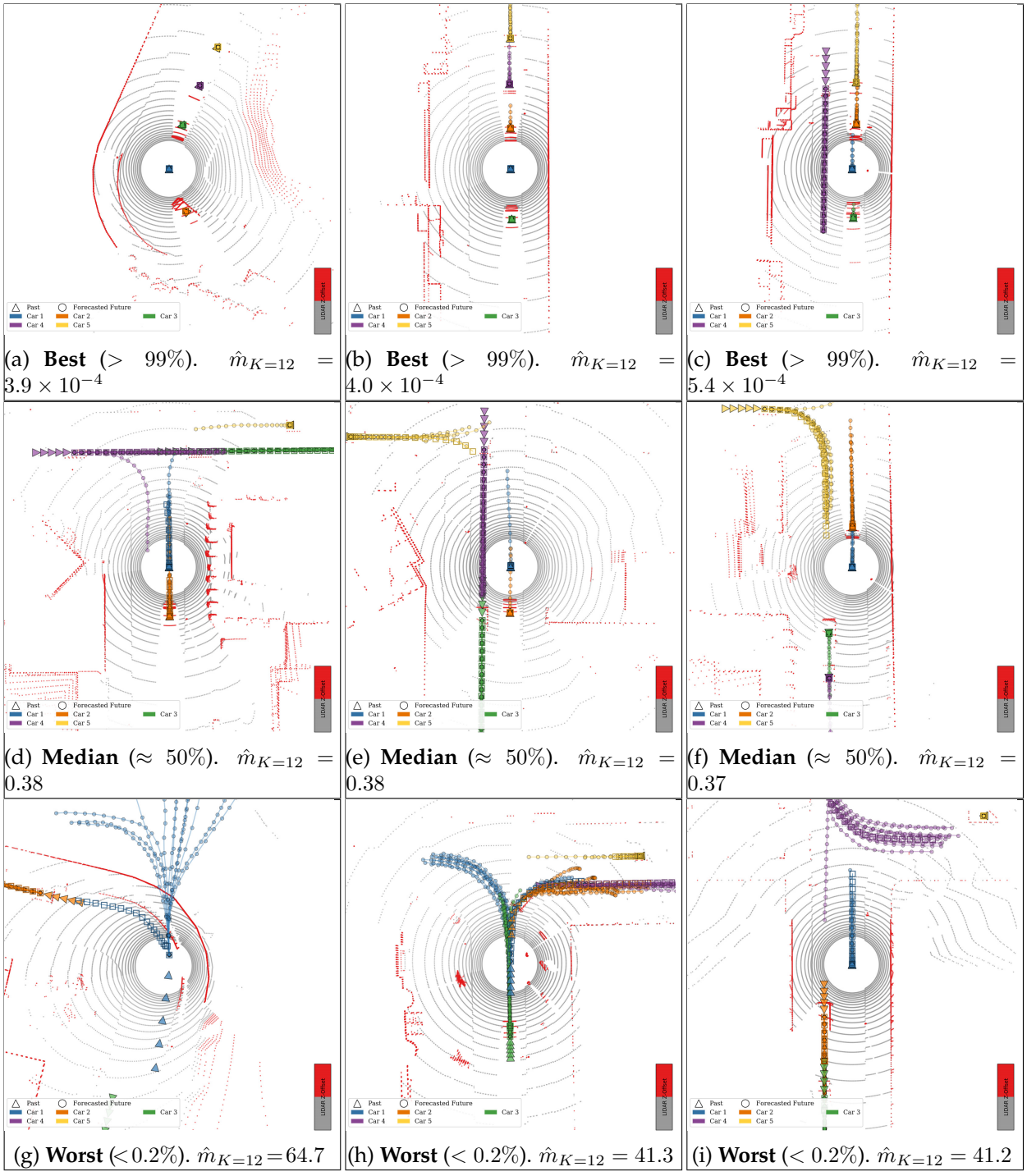


(e) Scene 3, forecasted



(f) Scene 3, planned

Figure 7.20: Additional examples of *planned* multi-agent forecasting (PRECOG) with our learned model in nuScenes. By using our planning approach and conditioning the robot on its true final position, our predictions for the robot become more accurate, and often our predictions of the other agent become more accurate.



(a) **Best** ($> 99\%$). $\hat{m}_{K=12} = 3.9 \times 10^{-4}$

(b) **Best** ($> 99\%$). $\hat{m}_{K=12} = 4.0 \times 10^{-4}$

(c) **Best** ($> 99\%$). $\hat{m}_{K=12} = 5.4 \times 10^{-4}$

(d) **Median** ($\approx 50\%$). $\hat{m}_{K=12} = 0.38$

(e) **Median** ($\approx 50\%$). $\hat{m}_{K=12} = 0.38$

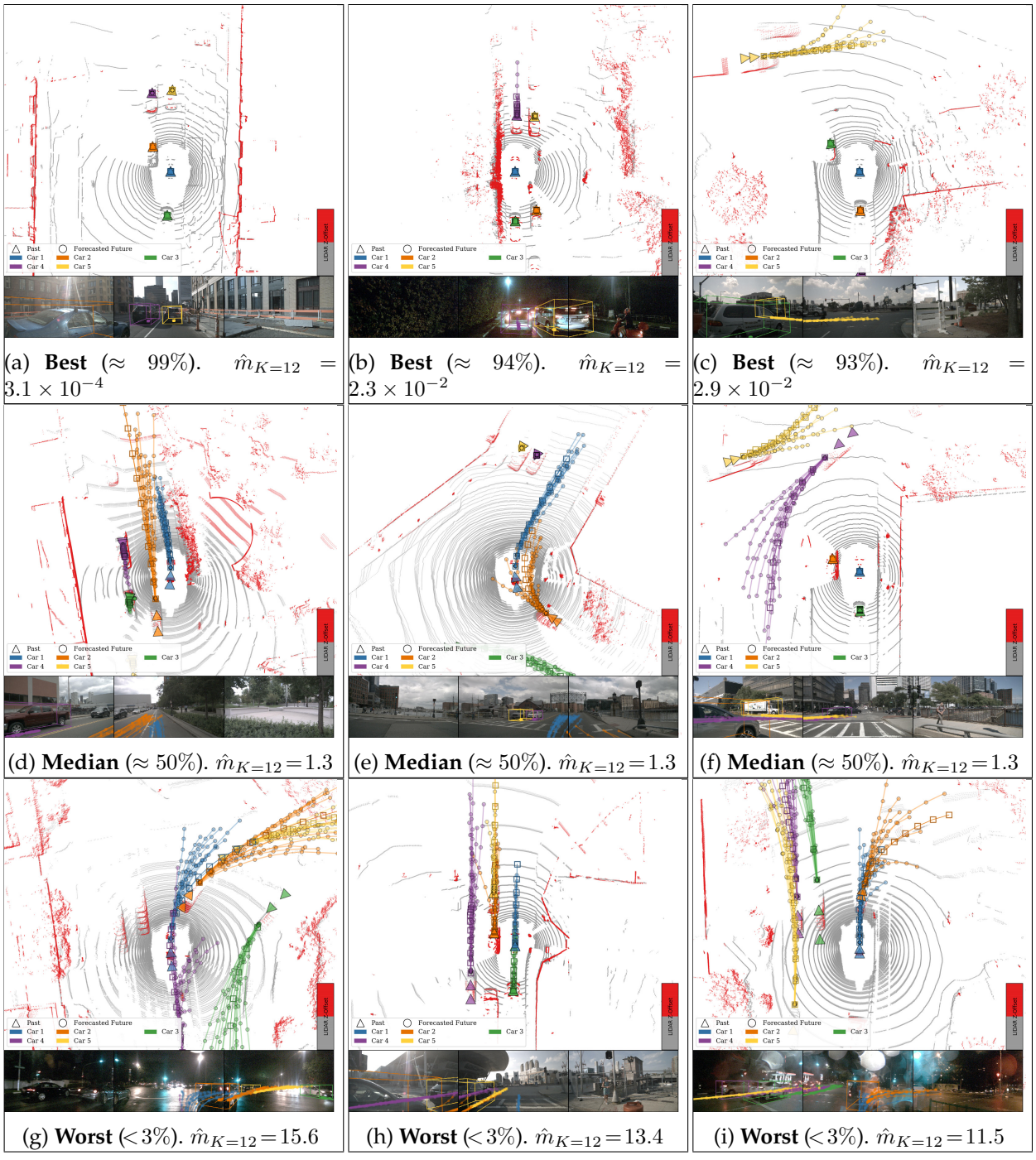
(f) **Median** ($\approx 50\%$). $\hat{m}_{K=12} = 0.37$

