UNIVERSITY OF CALIFORNIA SAN DIEGO

Behavior Prediction of Intelligent Agents in and Around Safe Autonomous Vehicles

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical & Computer Engineering (Signal and Image Processing)

by

Nachiket Deo

Committee in charge:

Professor Mohan M. Trivedi, Chair
Professor Nikolay A. Atanasov
Professor Manmohan Chandraker
Professor Garrison W. Cottrell
Professor Bhaskar D. Rao

2022

The Dissertation of Nachiket Deo is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my parents,
for their unshakeable and sometimes inexplicable faith in me.

EPIGRAPH

"It's tough to make predictions, especially about the future"
– Yogi Berra

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

As I conclude what has been a long and rewarding journey, I'd like to express my deepest gratitude to my mentors, colleagues, family and friends. Their advice, love and support have shaped who I am, and the way I think. Quite simply, this dissertation would not exist without them.

First and foremost, I must thank my advisor, Prof. Mohan Trivedi. Throughout the past six years he has been an exceptional mentor, a role model, and my greatest advocate. He provided me the space to explore my research interests and patiently offered words of encouragement, and constructive criticism through countless failures. His enthusiasm and optimism were the perfect foil to my perfectionistic tendencies. As my colleague Akshay Rangesh aptly put - the best antidote to imposter syndrome is a brief meeting with Prof. Trivedi. I hope to emulate his deep commitment to his research statement and to the success and well-being of his students.

I would also like to thank my thesis committee, Prof. Chandraker, Prof. Rao, Prof. Cottrell and Prof. Atanasov. Though our interactions were brief, they played a huge role in improving this work. In particular, I recall Prof. Rao's advice on developing deeper insights rather than a large breadth of work, during my preliminary exam. I also recall Prof. Chandraker's advice on developing better metrics for evaluating prediction models, which helped shape some of my most recent work.

I've had the privilege to work with some incredible colleagues at LISA. In particular, I owe a great debt of gratitude to Akshay Rangesh and Kevan Yuen. Much of this work directly builds upon the testbed, models and code that they meticulously developed. My co-authors - Daniela Ridel, Kaouther Messaoud and Ross Greer. Our collaborations were some of the most fulfilling experiences during my PhD and I learned a lot from them. I would also like to thank all my other friends and colleagues from LISA: Larry Ly, Sujitha Martin, Eshed Ohn-Bar, Sourabh Vora, Ishan Gupta, Borhan Vasli, Aida Khosroshahi, Bowen Zhang, Kirill Pirozhenko, Ole Salscheider, Walter Zimmer, Eduardo Romera, Jason

Isa, Maitrayee Keskar, Anish Gopalan, Akshay Gopalkrishnan and Lulua Rakla.

I had the opportunity to spend a wonderful winter (remotely) interning with the machine learning and prediction teams at Motional. I would like to thank my hosts, Oscar Beijbom and Eric Wolff for all the brainstorming sessions, help with writing and reviving the code for the CoRL rebuttal. Hopefully we'll get to meet in person soon.

Most importantly, I'm deeply grateful to my family. Their love and support sustains all my endeavors. I'm incredibly privileged to have been born in a loving family, who valued learning and curiosity, who supported my move to a different country and my pursuit of a PhD. I'm particularly thankful to my parents, grandparents, both uncles and aunt. I am who I am because of the values you inculcated.

I'm also thankful for my friends. In particular, Sourabh, Rishi, Megha and Sahil - I could not have asked for a better set of roommates as I moved to a new country and tried to find my bearings. The early years were a struggle, but I fondly recall the company. My oldest friends, Chirag, Ashutosh, Sagar, Keyur and Iyer - our weekend calls and AoE sessions kept me sane during the pandemic.

I'd like to thank my partner Nala for being by my side through the ups and downs of the PhD experience, for grounding me, and reminding me that I'm more than just my work, and for going on so many impromptu (and some planned) adventures with me.

And last but not the least, I'd like to thank my dogs Goldie and Luna for being a boundless source of joy and love. Their happy tail wags as they unfailingly greet me at the door can melt away any amount of stress.

Publication acknowledgements:

Chapter 2, in full, is a reprint of the material as it appears in: "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," Nachiket Deo, Akshay Rangesh, and Mohan M. Trivedi, IEEE Transactions on Intelligent Vehicles 2018. The dissertation author was the primary investigator and author of this

paper.

Chapter 3, in part, is a reprint of the material as it appears in: "Convolutional Social Pooling for Vehicle Trajectory Prediction," Nachiket Deo, and Mohan M. Trivedi, CVPR Workshops 2018. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it appears in: "Trajectory Forecasts in Unknown Enviroments Conditioned on Grid-based Plans," Nachiket Deo, and Mohan M. Trivedi, arXiv:2001.00735 (2020). The dissertation author was the primary investigator and author of this paper.

Chapter 5, in part, is a reprint of the material as it appears in: "Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals," Nachiket Deo, Eric Wolff and Oscar Beijbom, CoRL 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in full, is a reprint of the material as it appears in: "Looking at the driver/rider in autonomous vehicles to predict take-over readiness," Nachiket Deo and Mohan M. Trivedi, IEEE Transactions on Intelligent Vehicles 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 7, in part, is based on "Take-over Time Prediction for Autonomous Driving in the Real-World: Robust Models, Data Augmentation, and Evaluation," Akshay Rangesh, Nachiket Deo, Ross Greer, Pujitha Gunaratne, Mohan M. Trivedi, currently submitted to IEEE Transactions on Human Machine Systems. The dissertation author was one of the primary investigators and authors of this paper.

VITA

2015	B. Tech and M. Tech in Electrical Engineering, Indian Institute of Technology Bombay
2017	M. S. in Electrical and Computer Engineering (Signal and Image Processing), University of California San Diego
2016–2022	Graduate Student Researcher, University of California San Diego
2022	Ph. D. in Electrical and Computer Engineering (Signal and Image Processing), University of California San Diego

PUBLICATIONS

N. Deo, E. Wolff, and O. Beijbom. "Multimodal trajectory prediction conditioned on lane-graph traversals." Conference on Robot Learning, PMLR, 2022.

R. Greer, J. Isa, N. Deo, A. Rangesh, and M. M. Trivedi. "On Salience-Sensitive Sign Classification in Autonomous Vehicle Path Planning: Experimental Explorations with a Novel Dataset." Winter Conference on Applications of Computer Vision Workshops, 2022.

A. Rangesh, N. Deo, R. Greer, P. Gunaratne, and M. M. Trivedi. "Autonomous Vehicles that Alert Humans to Take-Over Controls: Modeling with Real-World Data." IEEE International Conference on Intelligent Transportation Systems (ITSC), 2021.

A. Rangesh, N. Deo, R. Greer, P. Gunaratne, and M. M. Trivedi. "Predicting Take-over Time for Autonomous Driving with Real-World Data: Robust Data Augmentation, Models, and Evaluation." arXiv preprint arXiv:2107.12932 (2021).

R. Greer, N. Deo, and M. Trivedi. "Trajectory prediction in autonomous driving with a lane heading auxiliary loss." IEEE Robotics and Automation Letters, 2021.

K. Messaoud, N. Deo, M. M. Trivedi, and F. Nashashibi. "Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation." IEEE Intelligent Vehicles Symposium (IV), 2021.

D. Ridel, N. Deo, D. Wolf, and M. Trivedi. "Scene compliant trajectory forecast with agent-centric spatio-temporal grids." IEEE Robotics and Automation Letters, 2020.

N. Deo, and M. M. Trivedi. "Trajectory forecasts in unknown environments conditioned on grid-based plans." arXiv preprint arXiv:2001.00735 (2020).

N. Deo, and M. M. Trivedi. "Looking at the driver/rider in autonomous vehicles to predict take-over readiness." IEEE Transactions on Intelligent Vehicles, 2019.

D. A. Ridel, N. Deo, D. Wolf, and M. Trivedi. "Understanding pedestrian-vehicle interactions with vehicle mounted vision: An LSTM model and empirical analysis." IEEE Intelligent Vehicles Symposium (IV), 2019.

N. Deo, and M. M. Trivedi. "Scene induced multi-modal trajectory forecasting via planning." IEEE International Conference on Robotics and Automation (ICRA) Workshops, 2019.

N. Deo, N. Meoli, A. Rangesh, and M. Trivedi. "On control transitions in autonomous driving: A framework and analysis for characterizing scene complexity." IEEE/CVF International Conference on Computer Vision Workshops, 2019.

A. Rangesh, N. Deo, K. Yuen, K. Pirozhenko, P. Gunaratne, H. Toyoda, and M. M. Trivedi. "Exploring the situational awareness of humans inside autonomous vehicles." IEEE International Conference on Intelligent Transportation Systems (ITSC), 2018.

N. Deo, and M. M. Trivedi. "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms." IEEE Intelligent Vehicles Symposium (IV), 2018.

N. Deo, A. Rangesh, and M. M. Trivedi. "How would surround vehicles move? a unified framework for maneuver classification and motion prediction." IEEE Transactions on Intelligent Vehicles, 2018.

N. Deo and M. M. Trivedi. "Convolutional social pooling for vehicle trajectory prediction." IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2018.

N. Deo, and M. M. Trivedi. "Learning and predicting on-road pedestrian behavior around vehicles." IEEEInternational Conference on Intelligent Transportation Systems (ITSC), 2017.

N. Deo, A. Rangesh, and M. Trivedi. "In-vehicle hand gesture recognition using hidden markov models." IEEE International Conference on Intelligent Transportation Systems (ITSC), 2016.

ABSTRACT OF THE DISSERTATION

Behavior Prediction of Intelligent Agents in and Around Safe Autonomous Vehicles

by

Nachiket Deo

Doctor of Philosophy in Electrical & Computer Engineering (Signal and Image Processing)

University of California San Diego, 2022

Professor Mohan M. Trivedi, Chair

Autonomous vehicles operate in highly interactive environments. They share the road with humans and human driven vehicles, and they may share control with humans in the cabin. Consider scenarios such as freeway merges, unsignalized intersections or unprotected turns. These require cooperating with other on-road agents and predicting their intent and future motion. Similarly, consider scenarios in partial or conditional autonomy where control needs to be transferred to a human driver. It is critical to predict the driver's takeover readiness and reaction times to ensure safe transfer of control. The goal of this dissertation is to develop models for predicting agent behavior in and around

autonomous vehicles. We present our contributions in two parts.

In part I, we address the task of trajectory prediction of surrounding agents. We propose models that incorporate multi-agent behavior, generalize to novel scene layouts, and output a multimodal distribution over future trajectories. Concretely, our contributions are: (i) A unified framework for maneuver classification and trajectory prediction for highway traffic. (ii) An LSTM encoder-decoder with convolutional social pooling for modeling agent-agent interaction, and maneuver conditioned decoders for predicting a multimodal distribution. (iii) P2T: a model that infers goals and path preferences of agents in novel scenes using a discrete grid-based policy and predicts scene compliant trajectories. (iv) PGP: a model that predicts trajectories conditioned on paths traversed by a discrete policy in a graph representation of the scene, leading to computational efficiency and better accuracy.

In part II we focus on agents inside the autonomous vehicle. We address control transitions where control needs to be transferred from the vehicle to a human driver via a takeover request. Given the driver's gaze, hand and foot activity prior to the takeover request, we predict their readiness and reaction times during takeovers. Our contributions are: (i) a metric for the driver's takeover readiness based purely on observable cues and a model to estimate it, (ii) a model for predicting takeover time and analysis using a real-world dataset of control transitions.

Chapter 1

Introduction

Autonomous driving is one of the most exciting technological challenges of our times. Not only is it a rich source of unsolved engineering problems, but it also promises to make our roads safer and transportation more accessible. Spurred by advances in computer vision and deep learning, as well as the availability of high quality datasets, there has been incredible progress in perception for autonomous driving. Autonomous vehicles now have the ability to understand the scene around them by detecting, tracking and segmenting objects from raw sensor data, as well as the scene within them through driver gaze, hand and foot analysis using driver facing cameras.

However, perception alone isn't sufficient for autonomous vehicles to make safety critical decisions. Autonomous vehicles need to operate in highly interactive environments. They need to share the road with humans and human driven vehicles, and they might need to share control with humans in the cabin. Consider scenarios such as merging onto the highway, making lane changes, passing through unsignalized intersections, or making unprotected turns. These require cooperating with other on-road agents, reasoning about their goals and intents and predicting their future motion. Similarly, autonomous vehicles need to cooperate with humans within the cabin in cases of partial or conditional autonomy. Consider scenarios where control needs to be transferred from the autonomous vehicle to the human driver. It is critical to predict the driver's takeover readiness and

reaction times to ensure safe transfer of control. *Behavior prediction* of agents thus serves as a bridge between perception and planning.

The goal of this dissertation is to develop models for predicting agent behavior for autonomous driving. We present our contributions in two parts. Part I focuses on agents *around* the autonomous vehicle that share the road with it. Part II focuses on agents *in* the autonomous vehicle, with whom it might need to share controls. For agents around the autonomous vehicle, we address the task of trajectory prediction. Given the past locations of all agents in the scene and a structured representation of static scene elements, we predict the future locations of agents over a 5 to 10 second prediction horizon. For agents in the autonomous vehicles, we address the tasks of takeover readiness estimation, and takeover time prediction. We consider control transitions in conditionally autonomous vehicles where control needs to be transferred from the autonomous vehicle to the human driver via a takeover request. Given the driver's gaze, hand and foot activity prior to the takeover request, we predict their takeover readiness and reaction times to get their eyes on the road, hands on the wheel and foot on the pedal.

1.1 Research themes

1.1.1 Predicting Trajectories of Surrounding agents

Modeling agent-agent interaction: Drivers and pedestrians cooperate with other drivers and pedestrians while navigating through traffic. Their motion is affected by nearby agents. Consider the examples shown in Figure 1.1. The speed of a vehicle on freeways is limited by the speed of its leading vehicle. The locations and speeds of vehicles in adjacent lanes determine the feasibility of lane changes. Finally, unprotected turns are affected by cross-traffic and pedestrians crossing the intersection. Throughout part I of this dissertation, we develop models that incorporate agent-agent interaction into trajectory prediction. In chapter 2, we propose a vehicle interaction module that jointly

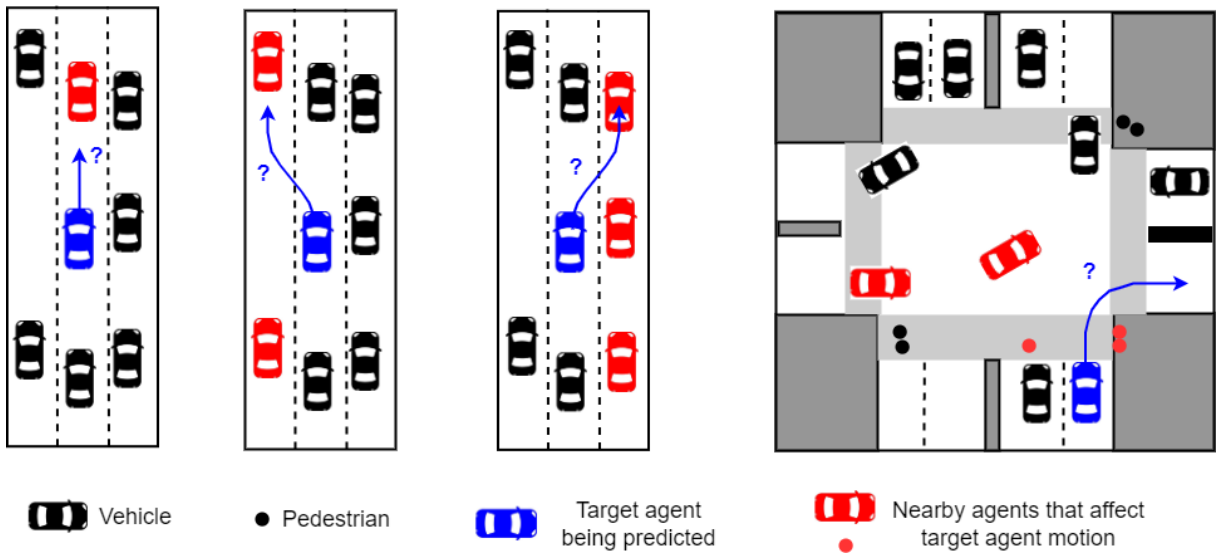


Figure 1.1. Modeling agent-agent interaction: The locations and motion of nearby agents affects the future trajectories of vehicles and pedestrians. Throughout part I, we present trajectory prediction models that incorporate agent-agent interaction.

assigns feasible maneuver classes for all vehicles around the ego-vehicle. In chapter 3, we propose *convolutional social pooling*, a more robust alternative to social pooling proposed in [3] for modeling agent-agent interaction. In chapters 4 and 5, we use attention [6,175] to selectively model the effect of agents along specific routes that the target agent may take.

Scene compliant trajectory prediction: Static scene elements such as the network of lanes and their curvature, the locations of crosswalks and sidewalks, as well as locations of buildings and parked cars contain useful cues to infer goals and path preferences of surrounding agents. Vehicles tend to move along lane centerlines, stop at stop lines and cross walks and make legal lane changes to adjacent lanes. Pedestrians tend to walk along sidewalks and crosswalks. Trajectory prediction models need to encode the static scene to accurately predict trajectories over long prediction horizons. In particular, we need models that can generalize to novel configurations of scene elements and still predict scene-compliant trajectories. In chapters 4 and 5, we model routes taken by agents in a given scene using a discrete policy exploring a grid (chapter 4) or graph (chapter 5)

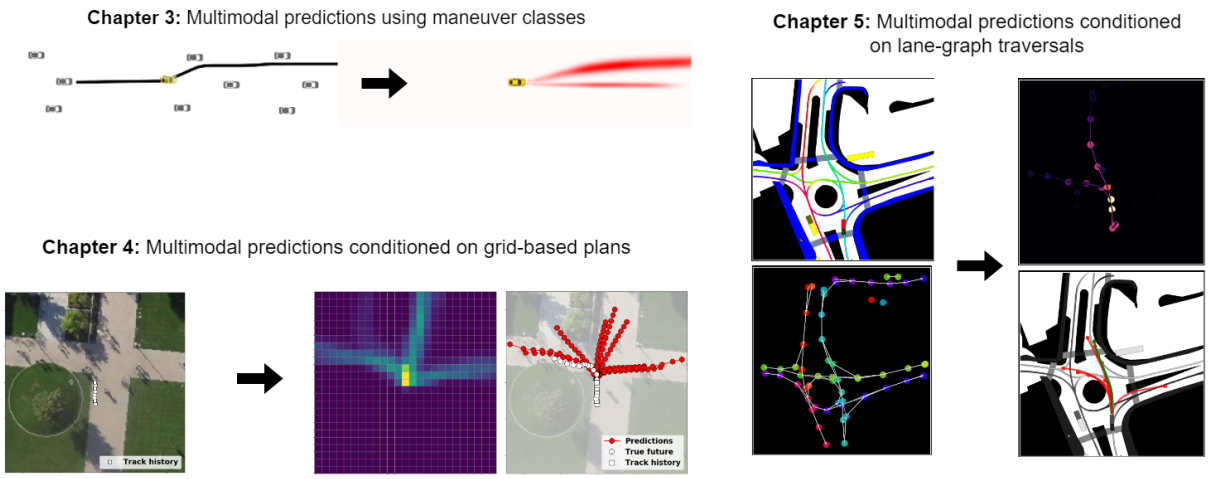


Figure 1.2. Multimodal trajectory prediction: We develop models that output a multimodal distribution over future trajectories conditioned on maneuver classes (chapter 3) or routes sampled from a discrete policy exploring a grid representation of the scene (chapter 4) or graph representation of the scene (chapter 5).

representation of the scene. This allows us to encode local scene elements at each grid cell, or node of the graph, rather than encoding the entire scene as a whole. This leads to better generalization to unknown scenes with a different configuration of scene elements, and scene compliant trajectories.