(g) **Worst** ($< 0.2\%$). $\hat{m}_{K=12} = 64.7$

(h) **Worst** ($< 0.2\%$). $\hat{m}_{K=12} = 41.3$

(i) **Worst** ($< 0.2\%$). $\hat{m}_{K=12} = 41.2$

Figure 7.21: Various qualities (*Row 1:* $\approx 100\%$, *Row 2:* $\approx 50\%$, and *Row 3:* $\approx 0\%$) of qualitative results of the ESP flex. count model on `Town01 Test`, $A = 5$, $T = 40$ at 10Hz (4 seconds of future), ordered by $\hat{m}_{K=12}$. Recall since $\hat{m}$ is a *joint-agent* statistic, per-agent trajectory sample coverage is insufficient for a good $\hat{m}$ score. Also, recall $\hat{m}$ measures the error of the *closest* joint trajectory to the true future, as opposed to the error of *all* joint trajectories, which is key to its property of not penalizing otherwise-plausible trajectories.



(a) **Best** ($\approx$ 99%). $\hat{m}_{K=12} = 3.1 \times 10^{-4}$

(b) **Best** ($\approx$ 94%). $\hat{m}_{K=12} = 2.3 \times 10^{-2}$

(c) **Best** ($\approx$ 93%). $\hat{m}_{K=12} = 2.9 \times 10^{-2}$

(d) **Median** ($\approx 50\%$). $\hat{m}_{K=12} = 1.3$

(e) **Median** ($\approx 50\%$). $\hat{m}_{K=12} = 1.3$

(f) **Median** ($\approx 50\%$). $\hat{m}_{K=12} = 1.3$

(g) **Worst** ($< 3\%$). $\hat{m}_{K=12} = 15.6$

(h) **Worst** ($< 3\%$). $\hat{m}_{K=12} = 13.4$

(i) **Worst** ($< 3\%$). $\hat{m}_{K=12} = 11.5$

Figure 7.22: Various qualities (*Row 1:* $\approx 100\%$, *Row 2:* $\approx 50\%$, and *Row 3:* $\approx 0\%$) of qualitative results of the ESP flex. count model on nuScenes Test, $A = 5$, $T = 20$ at 5Hz (4 seconds of future), ordered by $\hat{m}_{K=12}$. Recall since $\hat{m}$ is a *joint-agent* statistic, per-agent trajectory sample coverage is insufficient for a good $\hat{m}$ score. Also, recall $\hat{m}$ measures the error of the *closest* joint trajectory to the true future, as opposed to the error of *all* joint trajectories, which is key to its property of not penalizing otherwise-plausible trajectories.

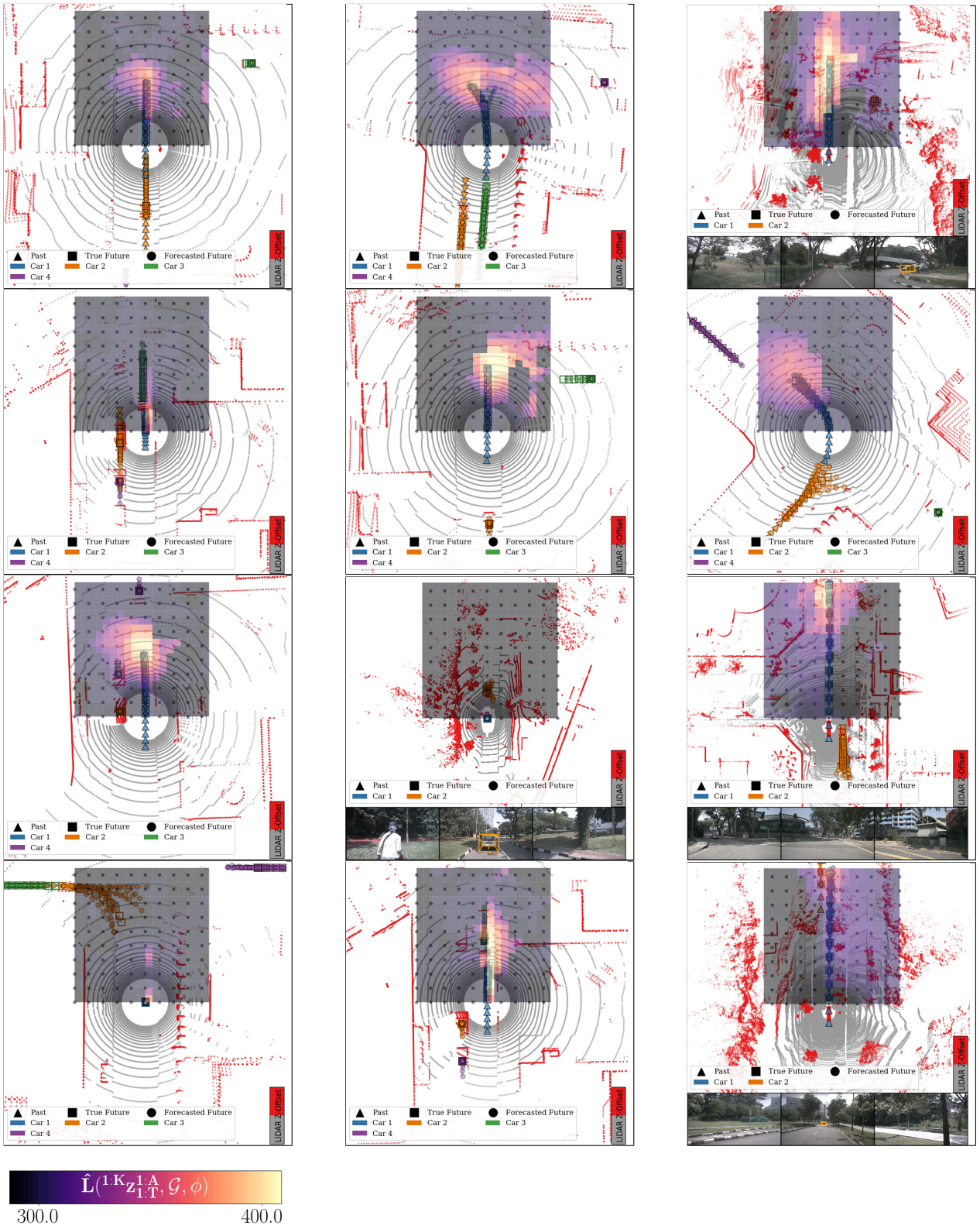


Figure 7.23: Plotting the planning criterion, $\hat{L}$, after planning to various positions (small circular dots in each plot) input to Alg. 9, with values interpolated between each position, in CARLA. The planning criterion input corresponds to a spatio-temporal goal at $T = 20$ in the future (4 seconds). The planning criterion prefers locations within its lane, unless it is uncertain about the possibility of turning. When the vehicle was stationary in the past, the planning criterion is highest at positions at or close in front of the vehicle.

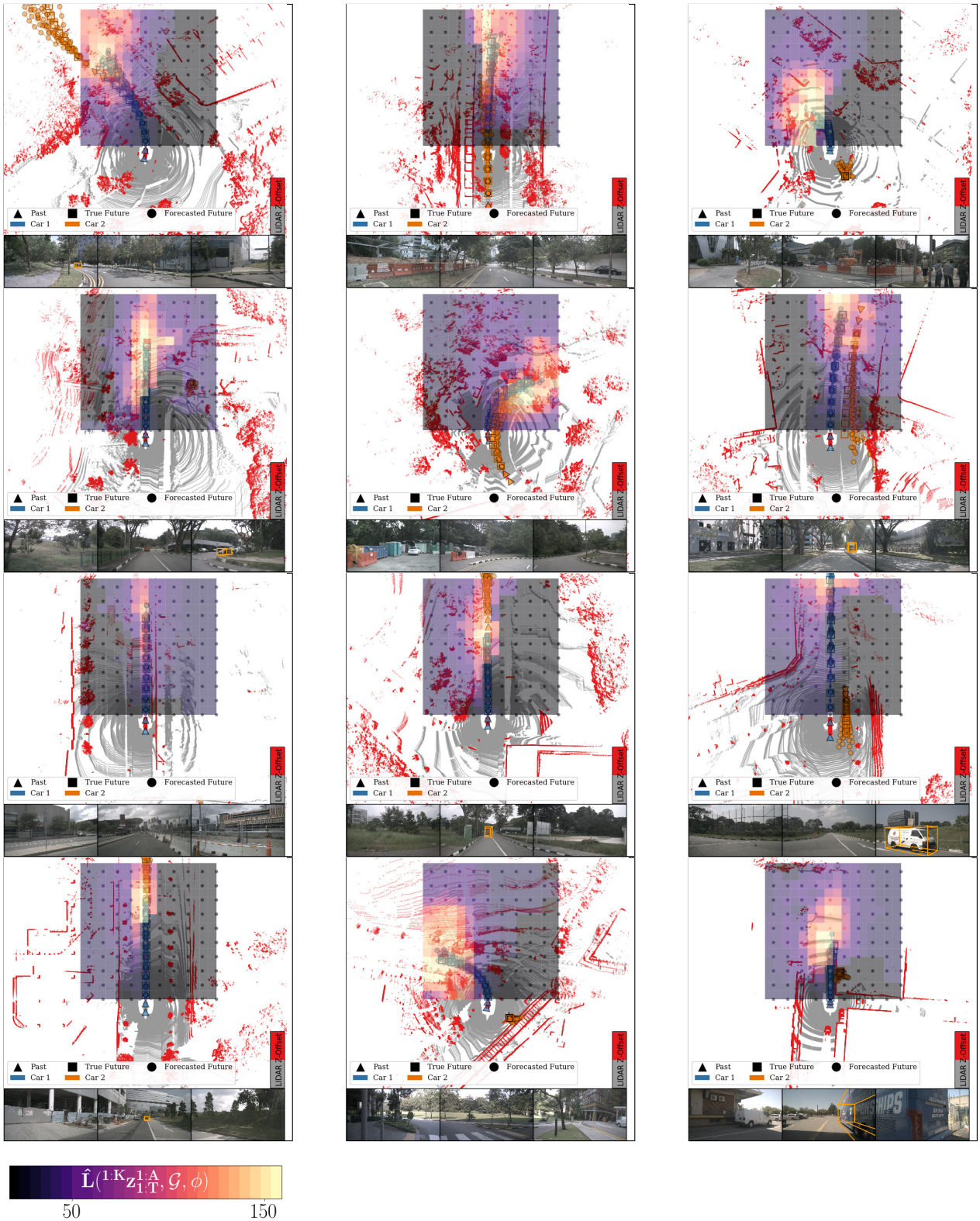


Figure 7.24: Plotting the planning criterion, $\hat{L}$, after planning to various positions (small circular dots in each plot) input to Alg. 9, with values interpolated between each position, in nuScenes. The planning criterion input corresponds to a spatio-temporal goal at $T = 20$ in the future (4 seconds). The planning criterion prefers locations within its lane, unless it is uncertain about the possibility of turning. When the vehicle was stationary in the past, the planning criterion is highest at positions at or close in front of the vehicle.

Part III

Conclusion and Future Work

7.6 Conclusion and Future Work

Mirroring the structure of this thesis, we restate our contributions and speculate on promising directions of future work.

7.6.1 Part I: Activity and Motion Forecasting from High-Dimensional Observations

We contributed several approaches to perform activity and motion forecasting from high-dimensional observations.