Multimodal trajectory prediction: In a given scene, drivers and pedestrians could have one of multiple plausible goals. Additionally, they could take one of multiple paths to their goals. Thus, the distribution of future trajectories of these agents is *multimodal*. The modes of the distribution correspond to different goals, routes and motion profiles along routes. Predicting a single trajectory or a unimodal distribution over trajectories can suffer from mode averaging. This can often lead to egregious predictions. For example, going straight and making a right turn can both be reasonable predictions, however their average trajectory might veer offroad. We thus develop models that output a multimodal distribution over future trajectories as shown in Figure 1.2. In chapter 3, we predict a multimodal distribution over maneuver classes for vehicle motion on freeways. In chapters 4 and 5, we learn a learn a discrete policy that explores a grid or graph representation of

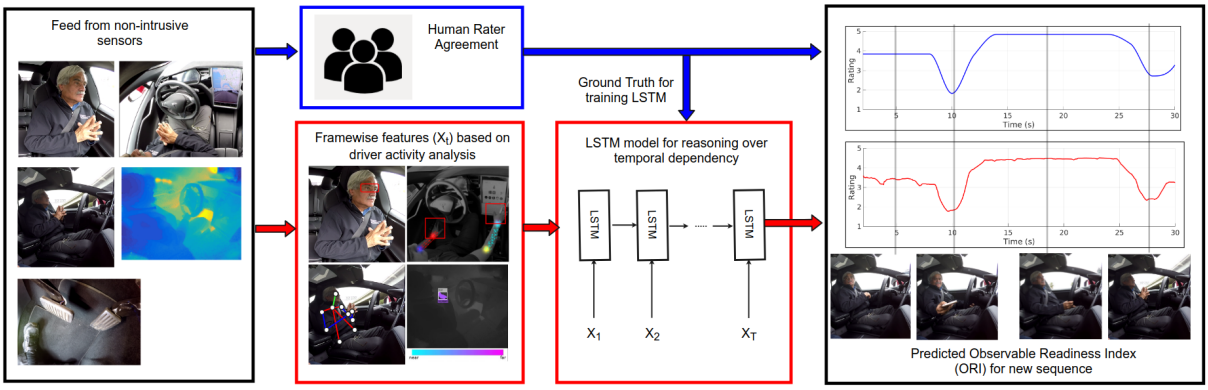


Figure 1.3. Predicting driver behavior during control transitions: We learn a holistic representation of the driver's state using CNNs for driver gaze, hand and foot activity recognition. The outputs of the CNNs are aggregated over a time window using LSTM models to encode driver activity prior to takeover requests. We then predict the takeover readiness of the driver (pictured here) based on the encoding, or predict the driver's reaction times to get their eyes on the road, hands on the wheel and foot on the pedal, if a takeover request is issued at a given instance.

the scene. The learned policy induces a multimodal distribution over plausible goals and routes to these goals. We then predict trajectories conditioned on these routes leading to a multimodal trajectory distribution.

1.1.2 Predicting Driver Behavior during Control Transitions

Holistic representation of driver activity: In conditionally autonomous vehicles, a driver can could be engaged in several non-driving tasks while the vehicle operates in autonomous mode. The nature of this non driving task can affect the driver's preparedness to takeover control when a takeover request is issued. Since we wish to develop models to predict takeover readiness and takeover time, it is crucial that we learn a useful representation of driver activity prior to the takeover request. We use multiple driver monitoring cameras and IR sensors and estimate frame by frame features using CNNs developed in prior work [22, 137, 139, 177, 184, 186]. These include the driver's gaze zones, hand locations, the hand's distance to the steering wheel, held objects, body pose keypoint locations and the foot's distance to the pedals. We then use RNNs to encode these features

over a time window. Through ablations we show the relative utility of each of these cues for predicting takeover readiness (chapter 6) and reaction times (chapter 7) of the driver.

Metrics for takeover readiness using purely observable cues: We wish to predict takeover readiness of drivers when a takeover request is issued purely using cameras. While wearable sensors such as electroencephalogram (EEG) or photoplethysmography (PPG) sensors provide the most faithful representation of the driver's state, they are too intrusive to be viable in commercial vehicles. An important question then, is whether it is even possible to define a metric for the driver's takeover readiness based on observable cues. In chapter 6, we collect subjective ratings of driver takeover readiness from multiple human observers viewing feed from driver monitoring cameras and show that there is remarkable consistency in the trend of ratings assigned by observers. We leverage this wisdom of the crowd by normalizing and averaging the ratings assigned by multiple raters, and using the resulting value as the ground truth metric for the driver's takeover readiness, termed the *observable readiness index* (ORI). We then train an LSTM model to predict this takeover readiness metric based on driver activity prior to the takeover request.

Predicting takeover time with real-world data: In chapter 7, we go a step further and predict reaction times of the driver during control transitions, termed *takeover time*. Specifically, we predict the time it takes the driver to get their eyes on the road, hands on the wheel and foot on the pedal post takeover request. While this is a more objective metric of takeover readiness, training models to predict takeover time require a dataset of takeovers performed by drivers. Chapter 7 presents a large realworld dataset of takeover events in autonomous vehicles as well as a data augmentation scheme, since such data is expensive to collect. We present LSTM models for predicting takeover time trained using the augmented dataset, and also show the utility of pretraining the model to predict ORI before takeover time prediction.

1.2 Contributions and Outline

The remainder of this dissertation is organized into 6 chapters. The key contributions of each chapter are summarized as follows.

- **A unified framework for maneuver classification and trajectory prediction:** In chapter 2, we propose a unified framework for surrounding vehicle maneuver classification and motion prediction that exploits multiple cues, namely, the estimated motion of vehicles, an understanding of typical motion patterns of freeway traffic and inter-vehicle interaction. We report our results in terms of maneuver classification accuracy and mean and median absolute error of predicted trajectories against the ground truth for real traffic data collected using vehicle mounted sensors on freeways. We perform an ablative analysis to analyze the relative importance of each cue for trajectory prediction. Additionally, we provide an analysis of execution time for the components of the framework is presented. Finally, we present multiple case studies analyzing the outputs of our model for complex traffic scenarios.

- **Convolutional Social Pooling and Maneuver-based LSTMs:** In chapter 3, we propose an LSTM encoder-decoder model that uses convolutional social pooling as an improvement to social pooling layers for robustly learning inter-dependencies in vehicle motion. Additionally, our model outputs a multimodal predictive distribution over future trajectories based on maneuver classes. We evaluate our model using the publicly available NGSIM US-101 and I-80 datasets. Our results show improvement over the state of the art in terms of RMS values of prediction error and negative log-likelihoods of true future trajectories under the model's predictive distribution. We also present qualitative analysis of the model's predicted distributions.

- **Trajectory Prediction Conditioned on Grid-based Plans:** In chapter 4, we address the problem of predicting pedestrian and vehicle trajectories in *unknown*

scenes, conditioned on their past motion and scene structure. Unlike prior approaches that directly learn one-to-many mappings from observed context to multiple future trajectories, we propose to condition trajectory forecasts on *plans* sampled from a grid based policy learned using maximum entropy inverse reinforcement learning (MaxEnt IRL). We reformulate MaxEnt IRL to allow the policy to jointly infer plausible agent goals, and paths to those goals on a coarse 2-D grid defined over the scene. We propose an attention based trajectory generator that generates continuous valued future trajectories conditioned on state sequences sampled from the MaxEnt policy. Quantitative and qualitative evaluation on the publicly available Stanford drone and NuScenes datasets shows that our model generates trajectories that are diverse, representing the multimodal predictive distribution, and precise, conforming to the underlying scene structure over long prediction horizons.

- **Trajectory Prediction Conditioned on Lane-graph Traversals:** Accurately predicting the future motion of surrounding vehicles requires reasoning about the inherent uncertainty in driving behavior. This uncertainty can be loosely decoupled into lateral (eg, keeping lane, turning) and longitudinal (eg, accelerating, braking). In chapter 5, we present a novel method that combines learned discrete policy rollouts with a trajecotry decoder focused on subsets of a lane graph representation of the scene. The policy rollouts explore different routes given current observations, ensuring that the model captures lateral variability. Longitudinal variability is captured by a latent variable model decoder that is conditioned on various subsets of the lane graph. Our model achieves state-of-the-art performance on the nuScenes prediction dataset, and qualitatively demonstrates high scene compliance. Detailed ablations highlight the importance of the policy rollouts and the decoder architecture.

- **Predicting Take-Over Readiness of Drivers using Vision Sensors:** In chapter 6, we propose a data-driven approach for estimating the driver's take-over readiness

based purely on observable cues from in-vehicle vision sensors. We present an extensive naturalistic driving dataset of drivers in a conditionally autonomous vehicle operating in freeway traffic. We collect subjective ratings for the driver's take-over readiness from multiple human observers viewing the sensor feed. Analysis of the ratings in terms of intra-class correlation coefficients (ICCs) shows a high degree of consistency in the ratings across raters. We define a metric for the driver's take-over readiness termed the 'Observable Readiness Index (ORI)' based on the ratings. Finally, we propose an LSTM model for continuous estimation of the driver's ORI based on a holistic representation of the driver's state, capturing gaze, hand, pose and foot activity.

- **Predicting Take-over Time for Autonomous Driving with Real-World Data:** In chapter 7, we present a model for predicting *takeover time* during control transitions based on driver gaze, hand and foot activity prior to takeover requests. Our model is trained and evaluated using a real-world dataset of control transitions in an autonomous vehicle with drivers engaged in various non-driving activities. Since such a dataset is costly to collect, we introduce a scheme for data augmentation. We perform ablations on driver activity cues (gaze, hand and foot), as well as model architectures, showing that a takeover time prediction model supported by augmented data can be used to produce continuous estimates of take-over times without delay, suitable for complex real-world scenarios.

Part I

Predicting Trajectories of Surrounding Agents

Chapter 2

A Unified Framework for Maneuver Recognition & Trajectory Prediction

2.1 Introduction

For successful deployment in challenging traffic scenarios, autonomous vehicles need to ensure the safety of its passengers and other occupants of the road, while navigating smoothly without disrupting traffic or causing discomfort to its passengers. Existing tactical path planning algorithms [118, 162, 172] hinge upon reliable estimation of future motion of surrounding vehicles over a prediction horizon of up to 10 s. While approaches leveraging vehicle-to-vehicle communication [61, 161, 163], offer a possible solution, these would require widespread adoption of autonomous driving technology in order to become viable. In order to safely share the road with human drivers, an autonomous vehicle needs to have the ability to predict the future motion of surrounding vehicles purely based on perception. Thus, we address the problem of surrounding vehicle motion prediction purely based on data captured using vehicle mounted sensors.

Prediction of surrounding vehicle motion is an extremely challenging problem due to a large number of factors that affect the future trajectories of vehicles. Prior works addressing the problem seem to incorporate three cues in particular: the instantaneous estimated motion of surrounding vehicles, an understanding of typical motion patterns of traffic and inter-vehicle interaction. A large body of work uses the estimated state

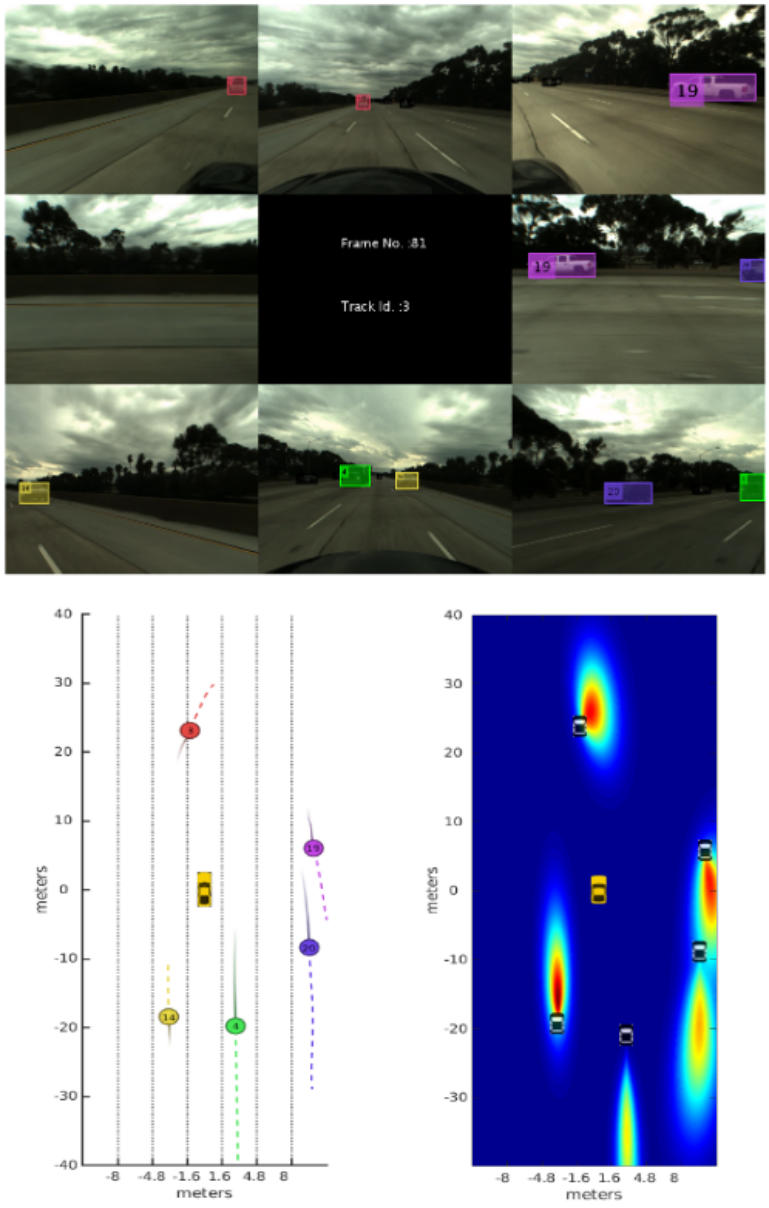


Figure 2.1. From surround perception to behavior prediction: We propose a unified model for trajectory prediction that leverages the instantaneous motion of the vehicles, the maneuver being performed by the vehicles and inter-vehicle interactions, while working purely with data captured using vehicle mounted sensors. The above figure shows the data captured by 8 surround cameras *(top)*, the track histories of surrounding vehicles, the mean predicted trajectories *(bottom left)* and a heat map of the predicted distribution in the ground plane *(bottom right)*.

of motion of surrounding vehicles along with a kinematic model to make predictions of their future trajectories [4, 8, 65, 69, 77, 131, 159, 167]. While these approaches are computationally efficient, they become less reliable for long term prediction, since they fail to model drivers as decision making entities capable of changing the motion of vehicles over long intervals. An alternative is offered by probabilistic trajectory prediction approaches [75, 88, 156, 157, 170, 180] that learn typical motion patterns of traffic from a trajectory dataset. However these approaches are prone to poorly modeling safety critical motion patterns that are under represented in the training data. Many works address these shortcomings of motion models and probabilistic models by defining a set of semantically interpretable *maneuvers* [5, 13, 45, 49, 68, 80, 104, 113, 158]. A separate motion model or probabilistic model can then be defined for each maneuver for making future predictions. Finally some works leverage inter-vehicle interaction for trajectory prediction [78, 89].

While many promising solutions have been proposed, they seem to have the following limitations. (i) Most works consider a restrictive setting such as only predicting longitudinal motion, a small subset of motion patterns, or specific cases of inter-vehicle interaction, whereas many of the biggest challenges for vehicle trajectory prediction originate from the generalized setting of simultaneous prediction of the complete motion of all vehicles in the scene. (ii) Many approaches have been evaluated using simulated data, or based on differential GPS, IMU readings of target vehicles, whereas evaluation using real traffic data captured using perceptual vehicle mounted sensors is more faithful to the setting being considered. (iii) There is a lack of a unifying approach that combines each of the three cues mentioned above and analyzes their relative importance for trajectory prediction.

2.1.1 Contributions

In this chapter, we propose a framework for holistic surrounding vehicle trajectory prediction based on three interacting modules: A hidden Markov model (HMM) based maneuver recognition module for assigning confidence values for maneuvers being performed

by surrounding vehicles, a trajectory prediction module based on the amalgamation of an interacting multiple model (IMM) based motion model and maneuver specific variational Gaussian mixture models (VGMMs), and a vehicle interaction module that considers the global context of surrounding vehicles and assigns final predictions by minimizing an energy function based on outputs of the other two modules. We work with vehicle tracks obtained using 8 vehicle mounted cameras capturing the full surround and generate the mean predicted trajectories and prediction uncertainties for all vehicles in the scene as shown in Figure 2.1. We evaluate the model using real data captured on Californian freeways. The main contributions of this chapter are:

1. A unified framework for surrounding vehicle trajectory prediction that exploits instantaneous vehicle motion, an understanding of typical motion patterns of traffic and inter-vehicle interaction.

2. Ablations for determining the relative importance of each cue in trajectory prediction.

3. Evaluation based on real traffic data captured using vehicle mounted sensors.

2.2 Related Research

2.2.1 Data-driven trajectory prediction

Data driven trajectory prediction approaches can be broadly classified into clustering based approaches and probabilistic approaches. Clustering based approaches [64, 113, 174, 174] cluster the training data to give a set of prototype trajectories. Partially observed trajectories are matched with a prototype trajectory based on distance measures such as DTW, LCSS or Hausdorff distance, and the prototype trajectory used as a model for future motion. The main drawback of clustering based approaches is the deterministic nature of the predictions. Probabilistic approaches in contrast, learn a probability distribution over motion patterns and output the conditional distribution over future motion given partial

trajectories. These have the added advantage of associating a degree of uncertainty to the future predictions. Gaussian Processes are the most popular approach for modeling trajectories [75, 88, 170]. Other approaches include [156] and [157] where the authors use Gaussian mixture regression for predicting the longitudinal and lateral motion of vehicles respectively. Of particular interest is the work by Weist *et al.* [180] who use variational Gaussian mixture models (VGMMs) to model the conditional distribution over snippets of trajectory futures given snippets of trajectory history. This approach is much leaner and computationally efficient as compared to Gaussian process regression and was shown to be effective at predicting the highly non-linear motion in turns at intersections. While Weist *et al.* use the velocity and yaw angle of the predicted vehicle obtained from its Differential GPS data, we extend this approach by learning VGMMs for freeway traffic using positions and velocities of surrounding vehicles estimated using vehicle mounted sensors, similar to our prior work on pedestrian trajectory prediction [36].

2.2.2 Maneuver-based trajectory prediction

Classification of vehicle motion into semantically interpretable maneuver classes has been extensively addressed in both advanced driver assistance systems as well as naturalistic drive studies [5, 13, 45, 49, 68, 80, 88, 104, 113, 157, 158, 170]. Most approaches involve using heuristics [68] or training classifiers such as SVMs [5,104], HMMs [13,49,88,113], LSTMs [80] and Bayesian networks [158] using motion based features such as speed, acceleration, yaw rate and other context information such as lane position, turn signals, distance from leading vehicle. The works most closely related to our approach are those that use the recognized maneuvers to make predictions of future trajectories. Houenou *et al.* [68] classify a vehicle's motion as a keep lane or lane change maneuver based on distance to nearest lane marking and predict the future trajectory by fitting a quintic polynomial between the current motion state of the vehicle and a pre-defined final motion state for each maneuver class. Schreier *et al.* [158] classify vehicle motion into one of six different

maneuver classes using a Bayesian network based on multiple motion and context based features. A class specific motion model is then defined for each maneuver to generate future trajectories. Most similar in principle to our approach are [156], [170] and [88] where separate probabilistic prediction models are trained for each maneuver class. Tran and Firl [170] define a separate Gaussian process for three maneuver classes and generate a multi-modal distribution over future trajectories using each model. However, only case based evaluation has been presented. Laugier *et al.* [88] also define separate Gaussian processes for 4 different maneuvers that are classified using a hierarchical HMM. While they report results for maneuver classification on real highway data, they evaluate trajectory prediction in the context of risk assessment simulated data. Schlechtriemen *et al.* [157] use a random forest classifier to classify maneuvers into left or right lane changes or keep lane. They use a separate Gaussian mixture regression model for making predictions of lateral movement of vehicles for each class, reporting results on real highway data. Along similar lines, but without maneuver classes, they also predict longitudinal motion for surrounding vehicles [156]. Contrary to this approach, we make predictions for the complete motion of vehicles based on maneuver class, since detection of certain maneuvers like overtakes can help predict both lateral and longitudinal motion of vehicles.