7.6.1.1 Chapter 2: Forecasting Singular Actions with Action Maps

The first approach constructs "Action Maps" by observing behavior from a first-person camera and relating the actions to the visual characteristics of the surrounding environment. Through this relation, a matrix factorization approach is applied to generate functionality predictions: estimations of what *could* be done in various locations. There are two main limitations of this approach: the categories of activities are limited to those that can be reliably detected, and the visual representations of scenes were limited to objects and rooms that could be reliably classified.

7.6.1.2 Chapter 3: Forecasting Action Trajectories with Online Inverse Reinforcement Learning

Our second approach was specifically focused on predicting the future goals of a user: what high-level activity *is likely* to be done , which roughly estimates a person's intention across arbitrary horizons of space and time. Our algorithm used a first-person camera to observe behaviors, which is used to inform the states and actions (including localization, activity, and object detection) of an Inverse Reinforcement Learning approach. By estimating the reward function via the Maximum Entropy Inverse Reinforcement Learning approach, the approach recovers a mechanism for straightforwardly forecasting a distribution over possible future goals. The approach learns online from behavior by discovering the *goals*, or terminal states, of a person's behavior. We employed several heuristics to obtain these goal states (scene-specific goals and stop-based goals), however, these mechanisms are insufficient – they neither achieve perfect recall nor precision for the true goals. An interesting avenue of future work would be to improve the goal discovery component of the algorithm. Furthermore, although the algorithm enjoys some benefits by learning from scratch for each user, learning for a new users could be accelerated by employing a reward prior estimation across the set of existing users.

7.6.1.3 Chapter 4: Forecasting Motion Trajectories with Deep Reversible Generative Models

we contributed several approaches to forecast precise motion of agents. Our first approach, R2P2, estimates a conditional density function over future trajectories, which receives rich observations in the form of LIDAR. We extended recent techniques to learn likelihood-based deep generative models. We used the observation of the duality of trade-offs between forward and reverse KL, and employ a symmetric reverse KL (less the model entropy) as a training objective. Our model is both *reparameterized*, enabling efficient optimization of reverse KL, and can *compute the density function exactly*, enabling efficient optimization of the forward KL. In R2P2, we first learned a simple secondary model to use to penalize the full model in the reverse KL term. In C3PO, we focused on continually learning this secondary model in tandem with learning our model, which is similar to

other adversarial generative modeling approaches, but retains the desirable properties of efficient exact model density evaluation. One downside to this approach was that it reasoned about other agents *implicitly* through its observations of them in the LIDAR. A direction of future work that we pursued (PRECOG) was to explicitly reason about all agents present in a scene. Furthermore, one of the challenges in motion forecasting is proper evaluation of models. In R2P2, we pointed out the serious deficiency in the most popular metric for motion forecasting (mean-squared error from a batch of model samples to the single sample of the future). However, alternative metrics still have some flaws, and thus the proper evaluation of generative forecasting models is still an open area of research. We speculate two promising paradigms: a *metrics suite* that attempts to accommodate the drawbacks of each metric, and *combining budget-based metrics*. The latter specifies "given K samples, produce a diversity of plausible outcomes", where the definitions of plausible and diverse are problem-specific. When models are evaluated across many budgets $K_1 \dots K_N$, the evaluation provides information about the models at various regimes.

7.6.2 Part II: Jointly Forecasting and Controlling from High-Dimensional Observations

We contributed several approaches that marry aspects of forecasting with the task of control, with the goal of joining them together into a single framework.

7.6.2.1 Chapter 5: Forecasting Observations as Auxiliary Supervision for Implicitly-Planned Control

Our first contribution towards joint forecasting in control leverages theoretical insights about 1) the difficulty of training the internal state of RNNs 2) the learnability of dynamic systems through Predictive State Representations. Our resulting Predictive-State Decoders technique is straightforward in practice: augment the RNN training loss with a term that penalizes the internal RNN state for failing to be predictive of sufficient statistics of future observations. We found that this simple, well-motivated approach yielded performance gains across control tasks in filtering, Imitation Learning, and Reinforcement Learning. The main drawback of this approach is that, in the case of rich observations, it is very difficult to design featurization functions for the sufficient statistics of this future observation distribution. An alternative would be to learn this representation beforehand (e.g. using a VAE), and then predict the latent state of the VAE in order to generate observations.

7.6.2.2 Chapter 6: Forecasting Motion Trajectories for Explicitly-Planned Control

Leveraging the idea of directly estimating the (undirected) distribution over future expert trajectories from R2P2, we consider *directing* this distribution with test-time distributions that encourage trajectories to achieve abstract goals. These goals can be designed to cause trajectories to arrive near a specific position at a specific point in time, avoid areas or obstacles, et cetera. The key idea is that the resulting planned trajectories strike a *balance* between being likely under the expert prior of future trajectories, as well as achieving the specified goals at test-time. Using this simple paradigm, we demonstrated state-of-the-art performance on the CARLA static and dynamic navigation benchmarks, in comparison to behavior-cloning approaches and a dynamics model-based planning approach that did not internalize a prior. We showed that even when the goals are poorly specified, such as on the wrong side of the road or very noisy, the prior helps the model stay on the correct side of the road, and the goal likelihood still provides sufficient information for

directing the vehicle. When multiple goals are provided, the prior effectively enables the planned trajectories to "choose" the goal it prefers the most, which corresponds to the goal it estimates that the expert would prefer the most. One drawback of this approach is that if the goal likelihoods are improperly specified, it requires some interaction (on-policy) data from the learned model in order to tune the goal likelihoods. This tuning procedure took only a few steps for each likelihood, but it would be ideal to formalize it as a hyperparameter or model-selection procedure. Additionally, we demonstrated this procedure in simulation – however, a more impressive display would be to demonstrate it on an embodied robot, e.g. an autonomous car. Finally, this approach could be extended to control *multiple* agents in the presence of other uncontrolled agents to perform cooperative control tasks. Drawing again on the example domain of autonomous driving, this approach could be used to coordinate a fleet of robot vehicles in the presence of other human-driven vehicles. Theoretically, it is straightforward to do so using the multi-agent planning procedure developed in PRECOG.

7.6.2.3 Chapter 7: Forecasting Multi-Agent Motion Trajectories for Explicitly-Planned Interactions

Finally, we contributed a multi-agent forecasting approach, ESP, that uses rich observations of one of the agent's surrounding environments. By extending R2P2 to a multi-agent model and leveraging its existing factorization over time, we recover the ability to perform additional probabilistic inference queries that involve reasoning about the per-agent, per-time step behaviors. We showed how we can use this property to condition on additional knowledge of one of the agent's *goal*, and found that conditioning on this knowledge *improves predictions of the other agent's behaviors* (**PRE**diction **C**onditioned **O**n **G**oals). The ESP model (unconditional forecasting) achieved state-of-the-art performance on real data. The PRECOG approach is applicable across a spectrum of forecasting problems spanning "no prior information of intentions" to "full prior information of intentions". It introduces the notion of reasoning about a controlled agent's intentions affect the behaviors of other uncontrolled agents. We intend to further explore this approach's connections to causal inference.

7.6.3 Publication List

Publications Covered

1. Rhinehart, Kitani, CVPR 2016 [169] (Chapter 2)
2. Rhinehart, Kitani, ICCV 2017, [170] (Chapter 3)
3. Rhinehart, Kitani, TPAMI 2018 [168] (Chapter 3)
4. Rhinehart et al., ECCV 2018 [171] (Chapter 4)
5. Rhinehart et al., OpenReview 2018 [174] (Chapter 4)
6. Rhinehart*, Venkatraman*, et al. NeuRIPS 2017 [228] (Chapter 5)
7. Rhinehart et al., arXiv 2018 [176] (Chapter 6)
8. Rhinehart et al., ICCV 2019 [175] (Chapter 7)

Publications Uncovered

9. Rhinehart et al., ICRA 2015 [177]

10. Ashok et al., ICLR 2018 [13]
11. Pan et al., AAMAS 2018 [146]
12. Shankar et al., CORL 2018 [199]
13. Sharma et al., ICLR 2019 [200]
14. Guan et al., arXiv 2019 [73]

7.6.4 Concluding Summary

We motivated the problem of computational forecasting through the dual motivation of its connection to core goals in the natural sciences and the human ability to forecast. The brunt of our focus was to reason about what agents could do in order to plan controls. This focus on future agent behavior allowed us to tightly couple and jointly perform forecasting and control. In pursuit of this goal, our contributions widen the focus of Computer Vision and Reinforcement Learning. We hope to see Computer Vision approaches (1) pay more attention to the forecasting problem and (2) consider *how the forecasting model will be used to evoke useful behavior*. We hope to see more Reinforcement and Imitation Learning approaches (1) pay more attention to the problem of *explicitly forecasting the future* and (2) design approaches to *incorporate forecasting as a fundamental component of intelligent systems that evoke useful behavior*. The better we can forecast intelligent behavior, the better we can understand and produce it. Improving forecasting provides a clear path to building systems of greater behavioral capabilities.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, 2016, pp. 265–283.

[2] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, "Discriminative training of kalman filters.," in *Robotics: Science and Systems (RSS)*, 2005.

[3] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.

[4] ——, "Exploration and apprenticeship learning in reinforcement learning," in *ICML*, ACM, 2005, pp. 1–8.

[5] ——, "Learning first-order markov models for control," in *NIPS*, 2005, pp. 1–8.

[6] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," in *Readings in Computer Vision*, Elsevier, 1987, pp. 522–533.

[7] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 5074–5082.

[8] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[9] ——, "Social lstm: Human trajectory prediction in crowded spaces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[10] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," *arXiv preprint arXiv:1703.00443*, 2017.

[11] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and Z. Kolter., "Differentiable MPC for end-to-end planning and control.," 2018.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[13] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2n learning: Network to network compression via policy gradient reinforcement learning," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B1hcZZ-AW.

[14] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[15] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proceedings of International Conference on Robotics and Automation*, IEEE, vol. 4, 1997, pp. 3557–3564.

[16] L. Ballan, F. Castaldo, A. Alahi, F. Palmieri, and S. Savarese, "Knowledge transfer for scene-specific motion prediction," in *European Conference on Computer Vision*, Springer, 2016, pp. 697–713.

[17] N. Baram, O. Anschel, I. Caspi, and S. Mannor, "End-to-end differentiable adversarial imitation learning," in *International Conference on Machine Learning*, 2017, pp. 390–399.

[18] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[19] S. Barratt and R. Sharma, "A Note on the Inception Score," *ArXiv e-prints*, Jan. 2018. arXiv: 1801.01973 [stat.ML].

[20] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," *arXiv preprint arXiv:1705.02503*, 2017.

[21] D. Belanger and A. McCallum, "Structured prediction energy networks," in *International Conference on Machine Learning*, 2016, pp. 983–992.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[23] A. Bhattacharyya, M. Malinowski, B. Schiele, and M. Fritz, "Long-term image boundary prediction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI, 2017.

[24] A. Bhattacharyya, B. Schiele, and M. Fritz, "Accurate and diverse sampling of sequences based on a "best of many" sample objective," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8485–8493.

[25] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[26] B. Boots, "Spectral approaches to learning predictive representations," PhD thesis, Carnegie Mellon University, Dec. 2012.

[27] B. Boots, A. Gretton, and G. J. Gordon, "Hilbert space embeddings of predictive state representations," in *UAI-2013*, 2013.

[28] B. Boots, S. M. Siddiqi, and G. J. Gordon, "Closing the learning-planning loop with predictive state representations," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 954–966, 2011.

[29] R. J. Bowden and D. A. Turkington, *Instrumental variables*, 8. Cambridge University Press, 1990.

[30] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[32] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[33] F. Cakir and S. Sclaroff, "Adaptive hashing for fast similarity search," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[34] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Mark Siskind, and S. Wang, "Recognize human activities from partially observed videos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013.

[35] R. Caruana, "Multitask learning," in *Learning to learn*, Springer, 1998, pp. 95–133.

[36] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun, "Learning deep structured models," in *International Conference on Machine Learning*, 2015, pp. 1785–1794.

[37] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[38] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, 2015, pp. 2980–2988.

[39] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, pp. 746–752, 1998.

[40] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *ICML*, New York, NY, USA: ACM, 2008, pp. 144–151.

[41] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–9.

[42] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," *arXiv preprint arXiv:1904.08980*, 2019.

[43] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, "Analytic moment-based gaussian process filtering," in *International Conference on Machine Learning*, ACM, 2009, pp. 225–232.

[44] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning (ICML)*, 2011, pp. 465–472.

[45] V. Delaitre, D. F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. A. Efros, "Scene semantics from long-term observation of people," in *Computer Vision–ECCV 2012*, Springer, 2012, pp. 284–298.

[46] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs," *arXiv preprint arXiv:1805.05499*, 2018.

[47] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," *arXiv preprint arXiv:1605.08803*, 2016.

[48] ——, "Density estimation using Real NVP," *arXiv preprint arXiv:1605.08803*, 2016.

[49] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *arXiv preprint arXiv:1611.01779*, 2016.

[50] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on Robot Learning (CoRL)*, 2017, pp. 1–16.

[51] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[52] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters, "Probabilistic model-based imitation learning," *Adaptive Behavior*, vol. 21, no. 5, pp. 388–403, 2013.