2.2.3 Interaction-aware trajectory prediction

Relatively few works address the effect of inter-vehicle interaction in trajectory prediction. Kafer *et al.* [78] jointly assign maneuver classes for two vehicles approaching an intersection using a polynomial classifier that penalizes cases which would lead to near collisions. Closer to our proposed approach, Lawitzky *et al.* [89] consider the much more complex case of assigning maneuver classes to multiple interacting vehicles in a highway setting. However, predicted trajectories and states of vehicle motion are assumed to be given, and results reported using a simulated setting. Contrarily, our evaluation considers the combined complexity due to multiple interacting vehicles as well the difficulty of

estimating their future motion. We note that inter-vehicle interaction is implicitly modeled in [157] by including relative positions and velocities of nearby vehicles as features for maneuver classification and trajectory prediction.

2.3 Overview

Figure 2.2 shows our proposed approach. We restrict our setting to purely perception based prediction of surrounding vehicle motion, without any vehicle-to-vehicle communication. The ego vehicle is equipped with 8 cameras that capture the full surround. All vehicles within 40 m of the ego vehicle in the longitudinal direction are tracked for motion analysis and prediction. While vehicle tracking is not the focus of this work, we refer readers to a multi-perspective vision based vehicle trackers described in [49, 140]. The tracked vehicle locations are then projected to the *ground plane* to generate track histories of the surrounding vehicles in the frame of reference of the ego vehicle.

The goal of our model is to estimate the future positions and the associated prediction uncertainty for all vehicles in the ego vehicle's frame of reference over the next t_f seconds, given a t_h second snippet of their most recent track histories. The model essentially consists of three interacting modules, namely the *trajectory prediction module*, the *maneuver recognition module* and the *vehicle interaction module*. The trajectory prediction module is the most crucial among the three and can function as a standalone block independent of the remaining two modules. It outputs a linear combination of the trajectories predicted by a motion model that leverages the estimated instantaneous motion of the surrounding vehicles and a probabilistic trajectory prediction model which learns motion patterns of vehicles on freeways from a freeway trajectory training set. We use constant velocity (CV), constant acceleration (CA) and constant turn rate and velocity (CTRV) models in the interacting multiple model (IMM) framework as the motion models since these capture most instances of freeway motion, especially in light traffic conditions. We use

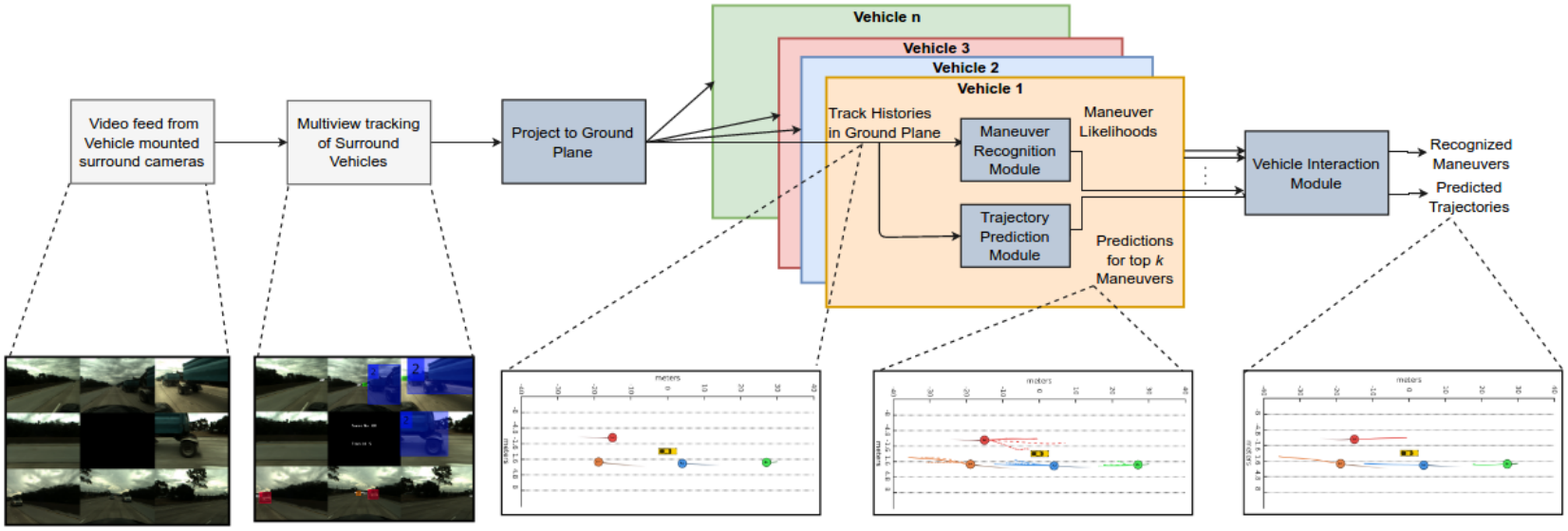


Figure 2.2. Overview of the proposed model: Track histories of all surrounding vehicles are obtained via a multi-perspective tracker and projected to the ground plane in the ego vehicle's frame of reference. The model consists of three interacting modules: The maneuver recognition module assigns confidence values to possible maneuvers being performed by each vehicle. The trajectory prediction module outputs future trajectories for each maneuver class. The vehicle interaction module assigns the true recognized maneuver for each vehicle by combining the confidence values provided by the maneuver recognition module and the feasibility of predicted trajectories given the relative configuration of all vehicles

Variational Gaussian Mixture Models (VGMM) for probabilistic trajectory prediction owing to promising results for vehicle trajectory prediction at intersections shown in [180].

The motion model becomes unreliable for long term trajectory prediction, especially in cases involving a greater degree of decision making by drivers such as overtakes, cut-ins or heavy traffic conditions. These cases are critical from a safety stand-point. However, since these are relatively rare occurrences, they tend to be poorly modeled by a monolithic probabilistic prediction model. Thus we bin surrounding vehicle motion on freeways into 10 maneuver classes, with each class capturing a distinct pattern of motion that can be useful for future prediction. The intra-maneuver variability of vehicle motion is captured through a VGMM learned for each maneuver class. The maneuver recognition module recognizes the maneuver being performed by a vehicle based on a snippet of it's most recent track history. We use hidden Markov models (HMM) for this purpose. The VGMM corresponding to the most likely maneuver can then be used for predicting the future trajectory. Thus the maneuver recognition and trajectory Prediction modules can be used in conjunction for each vehicle to make more reliable predictions.

Up to this point, our model predicts trajectories of vehicles independent of each other. However the relative configuration of all vehicles in the scene can make certain maneuvers infeasible and others more likely. This makes it a useful cue for trajectory prediction especially in heavy traffic. The vehicle interaction module (VIM) leverages this cue. The maneuver likelihoods and predicted trajectories for the K likeliest maneuvers for each vehicle being tracked are passed to the VIM. The VIM consists of a Markov random field that optimizes an energy function over the discrete space of maneuver classes for all vehicles in the scene. The energy function takes into account the confidence values for all maneuvers given by the HMM and the feasibility of the maneuvers given the relative configuration of all vehicles. Minimizing the energy function gives the recognized maneuvers and corresponding trajectory predictions for all vehicles in the scene.

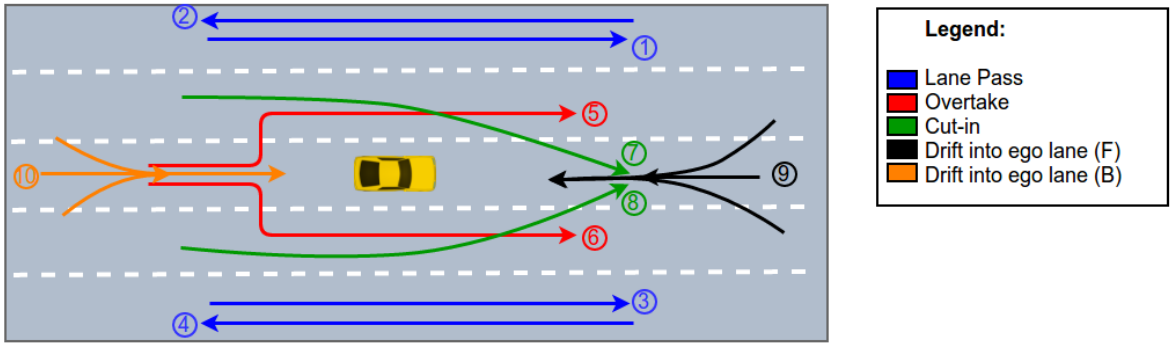


Figure 2.3. Maneuver Classes for Freeway Traffic: We bin the trajectories of surrounding vehicles in the ego-vehicle frame of reference into 10 maneuver classes: 4 lane pass maneuvers, 2 overtake maneuvers, 2 cut-in maneuvers and 2 maneuvers involving drifting into ego vehicle lane.

2.4 Maneuver Recognition Module

2.4.1 Maneuver classes

We define 10 maneuver classes for surrounding vehicle motion on freeways in the ego-vehicle's frame of reference. Figure 2.3 illustrates the maneuver classes.

1. *Lane Passes*: Lane pass maneuvers involve vehicles passing the ego vehicle without interacting with the ego vehicle lane. These constitute a majority of the surrounding vehicle motion on freeways and are relatively easy cases for trajectory prediction owing to approximately constant velocity profiles. We define 4 different lane pass maneuvers as shown in Figure 2.3

2. *Overtakes*: Overtakes start with the surrounding vehicle behind the ego vehicle in the ego lane. The surrounding vehicle changes lane and accelerates in order to pass the ego vehicle. We define 2 different overtake maneuvers, depending on which side the the surrounding vehicle overtakes.

3. *Cut-ins*: Cut-ins involve a surrounding vehicle passing the ego vehicle and entering the ego lane in front of the ego-vehicle. Cut-ins and overtakes, though relatively

rare, can be critical from a safety stand-point and also prove to be challenging cases for trajectory prediction. We define 2 different cut-ins depending on which side the surrounding vehicle cuts in from.

4. *Drift into Ego Lane:* Another important maneuver class is when a surrounding vehicle drifts into the ego vehicle lane in front or behind the ego vehicle. This is also important from a safety standpoint as it directly affects how sharply the ego vehicle can accelerate or decelerate. A separate class is defined for drifts into ego-lane in front and to the rear of the ego vehicle.

2.4.2 Hidden Markov Models

Hidden Markov models (HMMs) have previously been used for maneuver recognition [13,49,113] due to their ability to capture the spatial and temporal variability of trajectories. HMMs can be thought of as combining two stochastic models, an underlying Markov chain of states characterized by state transition probabilities and an emission probability distribution over the feature space for each state. The transition probabilities model the temporal variability of trajectories while the emission probabilities model the spatial variability, making HMMs a viable approach for maneuver recognition.

Previous works [49, 113] use HMMs for classifying maneuvers after they have been performed, where the HMM for a particular maneuver is trained using complete trajectories belonging to that maneuver class. In our case, the HMMs need to classify a maneuver based on a small t_h second snippet of the trajectory. Berndt *et al.* [13] address the problem of maneuver classification based on partially observed trajectories by using only the initial states of a trained HMM to fit the observed trajectory. However, this approach requires prior knowledge of the starting point of the maneuver. In our case, the trajectory snippet could be from any point in the maneuver, and not necessarily the start. We need the HMM to classify a maneuver based on any intermediate snippet of the trajectory. We thus

divide the trajectories in our training data into overlapping snippets of t_h seconds and train the maneuver HMMs using these snippets.

For each maneuver, we train a separate HMM with a left-right topology with only self transitions and transitions to the next state. The state emission probabilities are modeled as mixtures of Gaussians with diagonal covariances. The x and y ground plane co-ordinates and instantaneous velocity are used as features for training the HMMs. The parameters of the HMMs: the state transition probabilities and the means, variances and weights of the mixture components are estimated using the Baum-Welch algorithm [9]. For a car i, the HMM for maneuver k outputs the log likelihood:

$$\mathcal{L}_k^i = \log(P(\mathbf{x}_h^i, \mathbf{y}_h^i, \mathbf{v_x}_h^i, \mathbf{v_y}_h^i | m^i = k; \Theta_k)) \tag{2.1}$$

where $\mathbf{x}_h^i$, $\mathbf{y}_h^i$ are the x and y locations of vehicle i over the last t_h seconds and $\mathbf{v_x}_h^i$, $\mathbf{v_y}_h^i$ are the velocities along the x and y directions over the last t_h seconds. m^i is the maneuver assigned to car i and Θ_k are the parameters of the HMM for maneuver k

2.5 Trajectory Prediction Module

The trajectory prediction module predicts the future x and y locations of surrounding vehicles over a horizon of t_f seconds and assigns an uncertainty to the predicted locations in the form of a 2×2 covariance matrix. It averages the predicted future locations and covariances given by a motion model and a probabilistic trajectory prediction model. The outputs of the trajectory prediction module for a prediction instant t_{pred} are

$$x_f(t) = \frac{1}{2}\left(x_{f_{motion}}(t) + x_{f_{prob}}(t)\right) \tag{2.2}$$

$$y_f(t) = \frac{1}{2}\left(y_{f_{motion}}(t) + y_{f_{prob}}(t)\right) \tag{2.3}$$

$$\Sigma_f(t) = \frac{1}{2}\left(\Sigma_{f_{motion}}(t) + \Sigma_{f_{prob}}(t)\right) \tag{2.4}$$

where $t_{pred} \leq t \leq t_{pred} + t_f$

2.5.1 Motion Models

We use the interacting multiple model (IMM) framework for modeling vehicle motion, similar to [69, 167]. The IMM framework combines an ensemble of Bayesian filters for motion estimation and prediction by weighing the models with probability values. The probability values are estimated at each time step based on the transition probabilities of an underlying Markov model and how well each model fits the observed motion prior to that time step. We use the following motion models in our ensemble:

1. *Constant velocity (CV):* The CV model maintains an estimate of the position and velocity of the surrounding vehicles under the constraint that the vehicles move with a constant velocity. We use a Kalman filter for estimating the state and observations of the CV model. The CA model captures a majority of freeway vehicle motion.

2. *Constant acceleration (CA):* The constant acceleration model maintains estimates of the the vehicle position, velocity and acceleration under the constant acceleration assumption using a Kalman Filter. The CA model can be useful for describing freeway motion especially in dense traffic.

3. *Constant turn rate and velocity (CTRV):* The CTRV model maintains estimates of the the vehicle position, orientation and velocity magnitude under the constant yaw rate and velocity assumption. Since the state update for the CTRV model is non-linear, we use an extended Kalman filter for estimating the state and observations. The CTRV model can be useful for modeling motion during lane changes

2.5.2 Probabilistic Trajectory Prediction

We formulate probabilistic trajectory prediction as estimating the distribution:

$$P(\mathbf{v}_{\mathbf{x}f}, \mathbf{v}_{\mathbf{y}f} | \mathbf{x}_h, \mathbf{y}_h, \mathbf{v}_{\mathbf{x}h}, \mathbf{v}_{\mathbf{y}h}, m) \tag{2.5}$$

i.e. the conditional distribution of the vehicle's predicted velocities given the vehicles past positions, velocities and maneuver class. In particular, we are interested in estimating the conditional expected values $[\hat{\mathbf{v}_{\mathbf{x}f}}; \hat{\mathbf{v}_{\mathbf{y}f}}]$ and conditional covariance Σ_{v_f} of the distribution 2.5. The predicted locations and $\mathbf{x}_{f_{prob}}$, $\mathbf{y}_{f_{prob}}$ can then be obtained by taking the cumulative sum of the predicted velocities, which can be represented using an accumulator matrix $\mathbf{A}$

$$[\mathbf{x}_{f_{prob}}; \mathbf{y}_{f_{prob}}] = \mathbf{A}[\hat{\mathbf{v}_{\mathbf{x}f}}; \hat{\mathbf{v}_{\mathbf{y}f}}] \tag{2.6}$$

Similarly, the uncertainty of prediction Σ_{prob} can be obtained using the expression:

$$\Sigma_{f_{prob}} = \mathbf{A}\Sigma_{v_f}\mathbf{A}^T \tag{2.7}$$

We use the framework proposed by Weist *et al.* [180] for estimating the conditional distribution 2.5. ($\mathbf{x}_h$,$\mathbf{y}_h$,$\mathbf{v}_{\mathbf{x}h}$,$\mathbf{v}_{\mathbf{y}h}$) and ($\mathbf{v}_{\mathbf{x}f}$,$\mathbf{v}_{\mathbf{y}f}$) are represented in terms of their Chebyshev coefficients, $\mathbf{c}_h$ and $\mathbf{c}_f$. The joint distribution $P(\mathbf{c}_f, \mathbf{c}_h|m)$ for each maneuver class is estimated as the predictive distribution of a variational Gaussian mixture model (VGMM). The conditional distribution $P(\mathbf{c}_f|\mathbf{c}_h, m)$ can then be estimated in terms of the parameters of the predictive distribution. We briefly review the the expressions for $P(\mathbf{c}_f, \mathbf{c}_h|m)$ and $P(\mathbf{c}_f|\mathbf{c}_h, m)$. However, the reader is encouraged to refer to [180] for more details.

VGMMs are the Bayesian analogue to standard GMMs, where the model parameters, $\{\pi, \mu_1, \mu_2, \ldots \mu_K, \mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \ldots \mathbf{\Lambda}_K\}$ are given conjugate prior distributions. The prior over

mixture weights π is a Dirichlet distribution

$$P(\pi) = Dir(\pi|\alpha_{\mathbf{0}}) \tag{2.8}$$

The prior over each component mean μ_k and component precision $\mathbf{\Lambda}_k$ is an independent Gauss-Wishart distribution

$$P(\mu_k, \mathbf{\Lambda}_k) = \mathcal{N}(\mu_k|\mathbf{m}_{0_k}, (\beta_{0_k}\mathbf{\Lambda}_k)^{-1})\mathcal{W}(\mathbf{\Lambda}_k|\mathbf{W}_{0_k}, \nu_{0_k}) \tag{2.9}$$

The parameters of the posterior distributions are estimated using the Variational Bayesian Expectation Maximization algorithm [17]. The predictive distribution for a VGMM is given by a mixture of Student's t-distributions

$$P(\mathbf{c}_h, \mathbf{c}_f) = \frac{1}{sum(\alpha)} \sum_{k=1}^{K} \alpha_k St(\mathbf{c}_h, \mathbf{c}_f|\mathbf{m}_k, \mathbf{L}_k, \nu_k + 1 - d) \tag{2.10}$$

where d is the number of degrees of freedom of the Wishart distribution and

$$\mathbf{L}_k = \frac{(\nu_k + 1 - d)\beta_k}{1 + \beta_k}\mathbf{W}_k \tag{2.11}$$

For a new trajectory history $\mathbf{c_h}$, the conditional predictive distribution $P(\mathbf{c_f}|\mathbf{c_h})$ is given

by:

$$P(\mathbf{c}_f|\mathbf{c}_h) = \frac{1}{sum(\hat{\alpha})}\sum_{k=1}^{K}\hat{\alpha}_k St\left(\mathbf{c}_f|\mathbf{c}_h, \hat{\mathbf{m}}_k, \mathbf{L}_k, \nu_k + 1 - d\right) \tag{2.12}$$

where (2.13)

$$\hat{\nu_k} = \nu_k + 1 - d \tag{2.14}$$

$$\hat{\alpha_k} = \frac{\alpha_k St(\mathbf{c}_h|\mathbf{m}_{k,\mathbf{c}_h}, \mathbf{L}_{k,\mathbf{c}_h}, \hat{\nu}_k)}{\sum_{j=1}^{K}\alpha_j St(\mathbf{c}_h|\mathbf{m}_{j,\mathbf{c}_h}, \mathbf{L}_{j,\mathbf{c}_h}, \hat{\nu}_j)} \tag{2.15}$$

$$\hat{\mathbf{m}}_k = \mathbf{m}_{k,\mathbf{c}_f} + \mathbf{\Sigma}_{k,\mathbf{c}_f\mathbf{c}_h}\mathbf{\Sigma}^{-1}_{k,\mathbf{c}_h\mathbf{c}_h}(\mathbf{c}_h - \mathbf{m}_{k,\mathbf{c}_h}) \tag{2.16}$$

$$\hat{\mathbf{L}}^{-1}_k = \frac{\hat{\nu}_k}{\hat{\nu}_k + d - 2}\left(1 + \Delta_k^T\frac{\mathbf{\Sigma}_{k,\mathbf{c}_h\mathbf{c}_h}}{\hat{\nu}_k}\Delta_k\right)\mathbf{\Sigma}^*_k \tag{2.17}$$

$$\Delta_k = (\mathbf{c}_h - \mathbf{m}_{k,\mathbf{c}_h}) \tag{2.18}$$

$$\mathbf{\Sigma}^*_k = \mathbf{\Sigma}_{k,\mathbf{c}_f\mathbf{c}_f} - \mathbf{\Sigma}_{k,\mathbf{c}_f\mathbf{c}_h}\mathbf{\Sigma}_{k,\mathbf{c}_h\mathbf{c}_h^{-1}}\mathbf{\Sigma}_{k,\mathbf{c}_h\mathbf{c}_f} \tag{2.19}$$

$$\mathbf{\Sigma}_k = \frac{\hat{\nu}_k + d - 2}{\hat{\nu}_k + d}\mathbf{L}^{-1}_k \tag{2.20}$$

2.6 Vehicle Interaction Module

The vehicle interaction module is tasked with assigning discrete maneuver labels to all vehicles in the scene at a particular prediction instant based on the confidence of the HMM in each maneuver class and the feasibility of the future trajectories of all vehicles based on those maneuvers given the current configuration of all vehicles in the scene. We set this up as an energy minimization problem. For a given prediction instant, let there be N surrounding vehicles in the scene with the top K maneuvers given by the HMM being considered for each vehicle. The minimization objective is given by:

$$\mathbf{y}^* = \arg\min_{\mathbf{y}}\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^i\left[E_{ik}^{hmm} + \lambda E_{ik}^{ego}\right] + \lambda\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{\substack{j=1\\j\neq i}}^{n}\sum_{l=1}^{K} y_k^i y_l^j E_{ijkl}^{vi} \tag{2.21}$$

s.t.

$$\sum_k y_k^i = 1 \quad \forall i \tag{2.22}$$

$$y_k^i = \begin{cases} 1, & \text{if car } i \text{ is assigned maneuver } k \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

The objective consists of three types of energies, the individual Energy terms E_{ik}^{hmm}, E_{ik}^{ego} and the pairwise energy terms E_{ijkl}^{vi}. The individual energy terms E_{ik}^{hmm} are given by the negative of the log likelihoods provided by the HMM. Higher the confidence of an HMM in a particular maneuver, lower is $-\mathcal{L}_k^i$ and thus the individual energy term. The individual energy term E_{ik}^{ego} takes into account the interaction between surrounding vehicles and the ego vehicle. We define the E_{ik}^{ego} as the reciprocal of the closest point of approach for vehicle i and the ego vehicle over the entire prediction horizon, given that it is performing maneuver k, where the ego vehicle position is always fixed to 0, since it is the origin of the frame of reference. Similarly, the pairwise energy term E_{ijkl}^{vi} is defined as the reciprocal of the minimum distance between the corresponding predicted trajectories for the vehicles i and j, assuming them to be performing maneuvers k and l respectively . The terms E_{ik}^{ego} and E_{ijkl}^{vi} penalize predictions where at any point in the prediction horizon, two vehicles are very close to each other. This term leverages the fact that drivers tend to follow paths with low possibility of collisions with other vehicles. The weighting constant λ is experimentally determined through cross-validation.

The minimization objective in the formulation shown in Eq. 2.21, 2.22 and 2.23 has quadratic terms in y values. In order to leverage integer linear programming for minimizing the energy, we modify the formulation as follows:

$$\mathbf{y}^*, \mathbf{z}^* = \arg\min_{\mathbf{y},\mathbf{z}} \sum_{i=1}^{n}\sum_{k=1}^{K} y_k^i \left[E_{ik}^{hmm} + \lambda E_{ik}^{ego}\right] + \lambda \sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{\substack{j=1 \\ j\neq i}}^{n}\sum_{l=1}^{K} z_{k,l}^{i,j} E_{ijkl}^{vi} \tag{2.24}$$

s.t.

$$\sum_k y_k^i = 1 \quad \forall i \tag{2.25}$$

$$y_k^i \in 0, 1 \tag{2.26}$$

$$z_{k,l}^{i,j} \leq y_k^i \tag{2.27}$$

$$z_{k,l}^{i,j} \leq y_l^j \tag{2.28}$$

$$z_{k,l}^{i,j} \geq y_k^i + y_l^j - 1 \tag{2.29}$$

This objective can now be optimized using integer linear programming. The optimal values $\mathbf{y}^*$ give the maneuver assignments for all vehicles. These assigned maneuvers are used by the trajectory prediction module to make future predictions for all vehicles.

2.7 Experimental Evaluation

2.7.1 Dataset

We evaluate our framework using real freeway traffic data captured using the testbed described in [142]. The vehicle is equipped with 8 RGB video cameras, LIDARs and RADARs synchronously capturing the full surround at a frame rate of 15 fps. Our complete dataset consists of 52 video sequences extracted from multiple drives spanning approximately 45 minutes. The sequences were chosen to capture varying lighting conditions, vehicle types, and traffic density and behavior.

The 4 longest video sequences, of about 3 minutes each were ground-truthed by human annotators and used for evaluation. Three sequences from the evaluation

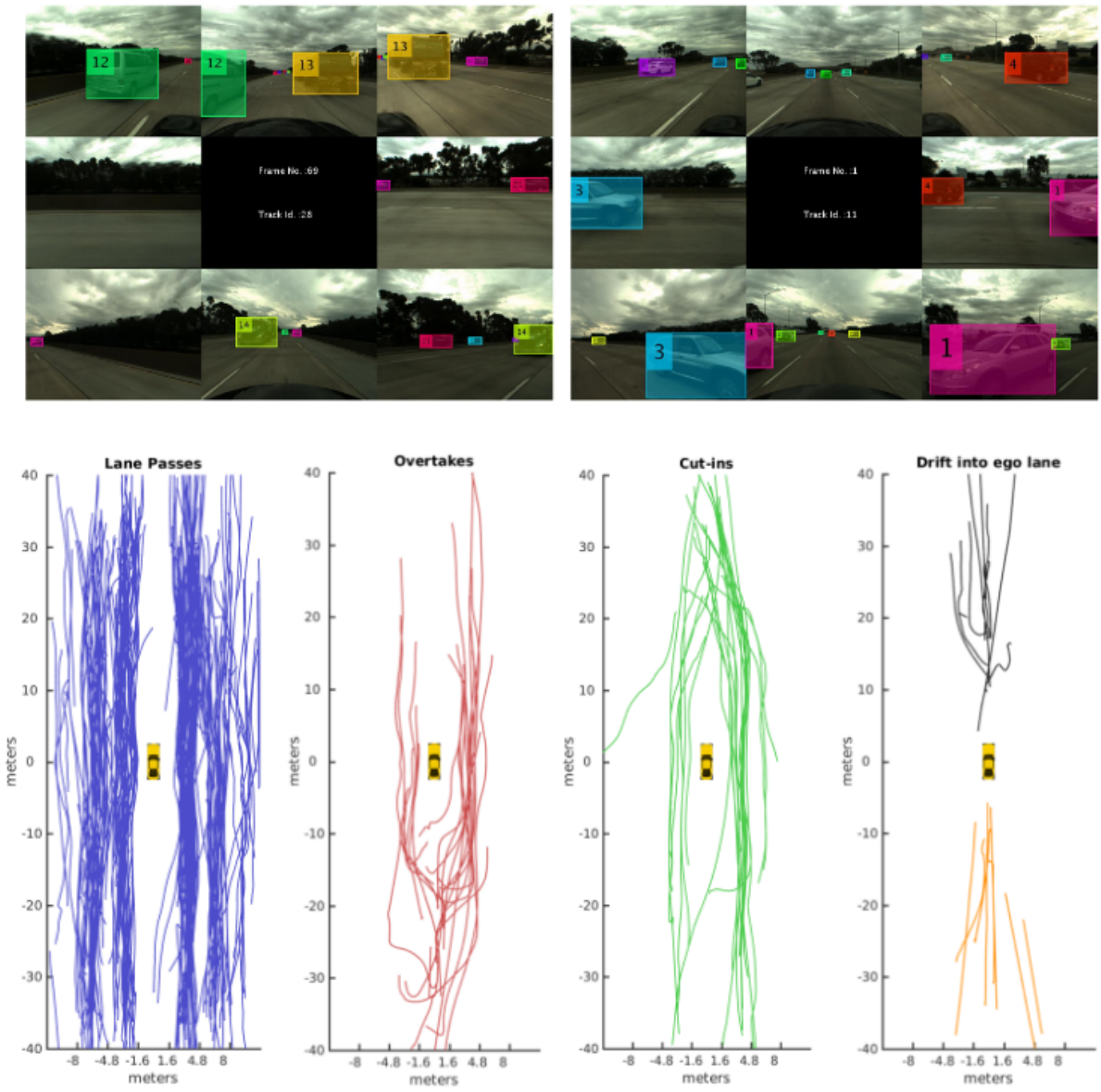


Figure 2.4. Dataset: Examples of annotated frames from the evaluation set (top left and top right) and trajectories belonging to all maneuver classes projected in the ground plane (bottom). We can observe that the trajectory patterns implicitly capture lane information

set represent light to moderate or *free-flowing* traffic conditions, while the remaining sequence represents heavy or *stop-and-go* traffic. The video feed from the evaluation set was annotated with detection boxes and vehicle track-ids for each of the 8 views. All tracks were then projected to the ground plane and assigned a maneuver class label corresponding to the 10 maneuver classes described in Section 2.4.1. If a vehicle track was

Table 2.1. Dataset Statistics

Maneuver	**Number of trajectories**	**Number of trajectory snippets**
Lane Pass (Left Forward)	59	9500
Lane Pass (Left Back)	75	10332
Lane Pass (Right Forward)	110	10123
Lane Pass (Right Back)	48	12523
Overtake (Left)	8	1629
Overtake (Right)	17	2840
Cut-in (Left)	8	1667
Cut-in (Right)	19	3201
Drift into ego lane (Front)	11	1317
Drift into ego lane (Rear)	8	553

comprised by multiple maneuvers, the start and end-point of each maneuver was marked. A multi-perspective tracker [49] was used for assigning vehicle tracks for the remaining 48 sequences. These tracks were only used for training the models. Figure 2.4 shows the track annotations as well as the complete set of trajectories for each maneuver. Since each trajectory is divided into overlapping snippets of $t_h = 3$ seconds for training and testing our models, we report the data statistics in terms of the total number of trajectories as well as the number of trajectory snippets for each maneuver class in Table 2.1

We report all results using a leave on sequence cross-validation scheme. For each of the 4 evaluation sequences, the HMMs and VGMMs are trained using data from the remaining 3 evaluation sequences as well as the 48 training sequences. Additionally, we use two simple data-augmentation schemes for increasing the size of our training datasets in order to reduce overfitting in the models:

1. *Lateral inversion:* We flip each trajectory along the lateral direction in the ego frame to give an instance of a different maneuver class. For example, a left cut-in on lateral inversion becomes a right cut in.

2. *Longitudinal shifts:* We shift each of the trajectories by ± 2, 4 and 6 m in the

longitudinal direction in the ego frame to give additional instances of the same maneuver class. We avoid lateral shifts since this would interfere with lane information that is implicitly learned by the probabilistic model.

2.7.2 Evaluation Measures and Experimental Settings

Our models predict the future trajectory over a prediction horizon of 5 seconds for each 3 second snippet of track history based on the maneuver classified by the HMMs or by the VIM. We use the following evaluation measures for reporting our results:

1. *Mean Absolute Error*: This measure gives the average absolute deviation of the predicted trajectories from the underlying ground truth trajectories. To compare how the models perform for short term and long term predictions, we report this measure separately for prediction instants up to 5 seconds into the future, sampled with increments of 1 second. The mean absolute error captures the effect of both the number of errors made by the models as well as the severity of the errors.

2. *Median Absolute Error*: We also report the median values of the absolute deviations for up to 5 seconds into the future with 1 second increments, as was done in [156]. The median absolute error better captures the distribution of the errors made by the models while sifting out the effect of a few drastic errors.

3. *Maneuver classification accuracy*: We report maneuver classification accuracy for configurations using the maneuver recognition module or the VIM.

4. *Execution time*: We report the average execution time per frame, where each frame involves predicting trajectories of all vehicles being tracked at a particular instant.

In order to analyze the effect of each of our proposed modules, we compare the trajectory prediction results for following systems

- *Motion model (**IMM**):* We use the trajectories predicted by the IMM based motion model as our baseline.

- *Monolithic VGMM (**M-VGMM**):* We consider the trajectories predicted by our trajectory prediction module, where the probabilistic model used is a single monolithic VGMM. This alleviates the need for the maneuver recognition module, since the same model makes predictions irrespective of the maneuver being performed

- *Class VGMMs (**C-VGMM**):* Here we consider separate VGMMs for each maneuver class in the trajectory prediction module. We use the VGMM corresponding to the maneuver with the highest HMM log likelihood for making the prediction. In this case, maneuver predictions for each vehicle are made independent of the other vehicles in the scene. To keep the comparison with the M-VGMM fair, we use 8 mixture components for each maneuver class for the C-VGMMs, while we use a single VGMM with 80 mixture components for the M-VGMM, ensuring that both models have the same complexity.

- *Class VGMMs with Vehicle Interaction Module (**C-VGMM + VIM**):* We finally consider the effect of using the vehicle interaction module. In this case, we use the C-VGMMs with the maneuver classes for each of the vehicles in the scene assigned by the vehicle interaction module

We report our results for the complete set of trajectories in the evaluation set. Additionally, we also report results on the subsets of overtake and cut-in maneuvers and stop-and-go traffic. Since overtakes and cut-ins are rare safety critical maneuvers with significant deviation from uniform motion, these are challenging cases for trajectory prediction. Similarly, due to the high traffic density in stop-and-go scenarios, vehicles affect each others motion to a much greater extent as compared to free-flowing traffic, making it a challenging scenario for trajectory prediction.

Table 2.2. Quantitative results showing ablative analysis of our proposed model

Metric	Setting	All Trajectories				Overtakes and Cut-ins			Stop-and-Go Traffic		
	Prediction Horizon (s)	IMM	M-VGMM	C-VGMM	C-VGMM + VIM	IMM	M-VGMM	C-VGMM	IMM	C-VGMM	C-VGMM +VIM
Mean Absolute Error (m)	1	0.25	**0.24**	**0.24**	**0.24**	0.29	0.32	**0.29**	0.22	0.20	**0.20**
	2	0.72	0.70	**0.69**	**0.69**	0.83	0.87	**0.82**	0.68	0.65	**0.64**
	3	1.25	1.19	**1.18**	**1.18**	1.47	1.46	**1.39**	1.21	1.17	**1.14**
	4	1.78	1.70	1.68	**1.66**	2.17	2.05	**1.94**	1.74	1.68	**1.65**
	5	2.36	2.24	2.20	**2.18**	2.90	2.68	**2.49**	2.29	2.21	**2.17**
Median Absolute Error (m)	1	0.19	**0.17**	**0.17**	**0.17**	**0.23**	**0.23**	**0.23**	0.15	**0.13**	**0.13**
	2	0.55	**0.52**	**0.52**	**0.52**	0.68	**0.65**	**0.65**	0.48	0.46	**0.45**
	3	0.96	0.92	**0.91**	**0.91**	1.24	1.13	**1.12**	0.89	0.87	**0.83**
	4	1.38	1.32	1.30	**1.29**	1.92	1.71	**1.68**	1.32	1.29	**1.27**
	5	1.85	1.77	**1.72**	**1.72**	2.64	2.27	**2.12**	1.8	1.78	**1.75**
Class. acc. (%)	-	-	-	83.49	**84.24**	-	-	55.89	-	84.84	**87.19**
Exec. time (s)	-	**0.0346**	0.1241	0.0891	0.1546	-	-	-	-	-	-

2.7.3 Ablative Analysis

Table 2.2 shows the quantitative results of our ablation experiments. We note from the results on the complete evaluation set that the probabilistic trajectory prediction models outperform the IMM. The M-VGMM has lower values for both mean as well as median absolute error as compared to the IMM suggesting that the probabilistic model makes fewer as well as less drastic errors on an average. We get further improvements in mean and median absolute deviations using the C-VGMMs suggesting that subcategorizing trajectories into maneuver classes leads to a better probabilistic prediction model.

This is further highlighted based on the prediction results for the challenging maneuver classes of overtakes and cut-ins. We note that the C-VGMM significantly outperforms the CV and M-VGMM models both in terms of mean and median absolute deviation for overtakes and cut-ins. This trend becomes more pronounced as the prediction horizon is increased. This suggests that the motion model is more error prone due to the non-uniform motion in overtakes and cut-ins while these rare classes get underrepresented in the distribution learned by the monolithic M-VGMM. Both of these issues get addressed through the C-VGMM. We analyze this further by considering specific cases of predictions made by the IMM, M-VGMM and C-VGMM in Section 2.7.5

Comparing the maneuver classification accuracies for the case of C-VGMM and C-VGMM + VIM, we note that the VIM corrects some of the maneuvers assigned by the HMM. This in turn leads to improved trajectory prediction as seen from the mean and median absolute error values. We note that this effect is more pronounced in case of stop-and-go traffic, since the dense traffic conditions cause more vehicles to affect each others motion leading to a greater proportion of maneuver class labels to be re-assigned by the VIM. Section 2.7.6 analyses cases where the VIM reassigns maneuver labels assigned by the HMM due to the relative configuration of all vehicles in the scene.

2.7.4 Analysis of execution time

Table 2.2 also shows the average execution time per frame for the 4 system configurations considered. As expected, the IMM baseline has the lowest execution time since all other configurations build upon it. We note that the C-VGMM runs faster than the M-VGMM in spite of having the overhead of the HMM based maneuver recognition module. This is because the M-VGMM is a much bulkier model as compared to any single maneuver C-VGMM. Thus in spite of involving an extra step, the maneuver recognition module allows us to choose a much leaner model, effectively reducing the execution time while improving performance. The VIM is a more time intensive overhead and almost doubles the run time of the C-VGMM. However, even in it's most complex setting, the proposed framework can be deployed at a frame rate of almost 6 fps, which is more than sufficient for the application being considered.