[53] A. Fathi, A. Farhadi, and J. M. Rehg, "Understanding egocentric activities," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, pp. 407–414.

[54] P. Felsen, P. Lucey, and S. Ganguly, "Where will they go? Predicting fine-grained adversarial multi-agent motion using conditional variational autoencoders," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 732–747.

[55] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection," *Neural networks*, vol. 108, pp. 466–478, 2018.

[56] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural computation*, vol. 21, no. 3, pp. 793–830, 2009.

[57] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 2786–2793.

[58] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," *arXiv preprint arXiv:1810.05766*, 2018.

[59] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic, "People watching: Human actions as a cue for single-view geometry," in *Proc. 12th European Conference on Computer Vision*, 2012.

[60] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo," in *CVPR*, 2010.

[61] Y. Gal, "Uncertainty in deep learning," PhD thesis, University of Cambridge, 2016.

[62] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015. [Online]. Available: http://www.roboticsproceedings.org/rss11/p43.html.

[63] J. Gall, A. Fossati, and L. Van Gool, "Functional categorization of objects using real-time markerless motion capture," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 1969–1976.

[64] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *International Conference on Machine Learning*, 2015, pp. 881–889.

[65] Z. Ghahramani and S. T. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," pp. 431–437, 1999.

[66] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[67] W. Grathwohl, R. T. Chen, J. Betterncourt, I. Sutskever, and D. Duvenaud, "FFJORD: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[68] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks.," in *ICML*, vol. 14, 2014, pp. 1764–1772.

[69] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, 2016.

[70] A. Grover, M. Dhar, and S. Ermon, "Flow-gan: Bridging implicit and prescribed learning in generative models," *arXiv preprint arXiv:1705.08868*, 2017.

[71] ——, "Flow-gan: Combining maximum likelihood and adversarial learning in generative models," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17409.

[72] Q. Gu, J. Zhou, and C. H. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs.," in *SDM*, SIAM, 2010, pp. 199–210.

[73] J. Guan, Y. Yuan, K. M. Kitani, and N. Rhinehart, "Generative hybrid representations for activity forecasting with no-regret learning," *arXiv preprint arXiv:1904.06250*, 2019.

[74] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5769–5779.

[75] A. Gupta, T. Chen, F. Chen, D. Kimber, and L. S. Davis, "Context and observation driven latent variable model for human pose estimation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, IEEE, 2008, pp. 1–8.

[76] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert, "From 3d scene geometry to human workspace," in *Computer Vision and Pattern Recognition(CVPR)*, 2011.

[77] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[78] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," *NIPS*, 2016.

[79] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[80] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.

[81] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[82] A. Hefny, C. Downey, and G. J. Gordon, "Supervised learning for dynamical system learning," in *NIPS*, 2015.

[83] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[84] M. Hoai and F. De la Torre, "Max-margin early event detectors," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 191–202, 2014.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] D. Hsu, S. M. Kakade, and T. Zhang, "A spectral algorithm for learning hidden markov models," in *COLT*, 2009.

[87] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing, "On unifying deep generative models," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rylSzl-R-.

[88] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" *arXiv preprint arXiv:1511.05101*, 2015.

[89] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 695–709, 2005.

[90] F. Itakura, "Analysis synthesis telephony based on the maximum likelihood method," in *The 6th international congress on acoustics, 1968*, 1968, pp. 280–292.

[91] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, "Generative modeling of multimodal multi-human behavior," *arXiv preprint arXiv:1803.02015*, 2018.

[92] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *CoRR*, vol. abs/1611.05397, 2016. [Online]. Available: http://arxiv.org/abs/1611.05397.

[93] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena, "Recurrent neural networks for driver activity anticipation via sensory-fusion architecture," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 3118–3125.

[94] Y. Jiang, H. Koppula, and A. Saxena, "Hallucinated humans as the hidden context for labeling 3d scenes," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, IEEE, 2013, pp. 2993–3000.

[95] S. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 2, pp. 1531–1538, 2002.

[96] S. M. Kakade *et al.*, "On the sample complexity of reinforcement learning," PhD thesis, 2003.

[97] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 4743–4751. [Online]. Available: http://papers.nips.cc/paper/6581-improved-variational-inference-with-inverse-autoregressive-flow.pdf.

[98] ——, "Improved variational inference with inverse autoregressive flow," in *Advances in Neural Information Processing Systems*, 2016, pp. 4743–4751.

[99] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[100] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 236–10 245.

[101] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision*, Springer, 2012, pp. 201–214.

[102] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "GP-UKF: Unscented kalman filters with Gaussian process prediction and observation models," pp. 1901–1907, 2007.

[103] I. Kokkinos, "Ubernet: Training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," *CoRR*, vol. abs/1609.02132, 2016.

[104] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 14–29, 2016.

[105] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[106] L. Kuvayev and R. S. Sutton, "Model-based reinforcement learning with an approximate, learned model," in *Yale Workshop on Adaptive and Learning Systems*, 1996, pp. 101–105.

[107] T. Lan, T.-C. Chen, and S. Savarese, "A hierarchical representation for future action prediction," in *European Conference on Computer Vision*, Springer, 2014, pp. 689–704.

[108] J. Langford, R. Salakhutdinov, and T. Zhang, "Learning nonlinear dynamic models," in *ICML*, ACM, 2009, pp. 593–600.

[109] S. M. LaValle, "Planning algorithms," in, Cambridge University Press, 2006, ch. 14, pp. 802–805.

[110] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *International Conference on Machine Learning*, 2017, pp. 1995–2003.

[111] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[112] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.

[113] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," 2017.

[114] N. Lee and K. M. Kitani, "Predicting wide receiver trajectories in american football," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2016, pp. 1–9.

[115] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization.," in *CVPR*, vol. 2, 2012, p. 7.

[116] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[117] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[118] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1644–1657, 2014.

[119] Y. Li, Z. Ye, and J. M. Rehg, "Delving into egocentric actions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[120] Y. Li, Z. Ye, and J. M. Rehg, "Delving into egocentric actions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 287–295.

[121] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3815–3825.

[122] Z. Li, T. Motoyoshi, K. Sasaki, T. Ogata, and S. Sugano, "Rethinking self-driving: Multi-task knowledge for better generalization and accident explanation ability," *arXiv preprint arXiv:1809.11100*, 2018.

[123] X. Liang, T. Wang, L. Yang, and E. Xing, "CIRL: Controllable imitative reinforcement learning for vision-based self-driving," *arXiv preprint arXiv:1807.03776*, 2018.

[124] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Exploring context with deep structured models for semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[125] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," in *Advances In Neural Information Processing Systems*, 2016, pp. 2378–2386.