2.7.5 Qualitative Analysis of Predictions

Figure 2.5 shows the trajectories predicted by the CV, M-VGMM and C-VGMM models for 8 different instances.

Figure 2.5a shows two prediction instants where the vehicle is just about to start the non-linear part of overtake maneuvers. We observe that the IMM makes erroneous predictions in both cases. However, both the M-VGMM and C-VGMM manage to predict the non-linear future trajectory.

Figure 2.5b shows two prediction instants in the early part of overtake maneuvers. We note that both the IMM and M-VGMM make errors in prediction. However the position of the surrounding vehicle along with the slight lateral motion provide enough context to the maneuver recognition module to detect the overtake maneuver early. Thus, the C-VGMM manages to predict that the surrounding vehicle would move to the adjacent lane and accelerate in the longitudinal direction, although there is no such cue from the

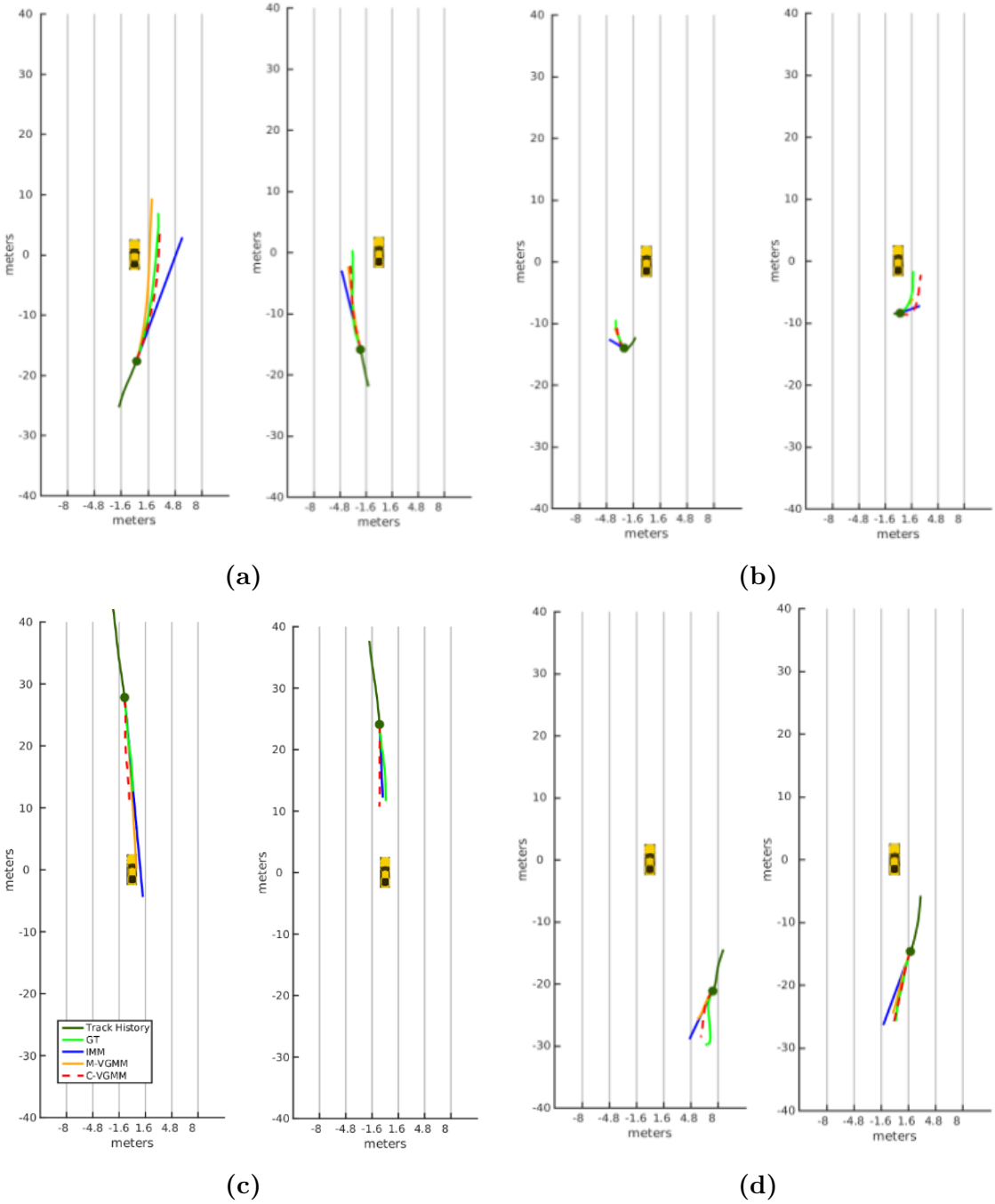
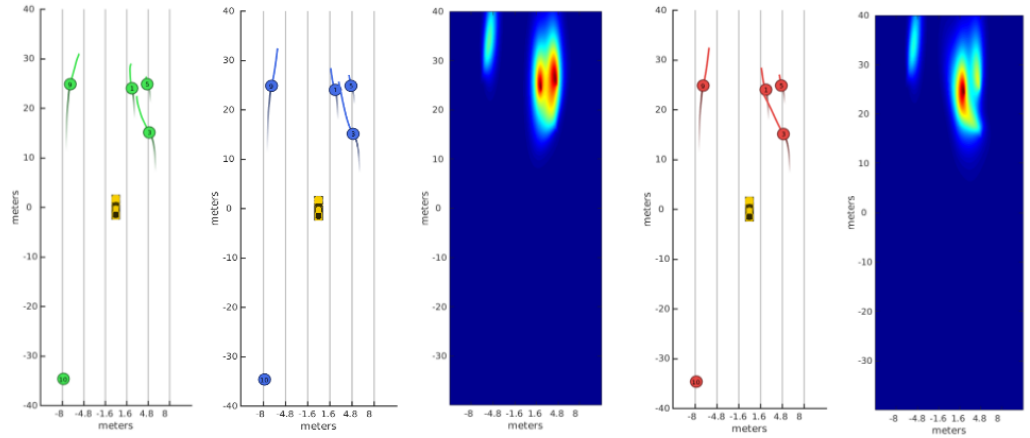
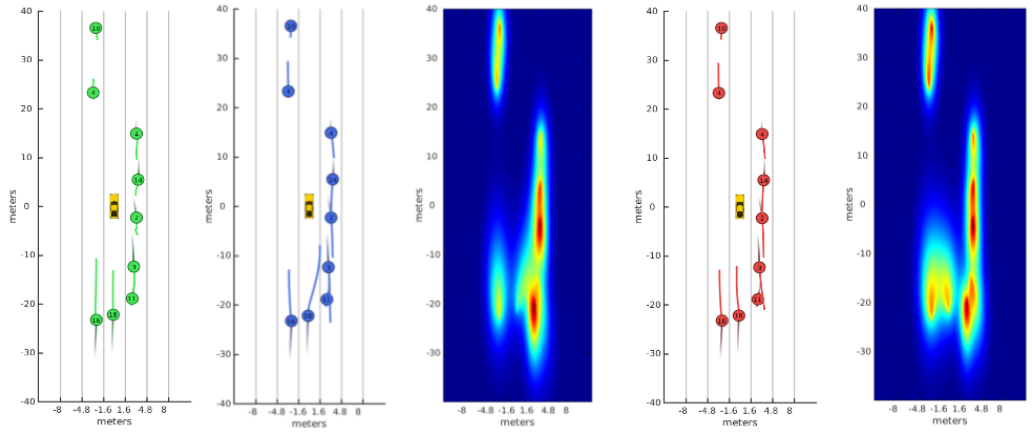


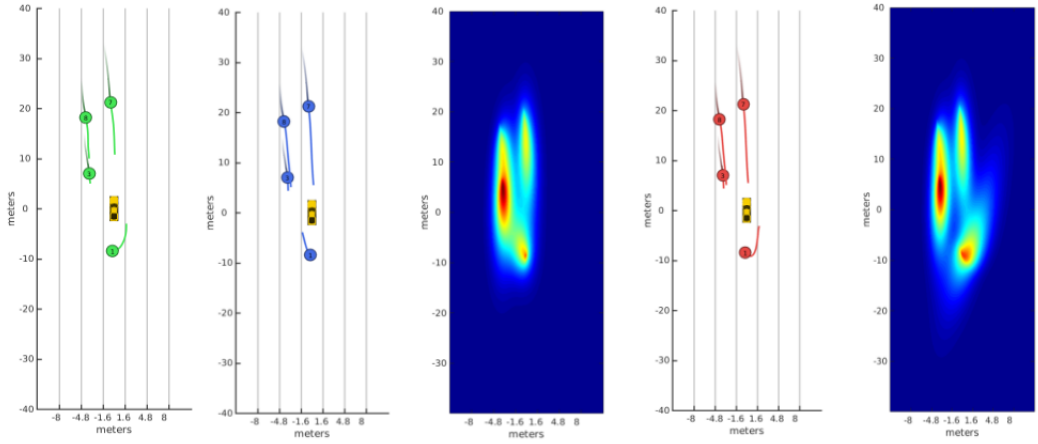
Figure 2.5. Predictions made by CV, M-VGMM and C-VGMM models: (a): Better prediction of lateral motion in overtakes by the probabilistic models. (b): Early detection of overtakes by the HMM. (c): Deceleration near the ego vehicle predicted by the C-VGMM. (d): Effect of lane information implicitly encoded by the M-VGMM and C-VGMM



(a) Infeasible lane pass is correctly changed to cut-in



(b) Infeasible overtake correctly changed to tail-gating



(c) Infeasible left overtake changed to the correct overtake direction

Figure 2.6. Effect of the VIM: Each case shows from left to right: The ground truth, predictions made independently for each vehicle, uncertainty of the independent predictions, predictions made with the VIM, uncertainties of the VIM predictions

vehicles existing state of motion

Figure 2.5c shows two instants the trajectory of a vehicle that decelerates as it approaches the ego vehicle from the front. This trajectory corresponds to the drift into ego-lane maneuver class. In the first case (left), the vehicle has not started decelerating, causing the IMM to assign a high probability to the CV model. The IMM thus predicts the vehicle to keep moving at a constant velocity and come dangerously close to the ego vehicle. Similarly, the M-VGMM makes a poor prediction since these maneuvers are underrepresented in the training data. The C-VGMM however manages to correctly predict the surrounding vehicle to decelerate. In the second case (right), we observe that the car has already started decelerating. This allows the IMM to assign a greater weight to the CA model and correct its prediction

Finally Figure 2.5d shows two interesting instances of the lane pass right back maneuver that is well represented in the training data. The vehicle makes a lane change in both of these instances. The IMM poorly predicts these trajectories. However both the M-VGMM and C-VGMM correctly predict the vehicle to merge into the lane, suggesting that the probabilistic models may have implicitly encoded lane information.

2.7.6 Vehicle Interaction Model Case Studies

Figure 2.6 shows three cases where the recognized maneuvers and predicted trajectories are affected by the VIM. In each case, the green plots show the ground truth of future tracks, the blue plots show the predictions made for each vehicle independently and the red plots show the predictions based on the VIM. Additionally we plot the prediction uncertainties for either case.

Consider the first case in Figure 2.6a, in particular vehicle 3. We note from the blue plot that the HMM predicts the vehicle to perform a lane pass. However the the vehicle's path forward is blocked by vehicles 1 and 5. The VIM thus infers vehicle 3 to perform a cut-in in with respect to the ego-vehicle in order to overtake vehicle 5.

In Figure 2.6b, the HMM predicts vehicle 18 to overtake the ego-vehicle from the right. However, we can see that the right lane is occupied by vehicles 11, 3 and 2. These vehicles yield high values of pairwise energies with vehicle 18 for the overtake maneuver. The VIM thus correctly manages to predict that vehicle 18 would end up tail-gating by assigning it the maneuver drift into ego lane (rear).

Finally Figure 2.6c shows a very interesting case where the HMM predicts vehicle 1 to overtake the ego vehicle from the left. Again, the left lane is occupied by other vehicles making the overtake impossible to execute from the left. However, compared to the previous case, these vehicles are slightly further away and can be expected to yield relatively smaller energy terms as compared to case (b). However, these terms are enough to offset the very slight difference in the HMM's confidence values between the left and right overtake since both maneuvers do seem plausible if we consider vehicle 1 independently. Thus the VIM reassigns the maneuver for vehicle 1 to a right overtake, making the prediction closely match the ground truth.

2.8 Conclusions

In this chapter, we presented a unified framework for surrounding vehicle maneuver recognition and motion prediction using vehicle mounted perceptual sensors, that leverages the instantaneous motion of vehicles, an understanding of motion patterns of freeway traffic and the effect of inter-vehicle interactions. The proposed framework outperforms an interacting multiple model based trajectory prediction baseline and runs in real time.

We presented an ablative analysis for the relative importance of each cue for trajectory prediction. In particular, we showed that probabilistic modeling of surrounding vehicle trajectories is a more versatile approach, and leads to better predictions compared to using motion models, especially for safety critical trajectories around the ego vehicle. Additionally, subcategorizing trajectories based on maneuvers leads to better modeling

of motion patterns. Finally, incorporating a model for interactions between surrounding vehicles to simultaneously predict each of their motion leads to better predictions as compared to predicting each vehicle's motion independently.

The proposed approach could be treated as a general framework, where improvements could be made to each of the three interacting modules.

Acknowledgements

Chapter 2, in full, is a reprint of the material as it appears in: "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," Nachiket Deo, Akshay Rangesh, and Mohan M. Trivedi, IEEE Transactions on Intelligent Vehicles 2018. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Convolutional Social Pooling and Maneuver Based LSTMs

3.1 Introduction

In the previous chapter, we introduced a unified framework for predicting the trajectories of surrounding vehicles in highway traffic that leverages vehicle maneuvers and models interaction between vehicles. While we obtain promising results, the framework suffers from a few limitations.

- **Independently trained modules:** The framework uses three independently trained modules for maneuver recognition, trajectory prediction and modeling vehicle interaction that are applied sequentially during inference. The errors made by upstream modules are propagated downstream during inference but not addressed during training. The upstream modules (e.g. the maneuver recognition module) are trained agnostic to their effect on the overall prediction error. On the other hand, the downstream modules (e.g. the trajectory prediction module) lack robustness to errors made by the upstream modules.

- **Redundant encoders:** The past trajectories of surrounding vehicles are independently encoded, by the parameters of the HMMs in the maneuver recognition modules, and by the parameters of the VGMMs in the trajectory prediction module.

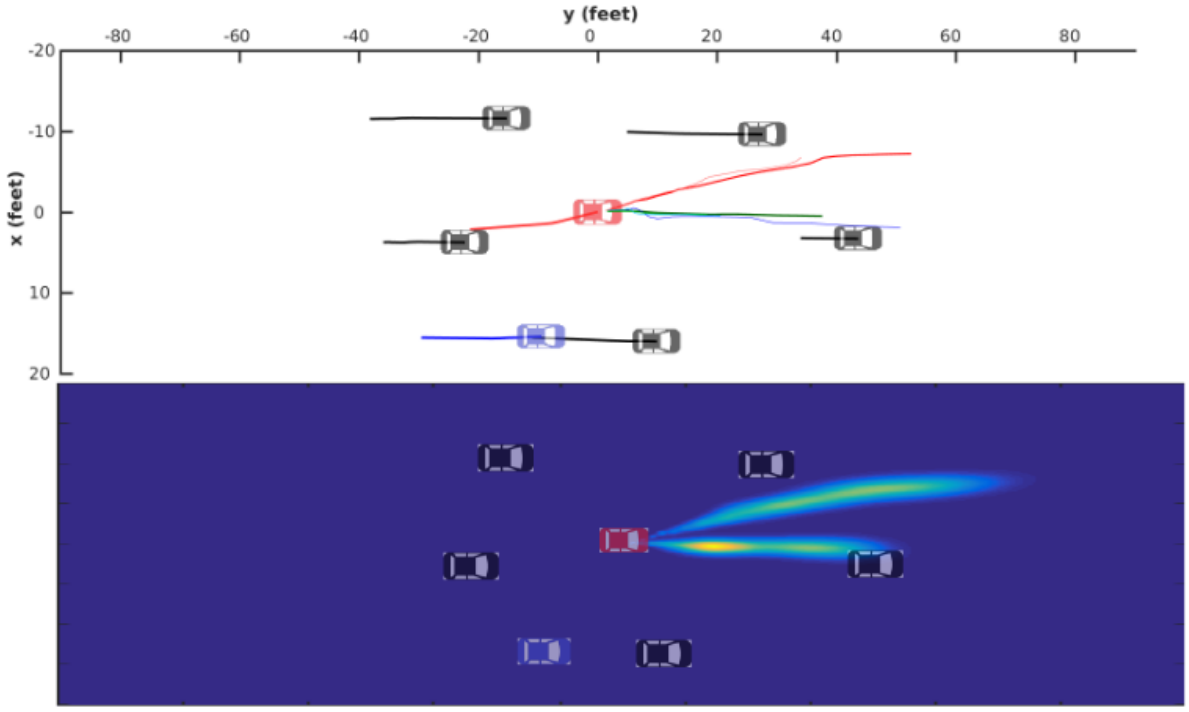


Figure 3.1. Multimodal predictions for highway traffic: Imagine the blue vehicle is an autonomous vehicle in the traffic scenario shown. Our proposed model allows it to make multimodal predictions of future motion of it's surrounding vehicles, along with prediction uncertainty shown here for the red vehicle

Both tasks are tightly linked and could use shared feature encoders.

- **Modeling relative motion:** The trajectory prediction module predicts the *relative motion* of surrounding vehicles with respect to the ego-vehicle. The predictive distribution does not decouple ego-vehicle motion and surrounding vehicle motion. We implicitly predict both, the ego-vehicle's trajectory as well as the surrounding vehicles' trajectories. This can be a problem, since path planners would have control over the ego-vehicle's future trajectory, but not on the surrounding vehicles'.

- **Unimodal predictions:** Finally, the predictive distribution from the model is *unimodal*, characterized by the most likely maneuver class for each surrounding vehicle. Driver behavior tends to be inherently multimodal, where a driver could

make one of many decisions under the same traffic circumstances. A model that predicts a multimodal distribution over future trajectories as shown in Figure 3.1, would help path planners in autonomous vehicles to plan for contingencies rather than being overly confident in the most likely maneuver.

3.1.1 Contributions

In this chapter, we address the above limitations by leveraging the modularity and end-to-end trainability of deep learning models. Instead of independently trained modules for maneuver recognition, trajectory prediction and modeling vehicle interaction, we propose a single model consisting of differentiable components performing these functions. Following the success of long-short term memory (LSTM) networks in modeling non-linear temporal dependencies in sequence learning and generation tasks [3, 28, 56], we propose an LSTM encoder-decoder model for vehicle trajectory prediction. Our model can be trained end-to-end via gradient based learning, and is characterized by:

1. **Shared LSTM encoders:** We use LSTM encoders with shared weights for encoding the past trajectories of all surrounding vehicles. The LSTM encodings are then used for both maneuver recognition and trajectory prediction.

2. **Convolutional social pooling:** We propose a novel social pooling layer as an alternative to that proposed in [3]. We apply convolutional and max-pooling layers instead of a fully connected layer to *social-tensors* of LSTM states that encode the past motion of neighboring vehicles. Through our experiments, we show that convolutional social pooling is more robust to varying spatial configurations of agents than fully connected social pooling.

3. **Maneuver based decoder:** We propose an LSTM decoder that generates the probability distribution over future motion for six maneuver classes and assigns a probability to each maneuver class. This accounts for the multimodal nature of

vehicle motion. We propose a loss function that allows us to train the model without collapsing the modes of the predictive distribution.

3.2 Related Research

3.2.1 Maneuver based models:

Classification of vehicle motion into meaningful maneuvers has been extensively addressed in both advanced driver assistance systems as well as naturalistic drive studies. Of particular interest are works that use recognized maneuvers to make better predictions of future trajectories [35,68,88,157,158,170]. These approaches usually involve a maneuver recognition module for classifying maneuvers and maneuver specific trajectory prediction modules. Maneuver recognition modules are typically classifiers that use past positions and motion states of the vehicles, and context cues as features. Heuristic based classifiers [68], Bayesian networks [158], hidden Markov models [35,88], random forest classifiers [157] and recurrent neural networks have been used for maneuver recognition. Trajectory prediction modules output the future locations of the vehicle given its maneuver class. Polynomial fitting [68], maneuver specific motion models [158], Gaussian processes [88, 170], and Gaussian mixture models [35] have been used for trajectory prediction.

3.2.2 Interaction aware models:

Interaction aware models for motion prediction take into account the effect of inter-vehicle interaction on the motion of vehicles. Two different approaches can be found for incorporating inter-vehicle interaction. The first set of approaches [7, 35] use hand crafted cost functions based on the relative configuration of vehicles and make optimal predictions of future motion with respect to these cost functions. Cost function based approaches do not depend on training data and can generalize to new traffic configurations. However, they can be limited by how well the hand-crafted cost function is designed. The second approach to incorporate inter-vehicle interaction is to implicitly learn it from

trajectory data of real traffic. However, due to the large variation in traffic configurations, this approach requires a large dataset for generalization. This approach has been used in prior works for the case of two vehicles approaching an intersection [78], and lateral motion prediction on highways [157]. We use the data-driven approach for inter-vehicle interaction in this work, since it it not limited by the design of a hand-crafted cost function, and also due to the availability of large datasets of real freeway traffic [30, 31].

3.2.3 Recurrent networks for motion prediction:

Since motion prediction can be viewed as a sequence classification or sequence generation task, a number of recurrent neural network (RNN) based approaches have been proposed in recent times for maneuver classification and trajectory prediction. Khosroshahi *et al.* [80] and Phillips *et al.* [130] use LSTMs to classify vehicle maneuvers at intersections. Kim *et al.* [81] propose an LSTM that predicts the location of vehicles in an occupancy grid at intervals of 0.5 s, 1 s and 2 s into the future. Contrary to this approach, our model outputs a continuous, multimodal probability distribution of future locations of the vehicles up to a prediction horizon of 5 s. Lee *at al.* [90] propose a model that combines conditional variational auto-encoders (CVAE) with RNN encoder-decoders for trajectory prediction. While this allows for multimodal predictions by sampling the CVAE, the model can only provide samples from the predictive distribution rather than an estimate of the distribution itself. In their seminal work, Alahi *et al.* [3] propose *social LSTMs*, which jointly model and predict the motion of pedestrians in dense crowds through the use of a social pooling layer. We improve upon this approach by using convolutional social pooling. We also incorporate the lane structure of freeways into our social pooling layer. Finally, Kuefler *et al.* [87] use a gated recurrent unit (GRU) based policy using the behavior cloning and generative adversarial imitation learning paradigms to generate the acceleration and yaw-rate values of a bicycle model of vehicle motion. We compare our trajectory prediction results with those reported in [87].

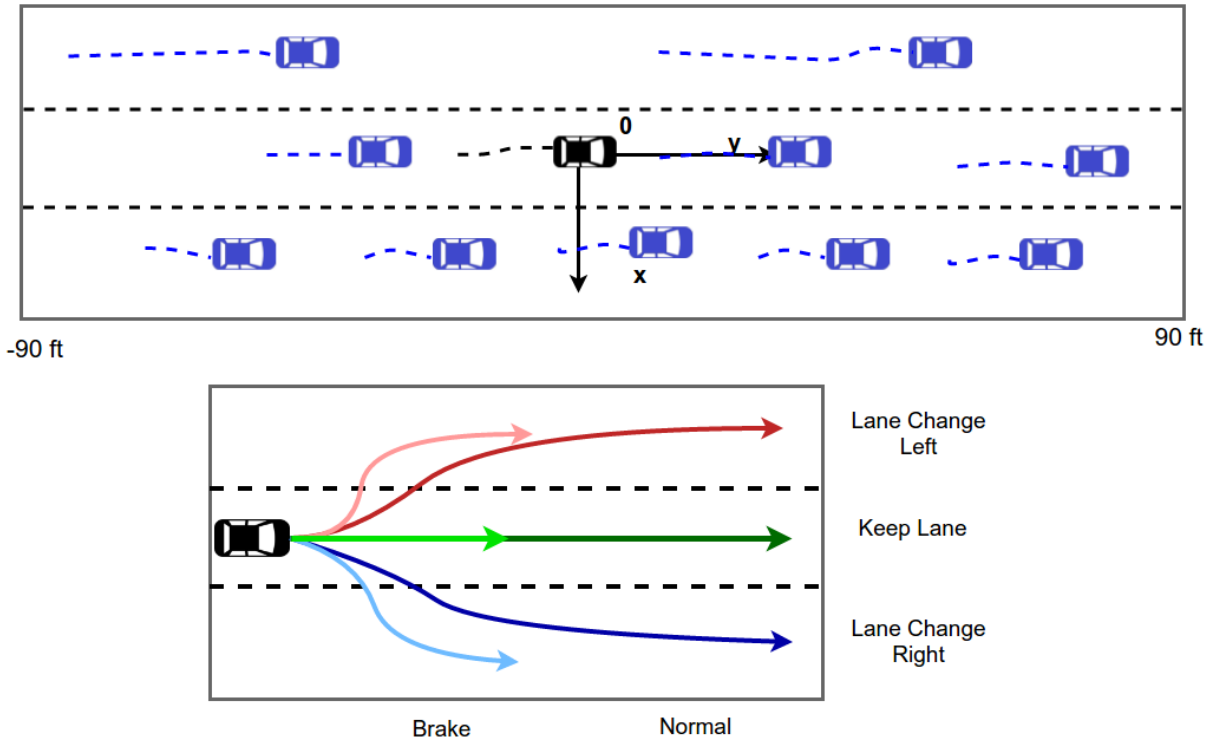


Figure 3.2. Formulation. Top: The co-ordinate system used for trajectory prediction. The vehicle being predicted is shown in black, neighboring vehicles considered are shown in blue. Bottom: Lateral and longitudinal maneuver classes

3.3 Formulation

We formulate motion prediction as estimating the probability distribution of the future positions of a vehicle conditioned on its track history and the track histories of vehicles around it, at each time instant t.

3.3.1 Frame of reference

We use a stationary frame of reference, with the origin fixed at the vehicle being predicted at time t as shown in Fig. 3.2. The *y-axis* points in the direction of motion of the freeway, and the *x-axis* is the direction perpendicular to it. This makes our model independent of how the vehicle tracks were obtained, and in particular, can be applied to the case of on-board sensors on an autonomous vehicle. This also makes the model

independent of the curvature of the road, and can be applied anywhere on a freeway as long as an on-board lane estimation algorithm is available.

3.3.2 Inputs and outputs

The inputs to our model are track histories

$$\mathbf{X} = [\mathbf{x}^{(t-t_h)}, ..., \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}] \tag{3.1}$$

where,

$$\mathbf{x}^{(t)} = [x_0^{(t)}, y_0^{(t)}, x_1^{(t)}, y_1^{(t)}, ..., x_n^{(t)}, y_n^{(t)}] \tag{3.2}$$

are the x and y co-ordinates at time t of the vehicle being predicted and all vehicles within $\pm$90 feet in the longitudinal direction and within the two adjacent lanes of the vehicle being predicted, as shown in Fig. 3.2.

The output of the model is a probability distribution over

$$\mathbf{Y} = [\mathbf{y}^{(t+1)}, ..., \mathbf{y}^{(t+t_f)}] \tag{3.3}$$

where,

$$\mathbf{y}^{(t)} = [x_0^{(t)}, y_0^{(t)}] \tag{3.4}$$

are the future co-ordinates of the vehicle being predicted

3.3.3 Probabilistic motion prediction

Our model estimates the conditional distribution $\mathrm{P}(\mathbf{Y}|\mathbf{X})$. In order to have the model produce multimodal distributions, we expand it in terms of maneuvers m_i, giving:

$$\mathrm{P}(\mathbf{Y}|\mathbf{X}) = \sum_i \mathrm{P}_\Theta(\mathbf{Y}|m_i, \mathbf{X})\mathrm{P}(m_i|\mathbf{X}) \tag{3.5}$$

where,

$$\Theta = [\Theta^{(t+1)}, ..., \Theta^{(t+t_f)}] \tag{3.6}$$

are the parameters of a bivariate Gaussian distribution at each time step in the future, corresponding to the means and variances of future locations.

3.3.4 Maneuver classes

We consider three lateral and two longitudinal maneuver classes as shown in Fig. 3.2. The lateral maneuvers consist of left and right lane changes and a lane keeping maneuver. Since lane changes involve preparation and stabilization, we define a vehicle to be in a lane changing state for $\pm$ 4s w.r.t. the actual cross-over. The longitudinal maneuvers are split into normal driving and braking. We define a vehicle to be performing a braking maneuver if it's average speed over the prediction horizon is less than 0.8 times its speed at the time of prediction. We define our maneuvers in this manner since these maneuver classes are communicated by vehicles to each other through turn signals and brake lights, which will be included as a cue in future work.

3.4 Proposed Model

Fig. 3.3 shows our proposed model. It consists of an LSTM encoder, convolutional social pooling layers and a maneuver based LSTM decoder.

3.4.1 LSTM Encoder

We use an LSTM encoder for learning the dynamics of vehicle motion. For each instant, snippets of the most recent t_h frames of track history are passed through the LSTM encoder for the vehicle being predicted, and all the vehicles surrounding it. The LSTM states for each vehicle are updated frame by frame over the t_h past frames. The final LSTM state for each vehicle can be expected to encode the state of motion of that

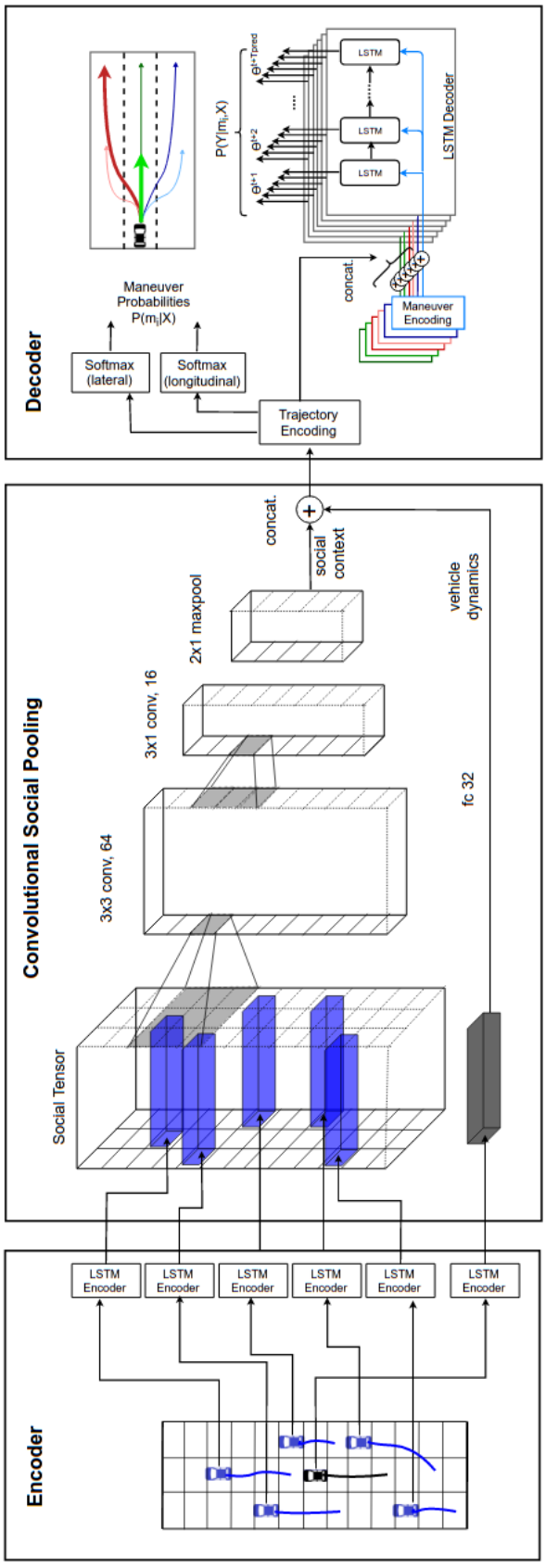


Figure 3.3. Proposed Model: The encoder is an LSTM with shared weights that learns vehicle dynamics based on track histories. The convolutional social pooling layers learn the spatial interdependencies of of the tracks. Finally, the maneuver based decoder outputs a multimodal predictive distribution for the future motion of the vehicle being predicted

vehicle. The LSTMs used for each vehicle have shared weights. This allows for a direct correspondence between the components of the LSTM states for all the vehicles.

3.4.2 Convolutional Social Pooling

While the LSTM encoder captures the vehicle motion dynamics, it fails to capture the interdependencies of the motion of all vehicles in the scene. Social pooling, proposed in [3], addresses this by pooling the LSTM states of all the agents around the agent being predicted into a *social tensor*. This is done by defining a spatial grid around the agent being predicted and populating the grid with LSTM states based on the spatial configuration of the agents in the scene. Fig. 3.3 shows an example of a social tensor. Using this social tensor as the input to the model in addition to the LSTM state of the agent being predicted, has been shown to improve the accuracy of future motion prediction [3, 90]. This makes sense since the model now gets access to the motion states of surrounding agents and their spatial configuration.

However, all previous instances of social pooling [3, 90] apply a fully connected layer to the social tensor. This is inefficient since it breaks up the spatial structure of the social tensor. Cells adjacent to each other in space become equivalent to cells far away from each other in the fully connected layer. This can lead to problems in generalization to a test set especially if the agents can be in various different spatial configurations. For example, let's suppose the training set doesn't have a single instance of an LSTM state at spatial location (m, n) of the social tensor. When such an instance is now encountered in the test set, the model will fail to generalize. In particular, this will hold even if there are training instances of LSTM states at spatial grid locations $(m + 1, n)$ and $(m, n + 1)$, say, in spite of these instances clearly being helpful due to spatial locality.

As a remedy, we propose the use of convolutional and pooling layers over the social tensor, termed *convolutional social pooling*. The equivariance of the convolutional layers can be expected to help learn locally useful features within the spatial grid of the social

tensor, and the max-pooling layer can be expected to add local translational invariance, both of which help address the problem described above. This phenomenon has been further explored in section 3.5.5.

We set up our social tensor by defining a grid based on the lanes. A 13×3 spatial grid is defined around the vehicle being predicted, where each column corresponds to a single lane, and the rows are separated by a distance of 15 feet which approximately equals one car length. The social tensor is formed by populating this grid with surrounding car locations. We then apply two convolutional layers and a pooling layer to the social tensor as shown in Fig. 3.3 to obtain the social context encoding. Additionally, the LSTM state of the predicted vehicle is passed through a fully connected layer to obtain the vehicle dynamics encoding. The two encodings are concatenated to form the complete trajectory encoding, which is then passed to the decoder.

3.4.3 Maneuver based LSTM decoder

We use an LSTM based decoder for generating the predictive distribution for future motion over the next t_f frames. We address the inherent multimodality of driver behavior by predicting the distribution for each of the six maneuver classes described in section 3.3.4 along with the probability for each maneuver class. The decoder has two softmax layers that output the lateral and longitudinal maneuver probabilities. These can be multiplied to give the values of $\mathrm{P}(m_i|\mathbf{X})$ from Eqn. 3.5. Additionally, an LSTM is used to generate the parameters of a bivariate Gaussian distribution over t_f frames to give the predictive distribution for vehicle motion. In order to obtain maneuver specific distributions $\mathrm{P}_\Theta(\mathbf{Y}|m_i, \mathbf{X})$ from Eqn, 3.5, we concatenate the trajectory encoding with a one-hot vector corresponding to the lateral maneuver class and a one-hot vector corresponding to the longitudinal maneuver class.

3.4.4 Training and Implementation details

We train the model end to end. Ideally, we would like to minimize the negative log likelihood

$$-\log\left(\sum_i \mathrm{P}_\Theta(\mathbf{Y}|m_i,\mathbf{X})\mathrm{P}(m_i|\mathbf{X})\right) \tag{3.7}$$

of the term from from Eqn. 3.5 over all the training data points. However, each training instance only provides the realization of one maneuver class that was actually performed. Thus we minimize the negative log likelihood

$$-\log\left(\mathrm{P}_\Theta(\mathbf{Y}|m_{true},\mathbf{X})\mathrm{P}(m_{true}|\mathbf{X})\right) \tag{3.8}$$

over all training instances, instead.

We train the model using Adam [82] with learning rate 0.001. The encoder LSTM has 64 dimensional state while the decoder has a 128 dimensional state. The sizes of the convolutional social pooling layers are as shown in Fig. 3.3. The fully connected layer for obtaining the vehicle dynamics encoding has size 32. We use the leaky-ReLU activation with α=0.1 for all layers. The model is implemented using PyTorch [127].

3.5 Experimental Evaluation

3.5.1 Dataset

We use the publicly available NGSIM US-101 [30] and I-80 [31] datasets for our experiments. Each dataset consists of trajectories of real freeway traffic captured at 10 Hz over a time span of 45 minutes. Each dataset consists of 15 min segments of mild, moderate and congested traffic conditions. The dataset provides the co-ordinates of vehicles projected to a local co-ordinate system, as defined in section 3.3.1. We split the complete dataset into train and test sets. The test set consists of a fourth of the trajectories from each of the 3 subsets of the US-101 and I-80 datasets. We split the trajectories into segments of 8

s, where we use 3 s of track history and a 5 s prediction horizon. These 8 s segments are sampled at the dataset sampling rate of 10 Hz. However we downsample each segment by a factor of 2 before feeding them to the LSTMs, to reduce the model complexity.

3.5.2 Evaluation metrics

We report results in terms of the root of the mean squared error (RMSE) of the predicted trajectories with respect to the true future trajectories, over a prediction horizon of 5 seconds, as done in [87]. For the LSTM models generating bivariate Gaussian distributions, the means of the Gaussian components are used for RMSE calculation. For models generating multimodal predictive distributions, we use the mode with the highest probability for calculating the RMSE.

While RMSE provides a tangible measure for the predictive accuracy of models, it has limitations while evaluating multimodal predictions. RMSE is skewed in favor of models that average modes. In particular, this average may not represent a good prediction. For example, a driver intending to overtake another vehicle may do so by switching to the immediate left or the immediate right lane, while at the same time accelerating. The average of these two modes would be to accelerate while maintaining lane.

To address this limitation, we additionally report the negative log-likelihood (NLL) of the true trajectories under the predictive distributions generated by the models. While the NLL values cannot be directly interpreted as a physical quantity, they allow us to compare uni-modal and multimodal predictive distributions.

3.5.3 Compared models

We compare the following baselines and system settings:

- Constant Velocity (**CV**): We use a CV Kalman filter as our simplest baseline
- **C-VGMM + VIM:** We use maneuver based variational Gaussian mixture models with a Markov random field based vehicle interaction module described in [35] as

our second baseline. We modify the model to use the maneuver classes described in this work to allow for a fair comparison

- **GAIL-GRU:** We consider the generative adversarial imitation learning model described in [87]. Since the same datasets have been used in both works, we use the results reported by the authors in the original article. There is a caveat that the GAIL-GRU trajectories were generated by running the policy one vehicle at a time, while all surrounding vehicles move according to the ground-truth of the NGSIM dataset. Thus, the model has access to the true trajectories of adjacent vehicles over the prediction horizon.