[126] M. Ma, H. Fan, and K. M. Kitani, "Going deeper into first-person activity recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1894–1903.

[127] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani, "A game-theoretic approach to multi-pedestrian activity forecasting," *arXiv preprint arXiv:1604.01431*, 2016.

[128] ——, "Forecasting interactive dynamics of pedestrians with fictitious play," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, IEEE, 2017, pp. 4636–4644.

[129] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller, "Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning," International Joint Conferences on Artificial Intelligence (IJCAI), 2017.

[130] R. J. McCann *et al.*, "Existence and uniqueness of monotone measure-preserving maps," 1995.

[131] F. S. Melo and M. Veloso, "Decentralized MDPs with sparse interactions," *Artificial Intelligence*, vol. 175, no. 11, pp. 1757–1789, 2011.

[132] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "DropoutDAgger: A Bayesian approach to safe imitation learning," *arXiv preprint arXiv:1709.06166*, 2017.

[133] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *CoRR*, vol. abs/1611.02163, 2016. arXiv: 1611.02163. [Online]. Available: http://arxiv.org/abs/1611.02163.

[134] D. J. Moore, I. Essa, M. H. Hayes III, *et al.*, "Exploiting human actions and object context for recognition tasks," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, IEEE, vol. 1, 1999, pp. 80–86.

[135] R. Mur-Artal, J. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[136] I. Najfeld and T. F. Havel, "Derivatives of the matrix exponential and their computation," *Advances in applied mathematics*, vol. 16, no. 3, pp. 321–375, 1995.

[137] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, 2000, pp. 663–670.

[138] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Estimating divergence functionals and the likelihood ratio by convex risk minimization," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5847–5861, 2010.

[139] S. Nowozin, B. Cseke, and R. Tomioka, "F-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[140] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[141] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, *et al.*, "Parallel WaveNet: Fast high-fidelity speech synthesis," *arXiv preprint arXiv:1711.10433*, 2017.

[142] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[143] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[144] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[145] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[146] X. Pan, E. Ohn-Bar, N. Rhinehart, Y. Xu, Y. Shen, and K. M. Kitani, "Human-interactive subgoal supervision for efficient inverse reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

[147] H. S. Park, J.-J. Hwang, Y. Niu, and J. Shi, "Egocentric future localization.," in *CVPR*, vol. 2, 2016, p. 4.

[148] S. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture," *arXiv preprint arXiv:1802.06338*, 2018.

[149] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[150] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks.," *ICML*, vol. 28, pp. 1310–1318, 2013.

[151] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[152] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, 2009, pp. 261–268.

[153] P. Peursum, G. West, and S. Venkatesh, "Combining image regions and human activity for indirect object recognition in indoor wide-angle views," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, IEEE, vol. 1, 2005, pp. 82–89.

[154] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *arXiv preprint arXiv:1703.02702*, 2017.

[155] H. Pirsiavash and D. Ramanan, "Detecting activities of daily living in first-person camera views," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 2847–2854.

[156] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[157] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 1989, pp. 305–313.

[158] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[159] L. Ralaivola and F. D'Alche-Buc, "Dynamical modeling with kernels for nonlinear time series prediction," *NIPS*, 2004.

[160] P. Ramachandran and G. Varoquaux, "Mayavi: 3D Visualization of Scientific Data," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 40–51, 2011, ISSN: 1521-9615.

[161] R. Ranftl and T. Pock, "A deep variational model for image segmentation," in *German Conference on Pattern Recognition*, Springer, 2014, pp. 107–118.

[162] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *ICLR*, 2016.

[163] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 729–736.

[164] B. Recht, *The policy of truth*, http://www.argmin.net/2018/02/20/reinforce/, Blog, 2018.

[165] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "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.

[166] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.

[167] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[168] N. Rhinehart and K. Kitani, "First-person activity forecasting from video with online inverse reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[169] N. Rhinehart and K. M. Kitani, "Learning action maps of large environments via first-person vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 580–588.

[170] N. Rhinehart and K. M. Kitani, "First-person activity forecasting with online inverse reinforcement learning," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[171] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *European Conference on Computer Vision (ECCV)*, Sep. 2018.

[172] ——, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *The European Conference on Computer Vision (ECCV)*, Sep. 2018.

[173] ——, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *The European Conference on Computer Vision (ECCV)*, Sep. 2018.

[174] N. Rhinehart, A. Liu, K. Sohn, and P. Vernaza, *Learning gibbs-regularized GANs with variational discriminator reparameterization*, 2019. [Online]. Available: https://openreview.net/forum?id=BJlpCsC5Km.

[175] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," *arXiv preprint arXiv:1905.01296*, 2019.

[176] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," *arXiv preprint arXiv:1810.06544*, 2018.

[177] N. Rhinehart, J. Zhou, M. Hebert, and J. A. Bagnell, "Visual chunking: A list prediction framework for region-based object detection," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 5448–5454.

[178] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*, Springer, 2016, pp. 549–565.

[179] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, pp. 832–837, 1956.

[180] S. Ross, G. Gordon, and D. 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*, 2011, pp. 627–635.

[181] ——, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[182] S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell, "Learning message-passing inference machines for structured prediction," in *CVPR*, IEEE, 2011.

[183] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *arXiv preprint arXiv:1905.06113*, 2019.

[184] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[185] M. S. Ryoo, T. J. Fuchs, L. Xia, J. K. Aggarwal, and L. H. Matthies, "Robot-centric activity prediction from first-person videos: What will they do to me'," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI 2015, Portland, OR, USA, March 2-5, 2015*, 2015, pp. 295–302. DOI: 10.1145/2696454.2696462. [Online]. Available: http://doi.acm.org/10.1145/2696454.2696462.

[186] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011.

[187] M. S. Ryoo and L. Matthies, "First-person activity recognition: What are they doing to me?" In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2730–2737.

[188] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, "Trajnet: Towards a benchmark for human trajectory prediction," *arXiv preprint*, 2018.

[189] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 791–798.

[190] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[191] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," *arXiv preprint arXiv:1806.06498*, 2018.

[192] M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, and M. Nießner, "Scenegrok: Inferring action maps in 3d environments," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, 2014.

[193] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–9.

[194] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," in *Advances in Neural Information Processing Systems*, 2015, pp. 3528–3536.

[195] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897.

[196] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, "Deep network flow for multi-object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6951–6960.

[197] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.

[198] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[199] T. Shankar, N. Rhinehart, K. Muelling, and K. M. Kitani, "Learning neural parsers with deterministic differentiable imitation learning," in *Proceedings of The 2nd Conference on Robot Learning*, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., ser. Proceedings of Machine Learning Research, vol. 87, PMLR, 29–31 Oct 2018, pp. 592–604. [Online]. Available: http://proceedings.mlr.press/v87/shankar18a.html.