- Vanilla LSTM (**V-LSTM**): This simply uses the track history of the predicted vehicle in the encoder LSTM and generates a unimodal output distribution with the LSTM decoder

- LSTM with fully connected social pooling (**S-LSTM**): This uses the fully connected social pooling described in [3] and generates a unimodal output distribution

- LSTM with convolutional social pooling (**CS-LSTM**): This uses convolutional social pooling and generates a unimodal output distribution

- LSTM with convolutional social pooling and maneuvers (**CS-LSTM(M)**): This is the complete model described in this chapter, including the maneuver based decoder generating a multimodal predictive distribution

3.5.4 Results

Table 3.1 shows the RMSE and NLL values for the models being compared. S-LSTM, CS-LSTM, and CS-LSTM(M) outperform the baselines [35, 87] in terms of RMSE and NLL values, showing the effectiveness of the proposed model.

Table 3.1. RMSE and negative log-likelihood values over a 5 second prediction horizon

Metric	Horizon (s)	CV	C-VGMM + VIM [35]	GAIL-GRU [87]	V-LSTM	S-LSTM	CS-LSTM	CS-LSTM (M)
RMSE (m)	1	0.73	0.66	0.69	0.68	0.65	**0.61**	0.62
	2	1.78	1.56	1.51	1.65	1.31	**1.27**	1.29
	3	3.13	2.75	2.55	2.91	2.16	**2.09**	2.13
	4	4.78	4.24	3.65	4.46	3.25	**3.10**	3.20
	5	6.68	5.99	4.71	6.27	4.55	**4.37**	4.52
NLL	1	3.72	2.02	-	1.17	1.01	0.89	**0.58**
	2	5.37	3.63	-	2.85	2.49	2.43	**2.14**
	3	6.40	4.62	-	3.80	3.36	3.30	**3.03**
	4	7.16	5.35	-	4.48	4.01	3.97	**3.68**
	5	7.76	5.93	-	4.99	4.54	4.51	**4.22**

We note that the vanilla LSTM and CV models produce higher RMSE values compared to the other models. Each of the other models use some information about the motion of neighboring vehicles. This shows that inter-vehicle interaction is a useful cue for motion prediction, consistent with the results reported in [3, 35, 90].

We also note that CS-LSTM outperforms the S-LSTM in terms of both RMSE and NLL values. This suggests that convolutional social pooling better models the interdependencies of vehicle motion compared to a fully connected social pooling layer. We further analyze this in the following section.

Finally, we note that CS-LSTM(M) leads to higher RMSE values compared to CS-LSTM. This could, in part, be due to misclassified maneuvers, since the RMSE values for CS-LSTM(M) are calculated using the trajectory corresponding to the maneuver with the highest probability. However we note that CS-LSTM(M) achieves significantly lower NLL values compared to CS-LSTM. Thus the predictive distribution generated by CS-LSTM(M) better fits the true trajectories compared to that generated by CS-LSTM. This points to the multimodal nature of the task.

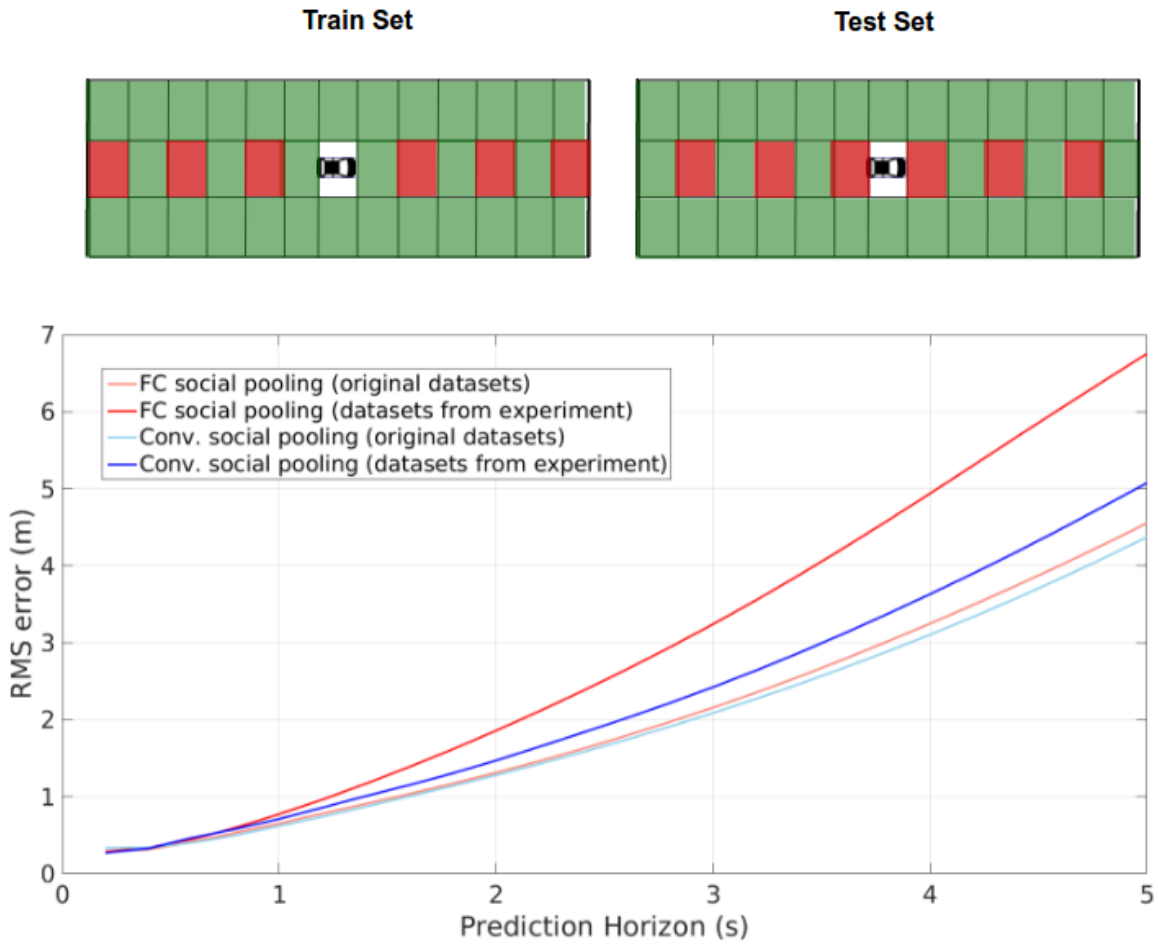


Figure 3.4. Fully connected and convolutional social pooling. *Top:* All training instances with vehicles at odd locations in ego lane of social tensor removed from train set; all instances with vehicles even locations removed from test set. *Bottom:* RMS values of prediction error for FC social pooling and convolutional social pooling for original datasets and datasets from experiment. Convolutional social pooling is more robust to missing spatial patterns in the social tensor

3.5.5 Fully connected vs. convolutional social pooling

We conjectured in section 3.4.2 that fully connected social pooling as described in [3] would poorly generalize to a test set with even slight differences in spatial patterns of agents in the scene as collected in the social tensor, and that convolutional social pooling would remedy this. The reduced prediction error from section 3.5.4 seems to suggest that this is true. However to further analyze this, we set up the following experiment. We remove all instances from the train set corresponding to the odd grid locations of vehicles from the ego lane, and remove all instances from the test set corresponding to even grid locations as shown in Fig. 3.4. Thus, we have a train and test set with zero overlap in terms of spatial configurations of the social tensors. However, we have plenty of spatially similar but not identical configurations common to both. We plot the RMS values of prediction error for this new train and test set, for fully connected social pooling and convolutional social pooling models. We see that the performance of the fully connected social pooling model drastically drops, almost to the point of the vanilla LSTM shown in section 3.5.4. The performance drop with convolutional social pooling is less severe in comparison. This suggests that using convolutional and pooling layers to aggregate social context is a much more robust approach compared to using a fully connected layer.

3.5.6 Qualitative analysis of predictions

In this section we qualitatively analyze the predictions made by our model to gain insights into its behavior in various traffic configurations.

Uni-modal vs. multimodal predictions:

Figure 3.5 shows a comparison of the unimodal predictive distribution generated by CS-LSTM and the multimodal distribution generated by CS-LSTM(M). The plots show three consecutive frames during a lane change maneuver from left to right. The top row shows the track history and the true future trajectory. The middle row shows the

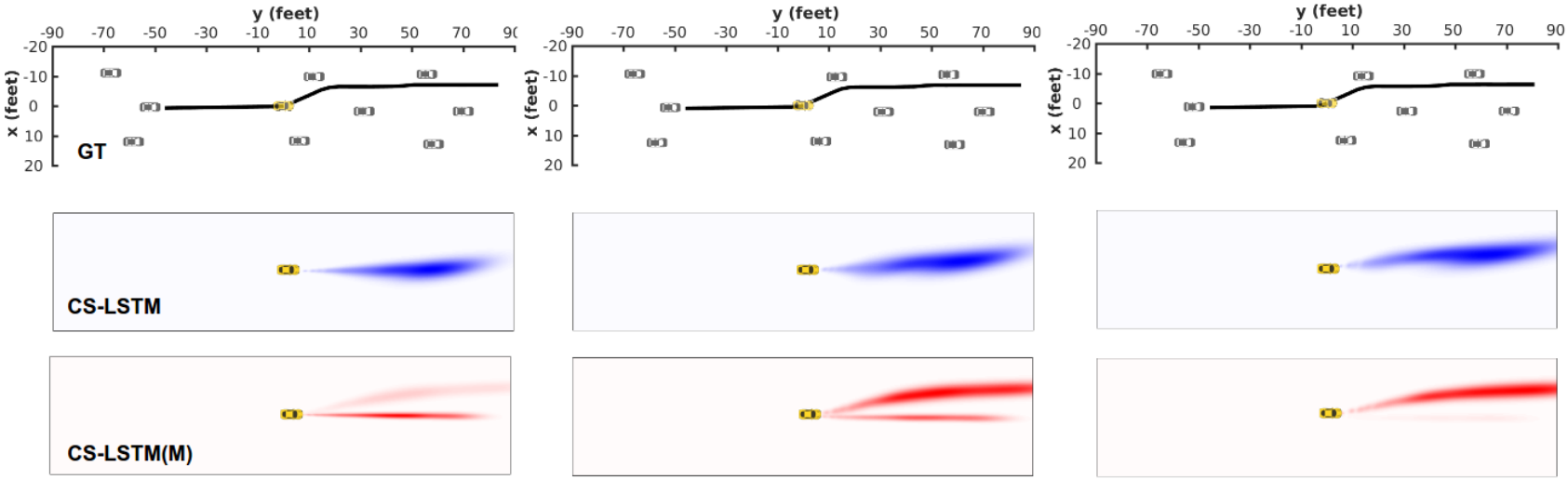


Figure 3.5. Comparison of uni-modal and multimodal predictions: The figure shows the true trajectory (top, black), CS-LSTM predictive distributions (middle, blue) and CS-LSTM(M) predictive distributions (bottom, red) for three consecutive frames of a lane change maneuver. The heat maps are generated by plotting the Gaussian components for each maneuver at each time step in the prediction horizon

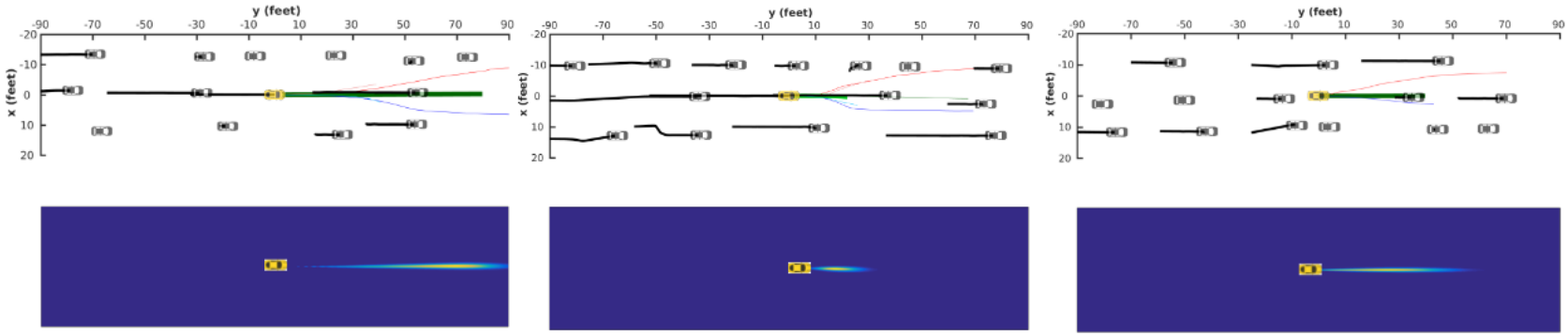
predictive distribution generated by CS-LSTM and the bottom row shows the predictive distribution generated by CS-LSTM(M). We can clearly observe two modes in the predictive distribution of CS-LSTM(M). The mode corresponding to the lane change becomes more and more prominent further into the maneuver while the mode corresponding to the keep lane maneuver fades away. We further note that for all three cases, the mode corresponding to the lane change closely matches the true future trajectory. However, the unimodal distribution generated by CS-LSTM shows an average of the two modes and also has greater variance. This illustrates why the CS-LSTM achieves lower RMSE values while leading to higher NLL values as compared to CS-LSTM(M).

Effect of surrounding vehicles on predictions:

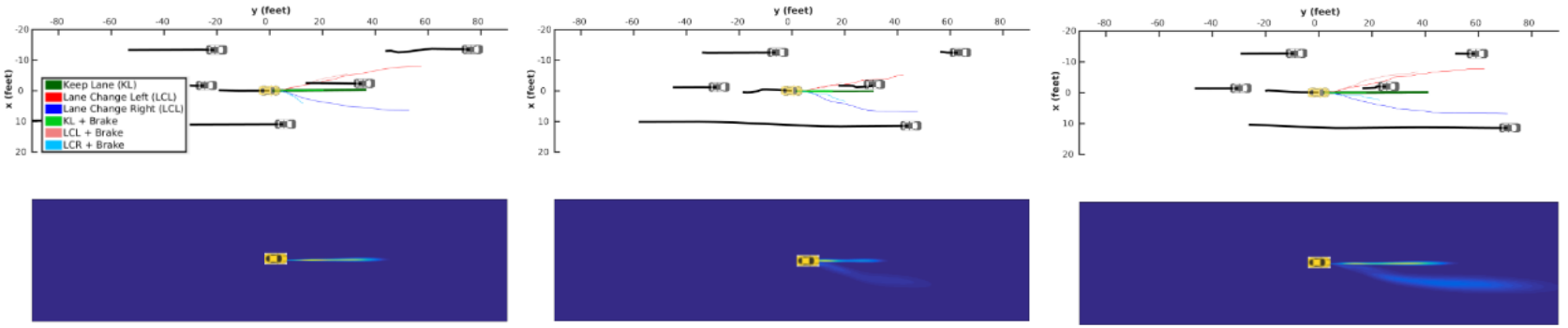
Figure 3.6 shows six different scenarios of traffic. Each figure shows a plot of track histories over the past 3 seconds and the mean predicted trajectories over the next 5 seconds for each maneuver class. The thickness of the plots of the predicted trajectories is proportional to the probabilities assigned to each maneuver class. Additionally, each figure shows a heat map of the complete predicted distribution.

Fig 6(a) shows the effect of the leading vehicle on the predictions made by the model. The first example (top-left) shows an example of free flowing traffic, where the predicted vehicle and the leading vehicle are moving at approximately the same speed. In the second example (top-middle), we note from the track histories that the leading vehicles are slowing down compared to the predicted vehicle. We see that the model predicts the vehicle to brake, although it's current motion suggests otherwise. Conversely, in the third example (top-right), we see that the vehicle being predicted is almost stationary, while the leading vehicles are beginning to move. The model predicts the vehicle to accelerate, as is expected in stop-and-go traffic.

Fig 6(b) shows the effect of vehicles in the adjacent lane on the model's predictions. The three examples show the same scenario separated by 0.5 s. We note that the vehicle



(a) Effect of leading vehicles



(b) Effect of vehicles in adjacent lane

Figure 3.6. Surrounding vehicles affect predictions: This figure shows the effect of surrounding vehicles on predictive distribution generated by the model. The heat maps are generated by plotting the Gaussian components for each maneuver at each time step in the prediction horizon

being predicted is in a congested lane, with its leading vehicle slowing down. We also note that the adjacent left lane is congested. On the other hand, the adjacent right lane is moving at a much faster speed. Based on this, the model assigns a high probability to the predicted vehicle staying in lane and braking, as expected. However, it also assigns a small probability to an overtake by moving to the right lane. We can observe that the model assigns a greater probability to the overtake as the adjacent vehicle moves further away, clearing up the lane.

3.6 Conclusions

In this chater, we presented an LSTM encoder-decoder based model for vehicle trajectory prediction for reasoning about the interdependencies neighboring vehicles' motion. Our model uses an improved social pooling layer using convolutional connections as opposed to fully connected layers that more robustly models and better generalizes the various spatial configurations of interacting agents in a scene. We term this *convolutional social pooling*. Our proposed model outperforms the reported state of the art on two large publicly available datasets of vehicle trajectories. It outputs multimodal distributions for future motion of vehicles based on maneuver classes. Through qualitative analysis we show how modeling the multimodal distribution of future trajectories addresses mode averaging in unimodal predictions. We also show how the predictions are affected by neighboring vehicle motion, modeled implicitly by our convolutional social pooling layers.

Acknowledgements

Chapter 3, in part, is a reprint of the material as it appears in: "Convolutional Social Pooling for Vehicle Trajectory Prediction," Nachiket Deo, and Mohan M. Trivedi, CVPR Workshops 2018. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Trajectory Prediction Conditioned on Grid-based Plans

4.1 Introduction

Chapters 2 and 3 addressed trajectory prediction of surrounding vehicles on multi-lane highways. Highway traffic is a relatively easy setting for trajectory prediction for a few reasons. First, all vehicles of interest have the same direction of motion and there is very little variation in the static scene. This allows us to model vehicle motion in the Frenet frame as described in section 3.3.1. It also allows us to predict future trajectories purely using past trajectories of agents, without encoding the static scene. All agents of interest are vehicles, which follow similar dynamics. Finally, multimodality of future trajectories can be modeled using a small set of maneuver classes.

In the next 2 chapters, we shift our focus to urban environments with mixed traffic consisting of vehicles, pedestrians and bicyclists. Agent motion can no longer be binned into pre-defined maneuver classes. Additionally, the static scene strongly affects the motion of agents. For example, vehicle motion is regulated by lanes, making lane curvature and connectivity important cues for predicting vehicle motion. Pedestrians prefer to walk on sidewalks and crosswalks, making the locations of these scene elements important cues for predicting pedestrian motion. We thus address the problem of predicting the future trajectories of agents, conditioned on their track history and a bird's eye view representation

of the static scene around them. In particular, we wish to forecast trajectories in *unknown scenes*, where prior observations of trajectories are unavailable. This is a challenging task due to a number of factors:

- **Unknown goals and path preferences:** Without prior observations of agent trajectories in a scene, goals and path preferences of agents need to be inferred purely from the scene layout.
- **Scene-compliance:** Predicted trajectories need to conform to the inferred goals and paths in the scene layout. Scene elements such as roads, sidewalks, crosswalks and buildings can be found in a variety of configurations. Thus, there's high variability in the inputs to the trajectory forecasting model.
- **Non-linearity of agent trajectories:** Drivers and pedestrians can make several decisions over long prediction horizons, leading to highly non-linear trajectories. Thus, there's high variability in the outputs of the trajectory forecasting model.
- **Multimodality:** Finally, the distribution of future trajectories is highly multimodal. Unlike highway traffic, multimodality of agent motion in urban scenes cannot be explained away by a small set of maneuvers. In any given scene, an agent can have one of multiple potential goals, with multiple paths to each goal. Regression based approaches lead to mode averaging as shown in [37, 60, 90]. This leads to trajectory forecasts that may not conform to the underlying scene and go off-road.