[200] M. Sharma, A. Sharma, N. Rhinehart, and K. M. Kitani, "Directed-info GAIL: Learning hierarchical policies from unsegmented demonstrations using directed information," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BJeWUs05KQ.

[201] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[202] ——, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[203] S. Singh, M. R. James, and M. R. Rudary, "Predictive state representations: A new theory for modeling dynamical systems," in *UAI*, 2004.

[204] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 3483–3491. [Online]. Available: http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative-models.pdf.

[205] L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. J. Smola, "Hilbert space embeddings of hidden markov models," in *ICML*, 2010, pp. 991–998.

[206] H. Soo Park, J.-J. Hwang, Y. Niu, and J. Shi, "Egocentric future localization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[207] E. H. Spriggs, F. De la Torre Frade, and M. Hebert, "Temporal segmentation and activity classification from first-person sensing," in *IEEE Workshop on Egocentric Vision, CVPR 2009*, Jun. 2009.

[208] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks: Learning generalizable representations for visuomotor control," in *International Conference on Machine Learning*, 2018, pp. 4739–4748.

[209] S. Su, J. P. Hong, J. Shi, and H. S. Park, "Social behavior prediction from first person videos," *arXiv preprint arXiv:1611.09464*, 2016.

[210] C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy, "Stochastic prediction of multi-agent interactions from partial observations," *arXiv preprint arXiv:1902.09641*, 2019.

[211] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A fast integrated planning and control framework for autonomous driving via imitation learning," *arXiv preprint arXiv:1707.02515*, 2017.

[212] W. Sun, R. Capobianco, G. J. Gordon, J. A. Bagnell, and B. Boots, "Learning to smooth with bidirectional predictive state inference machines," in *Proceedings of The International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

[213] W. Sun, N. Jiang, A. Krishnamurthy, A. Agarwal, and J. Langford, "Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches," in *Proceedings of the Thirty-Second Conference on Learning Theory*, A. Beygelzimer and D. Hsu, Eds., ser. Proceedings of Machine Learning Research, vol. 99, Phoenix, USA: PMLR, 25–28 Jun 2019, pp. 2898–2933. [Online]. Available: http://proceedings.mlr.press/v99/sun19a.html.

[214] W. Sun, A. Venkatraman, B. Boots, and J. A. Bagnell, "Learning to filter with predictive state inference machines," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1197–1205.

[215] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell, "Deeply aggrevated: Differentiable imitation learning for sequential prediction," in *ICML*, 2017.

[216] I. Sutskever, "Training recurrent neural networks," PhD thesis, University of Toronto, 2013.

[217] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

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

[219] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions," Cvpr, 2015.

[220] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2154–2162.

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

[222] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[223] E. Todorov, "Linearly-solvable Markov decision problems," in *Advances in neural information processing systems (NIPS)*, 2007, pp. 1369–1376.

[224] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural autoregressive distribution estimation," *Journal of Machine Learning Research*, vol. 17, no. 205, pp. 1–37, 2016.

[225] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory-Implementation-Applications*. Springer Science & Business Media, 2012.

[226] W. Vega-Brown and N. Roy, "Cello-em: Adaptive sensor models without ground truth," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 1907–1914.

[227] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models.," in *AAAI*, 2015, pp. 3024–3030.

[228] A. Venkatraman*, N. Rhinehart*, W. Sun, L. Pinto, M. Hebert, B. Boots, K. Kitani, and J. Bagnell, "Predictive-state decoders: Encoding the future into recurrent networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1172–1183.

[229] A. Venkatraman, W. Sun, M. Hebert, B. Boots, and J. A. (Bagnell, "Inference machines for nonparametric filter learning," in *25th International Joint Conference on Artificial Intelligence (IJCAI-16)*, Jun. 2016.

[230] L. Verlet, "Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules," *Physical review*, vol. 159, no. 1, p. 98, 1967.

[231] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, "Learning to generate long-term future via hierarchical prediction," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 3560–3569. [Online]. Available: http://proceedings.mlr.press/v70/villegas17a.html.

[232] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating visual representations from unlabeled video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 98–106.

[233] ——, "Generating videos with scene dynamics," in *Advances In Neural Information Processing Systems*, 2016, pp. 613–621.

[234] C. Vondrick and A. Torralba, "Generating the future with adversarial transformers," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 2992–3000. DOI: 10.1109/CVPR.2017.319. [Online]. Available: https://doi.org/10.1109/CVPR.2017.319.

[235] J. Walker, A. Gupta, and M. Hebert, "Patch to the future: Unsupervised visual prediction," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2014, pp. 3302–3309.

[236] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The pose knows: Video forecasting by generating pose futures," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 3352–3361.

[237] S. Wang, S. Fidler, and R. Urtasun, "Proximal deep structured models," in *Advances in Neural Information Processing Systems*, 2016, pp. 865–873.

[238] W. W.-S. Wei, *Time series analysis*. Addison-Wesley publ Reading, 1994.

[239] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[240] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*, Springer, 1992, pp. 5–32.

[241] D. Wingate and S. Singh, "Kernel predictive linear gaussian models for nonlinear stochastic dynamical systems," in *International Conference on Machine Learning*, ACM, 2006, pp. 1017–1024.

[242] C. Wu, "Towards linear-time incremental structure from motion," in *3D Vision-3DV 2013, 2013 International Conference on*, IEEE, 2013, pp. 127–134.

[243] Y. Wu, Y. Burda, R. Salakhutdinov, and R. B. Grosse, "On the quantitative analysis of decoder-based generative models," *CoRR*, vol. abs/1611.04273, 2016. arXiv: 1611.04273. [Online]. Available: http://arxiv.org/abs/1611.04273.

[244] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[245] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[246] D. Xie, S. Todorovic, and S.-C. Zhu, "Inferring "dark matter" and "dark energy" from videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2224–2231.

[247] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 2010, pp. 17–24.

[248] E. Zhan, S. Zheng, Y. Yue, and P. Lucey, "Generative multi-agent behavioral cloning," *arXiv preprint arXiv:1803.07612*, 2018.

[249] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end simulated driving," in *AAAI*, 2017, pp. 2891–2897.

[250] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," *arXiv preprint arXiv:1904.04776*, 2019.

[251] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[252] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *arXiv preprint arXiv:1412.6856*, 2014.

[253] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[254] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244. [Online]. Available: https://doi.org/10.1109/ICCV.2017.244.

[255] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," PhD thesis, Carnegie Mellon University, 2010.

[256] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 2008, pp. 1433–1438. [Online]. Available: http://www.aaai.org/Library/AAAI/2008/aaai08-227.php.

[257] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[258] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 3931–3936.