Recent work has addressed multimodality of the distribution of future trajectories by learning one-to-many mappings from input context to multiple trajectories. Mixture models [25, 32, 37, 38, 67, 109, 149, 199] assign a mixture component to each mode of the trajectory distribution. They output mean trajectories and probabilities for each mixture component, along with prediction uncertainty. Alternatively, conditional generative models [14, 15, 60, 90, 147, 148, 154, 195] map input context and a sample from a simple

latent distribution to a trajectory output. They can be sampled from indefinitely, to output multiple trajectories. Both, conditional generative models and mixture models need to learn a mapping from a high dimensional input space (variable scene and agent configuration) to a high dimensional output space (continuous valued trajectories).

Several recent works thus incorporate inductive bias into the predicted modes by conditioning on agent goals [105, 106, 194], lane center-lines [27, 101, 191] or anchor trajectories obtained by clustering a training dataset of trajectories [26,129]. However, over long prediction horizons, agents can take multiple paths to the same goal location, and can change lanes. Finally anchor trajectories learned by clustering a training dataset may not generalize to novel scene configurations not seen in the training set. We thus need an approach to add inductive bias to the model that generalizes to novel scene configurations and accounts for variable paths that agents can follow.

A completely different approach to motion prediction can be found in [84, 183, 192, 198] pioneered by Ziebart *et al.* [198]. Agents are modeled as Markov decision processes (MDPs) exploring a 2-D grid defined over the scene. A reward map for the MDP is learned via maximum-entropy inverse reinforcement learning (MaxEnt IRL) [197]. MDPs are naturally suited to model the agent's sequential decision making. Additionally, since the reward is learned from local scene cues at each grid location, it can be transferred to unknown scenes with a different configuration of scene elements. However, MaxEnt IRL approaches suffer from two limitations: First, they require a pre-defined absorbing goal state, limiting them to applications where goals of agents are known beforehand. As opposed to this, we need to *infer* goals of agents. Second, they only provide future locations of the agent in the grid, without mapping them to specific times [1]. This does not take into account the agent's dynamics.

[1]We refer to agent locations without assigned times as *paths*, and agent locations with assigned times as *trajectories*

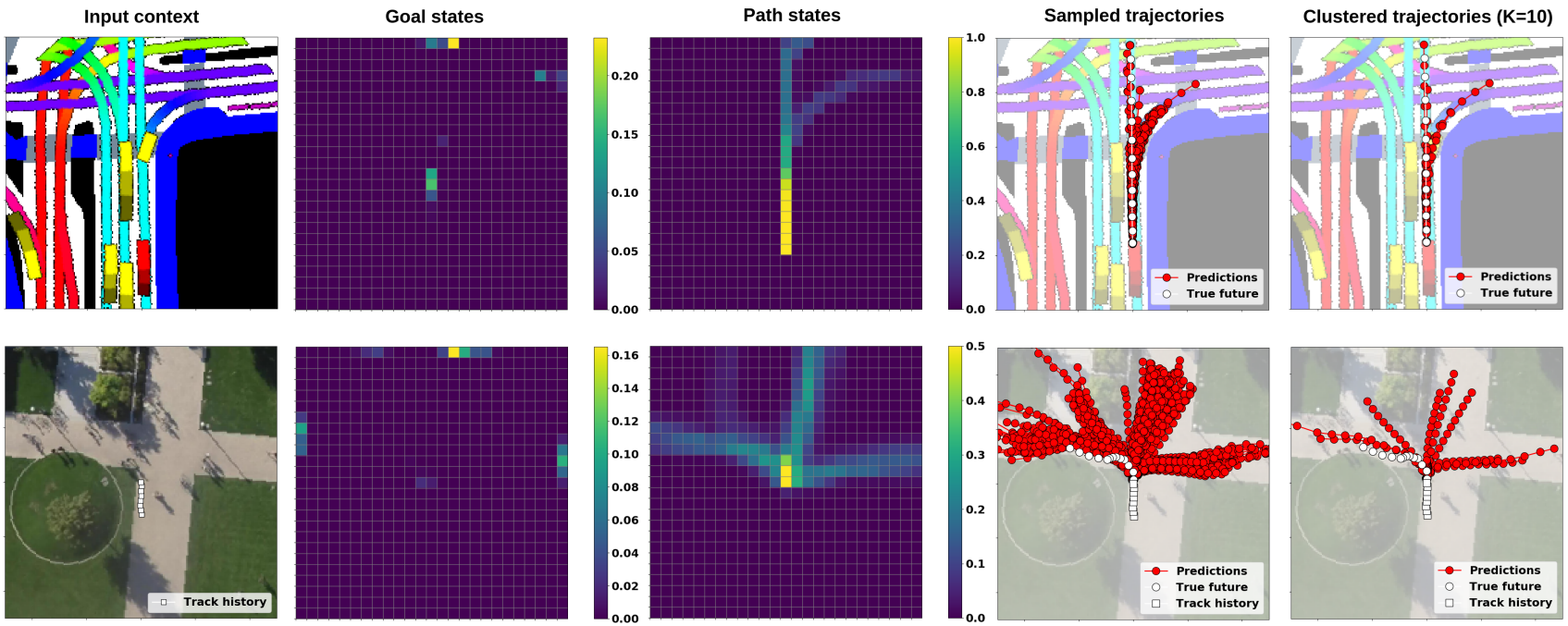


Figure 4.1. Forecasts generated by P2T: We address the problem of forecasting agent trajectories in unknown scenes. The inputs to our model *(left)* are snippets of the agents' past trajectories, and a bird's eye view representation of the scene around them. Our model infers potential goals of the agents *(left-middle)* and paths to these goals *(middle)* over a coarse 2-D grid defined over the scene by modeling the agent as a MaxEnt policy exploring the grid. It generates continuous valued trajectories conditioned on the grid-based plans sampled from the policy *(middle-right)*. Finally it outputs K predicted trajectories by clustering the sampled trajectories *(right)*.

4.1.1 Contributions

We seek to leverage the transferability of grid based MaxEnt IRL approaches, while allowing for sampling of continuous valued trajectories similar to conditional generative models. We present *P2T* (Plans-to-Trajectories), a planning based approach to generate long-term trajectory forecasts in unknown scenes. Our approach relies on two key ideas.

1. **Joint inference of goals and paths by learning rewards:** We reformulate the maximum entropy inverse reinforcement learning framework to learn transient path state rewards and terminal goal state rewards. Our reformulation allows for joint inference of goals, and paths to goals. This alleviates the need for a pre-defined absorbing goal state in the original formulation [197].

2. **Trajectories conditioned on plans:** We refer to state sequences sampled from the MaxEnt policy as *plans*. We propose an attention based trajectory generator that outputs continuous valued trajectories conditioned on sampled plans, rather than a latent variable. Compared to conditional generative models, our model outputs trajectories that better conform to the underlying scene over longer prediction horizons. Additionally, the state sequences of the MaxEnt policy allow for better interpretability compared to the latent space of a conditional generative model

We evaluate our model on two publicly available trajectory datasets: the Stanford drone dataset [150] (SDD) consisting of pedestrians, bicyclists, skateboarders and slow moving vehicles at various locations on a university campus, and the NuScenes dataset [21] consisting of vehicles navigating complex urban traffic. We report results in terms of minimum over K average displacement error (MinADE_K), final displacement error (MinFDE_K) and miss rate (MR_K) metrics reported in prior work [27, 60, 90, 148, 154, 195], as well as sample quality metrics such as off-road rate [117] and off-yaw rate [59]. Our model achieves state of the art results on several metrics, while being competitive on others. In particu-

lar, it significantly outperforms existing approaches in terms of sample quality metrics, forecasting trajectories that are both diverse as well as precise. Figure 4.1 shows forecasts generated by P2T for two example scenarios from the NuScenes and Standord drone datasets. We make our code publicly available at https://github.com/nachiket92/P2T.

4.2 Preliminaries

In this section, we briefly review maximum entropy inverse reinforcement learning (MaxEnt IRL) for path forecasting, conditioned on pre-defined goal states [84, 183, 198].

MDP formulation: We consider a Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$, for a finite horizon setting with N steps. $\mathcal{S}$ is the state space consisting of cells in a 2-D grid defined over the scene. $\mathcal{A}$ is the action space consisting of 4 discrete actions, {*up, down, left, right*}, to move to adjacent cells. We assume deterministic dynamics, where $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function. Finally, $r : \mathcal{S} \rightarrow \mathbb{R}_0^-$ is the reward function mapping each state to a real value less than or equal to 0. We assume that the initial state s_{init} and the goal state s_{goal} of the MDP are known.

MaxEnt IRL objective: Under the maximum entropy distribution, the probability of observing a state action sequence $\tau = \{(s_1, a_1), (s_2, a_2), \ldots (s_N, a_N)\}$ is proportional to the exponential of its reward.

$$P(\tau) = \frac{1}{Z}\exp\left(r(\tau)\right) = \frac{1}{Z}\exp\left(\sum_{i=1}^{N} r(s_i)\right), \tag{1}$$

where Z the normalizing constant. MaxEnt IRL involves learning a reward function $r_\theta(s)$ parametrized by a set of parameters θ, operating on a set of features extracted for each state s. The objective is to learn a reward function that maximizes the log likelihood of observing a training set of demonstrations T = $\{\tau_1, \tau_2, \ldots \tau_K\}$

$$\arg\max_{\theta} \mathcal{L}_{\theta} = \arg\max_{\theta} \sum_{\tau \in \mathrm{T}} \log \left(\frac{1}{Z_{\theta}} \exp(r_{\theta}(\tau)) \right). \tag{2}$$

This can be solved using stochastic gradient descent, with the gradient of the log likelihood $\mathcal{L}_{\theta}$ simplifying to

$$\frac{d\mathcal{L}_{\theta}}{d\theta} = \sum_{\tau \in \mathrm{T}} (D_{\tau} - D_{\theta}) \frac{dr_{\theta}}{d\theta}, \tag{3}$$

where, D_{τ} are the state visitation frequencies (SVFs) for the training demonstration τ and D_{θ} are the expected SVFs for the MaxEnt policy given the current set of reward parameters θ. If a deep neural network is used to model the reward function $r_{\theta}(s)$, $\frac{dr_{\theta}}{d\theta}$ can be obtained using backpropagation as described in [182]. D_{θ} is obtained using Algorithm 1 and Algorithm 2.

Approximate value iteration: Algorithm 1 involves solving for the MaxEnt policy π_{θ}, given the current reward function r_{θ}, and the goal state s_{goal}. π_{θ} represents the probability of taking action a, given state s. The policy can be stationary, *ie.*, independent of the time step $\pi_{\theta}(a|s)$, or non-stationary $\pi_{\theta}^{(n)}(a|s)$. We use a non-stationary policy as used in [93, 196]. Algorithm 1 involves iterative updates of the state and action log partition functions $V(s)$ and $Q(s, a)$. These can be interpreted as soft estimates of the expected future reward given state s and the expected future reward given state-action pair (s, a) respectively. $V(s)$ is initialized to 0 for s_{goal} and $-\infty$ for all other states. $V(s)$ and $Q(s, a)$ are then iteratively updated over N steps, while holding $V(s_{goal})$ fixed at 0. For each step, π_{θ} is given by

$$\pi_{\theta}^{(n)}(a|s) = \exp\left(Q^{(n)}(s, a) - V^{(n)}(s)\right). \tag{4}$$

Holding $V(s_{goal})$ fixed to 0, while initializing all other $V(s)$ values to $-\infty$ ensures that the MDP ends at s_{goal}.

Policy propagation: Algorithm 2 involves calculating the SVFs. It involves repeatedly

applying π_θ for N steps, starting with the initial state distribution, to give SVF at each step. The SVF corresponding to the goal state is set to 0 at each step, since the goal state absorbs any probability mass that reaches it. The expected SVF D_θ is obtained by summing the SVFs over the N steps.

Algorithm 1. Approx. value iteration (goal conditioned)

Inputs: $r_\theta,\ s_{goal}$

$V^{(N)}(s) \leftarrow -\infty,\ \forall s \in \mathcal{S}$

for $n = N, ..., 2, 1$ **do**

  $V^{(n)}(s_{goal}) \leftarrow 0$

  $Q^{(n)}(s,a) = r_\theta(s) + V^{(n)}(s'), \qquad s' = \mathcal{T}(s,a)$

  $V^{(n-1)}(s) = \text{logsumexp}_a\ Q^{(n)}(s,a)$

  $\pi_\theta^{(n)}(a|s) = \exp\left(Q^{(n)}(s,a) - V^{(n)}(s)\right)$

end for

Algorithm 2. Policy propagation (goal conditioned)

Inputs: $\pi_\theta,\ s_{init},\ s_{goal}$

$D^{(1)}(s) \leftarrow 0,\ \forall s \in \mathcal{S}$

$D^{(1)}(s_{init}) \leftarrow 1$

for $n = 1, 2..., N$ **do**

  $D^{(n)}(s_{goal}) \leftarrow 0$

  $D^{(n+1)}(s) = \sum_{s',a} \pi_\theta^{(n)}(a|s') D^{(n)}(s'), \quad \mathcal{T}(s',a) = s$

end for

$D(s) = \sum_n D^{(n)}(s)$

Path forecasting conditioned on goals: The MaxEnt policy π_θ^*, for the converged reward model r_θ, can be sampled from, to give path forecasts on the 2-D grid from the s_{init} to s_{goal}. Since π_θ^* is stochastic, the policy can explore multiple paths within the scene to the goal state. However, for most cases of pedestrian or vehicle trajectory forecasting, s_{goal}

is unknown, and needs to be inferred. Additionally, sampling π^*_θ only provides future paths, without mapping them to specific times. A step for the MDP need not correspond to a fixed time interval. Different agents can have different speeds. Agents can also accelerate or decelerate over the course of the 10s prediction horizon.

4.3 Proposed Approach

We leverage the transferability of grid based MaxEnt IRL, while not requiring knowledge of s_{goal}, and generate continuous valued trajectories, mapped to specific times in the future. Figure 4.2 provides an overview of P2T, our proposed approach. P2T consists of three components.

The first component is a reward model, comprised by convolutional and pooling layers. At each cell on a coarse 2-D grid, the reward model maps local scene context and motion features capturing the agent's track history, to a transient path state reward and a terminal goal state reward. Section 4.3.2 describes the reward model in greater detail.

The next component is a MaxEnt policy independent of pre-defined goal states. We reformulate MaxEnt IRL to allow for inference of goal and path states, given the path and goal rewards learned by the reward model (see section 4.3.1). We obtain a single policy that can be sampled to generate paths to different plausible goals on the 2-D grid. We refer to each state sequence sampled from the policy as a *plan*.

The final component of P2T is an attention based trajectory generator, that outputs continuous valued trajectories conditioned on the sampled plans. The trajectory generator encodes the track history of the agent using a gated recurrent unit (GRU), and the sampled plans using a bidirectional GRU (BiGRU). Finally, a GRU decoder equipped with soft-attention [6], attends to the plan encoding to output trajectories over the prediction horizon. Section 4.3.3 describes the trajectory generator in greater detail.

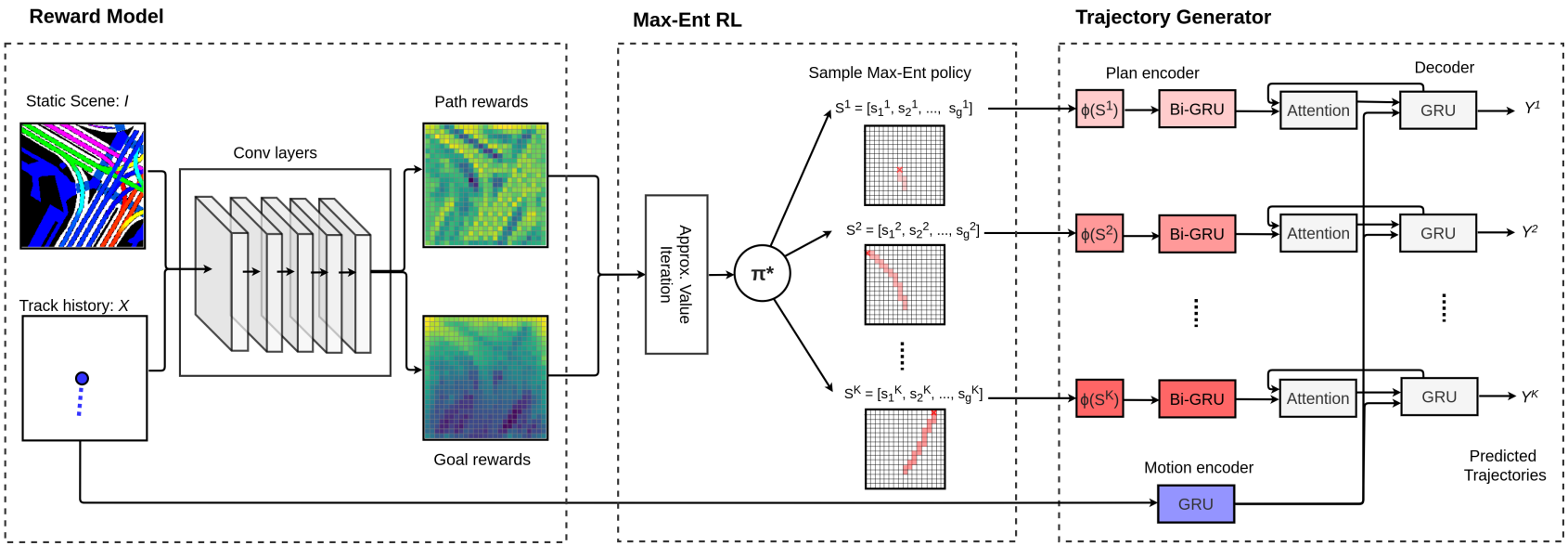


Figure 4.2. P2T: P2T consists of three modules: (1) a fully convolutional reward model, that outputs transient path state rewards and terminal goal state rewards on a coarse 2-D grid, (2) a MaxEnt RL policy for the learned path and state rewards, that can be sampled to generate multimodal plans on the 2-D grid, and (3) an attention based trajectory generator, that outputs continuous valued trajectories conditioned on the sampled plans.

4.3.1 Inferring goals and paths by learning rewards

We wish to relax the requirement of prior knowledge of s_{goal} in MaxEnt IRL. Certain locations in a scene are likelier to be goals of agents. For pedestrians, these can be points where paths and sidewalks exit the scene, entrances to buildings, or parked cars. For vehicles, these can be points where lanes exit the scene, stop signs or parking lots. Goals are also likelier to be along the direction of the agent's motion. Rather than always terminating at a predefined goal, we would like our policy to induce a distribution over possible goal states. This would allow us to sample paths from the policy terminating at different goals in the scene. We propose to do this by learning path and goal state rewards, conditioned on the scene and past motion of the agent, and learning a policy unconstrained by s_{goal}. We reformulate the MDP and modify the approximate value iteration algorithm.

MDP formulation:

- **State space:** Potentially any cell location on the 2-D grid could be the goal of the agent, or a point on their future path. We define the state space $\mathcal{S} = \{\mathcal{S}_p, \mathcal{S}_g\}$. $\mathcal{S}_p$ is the set of path states and $\mathcal{S}_g$ is the set of goal states. Each cell location on the 2-D grid has an associated path state belonging to $\mathcal{S}_p$ and a goal state belonging to $\mathcal{S}_g$. The policy terminates on reaching any goal state.

- **Action space:** $\mathcal{A} = \{$*up, down, left, right, end*$\}$. The *up*, *down*, *left* and *right* actions allow transitions from path states to adjacent path states. Additionally, we define an *end* action that transitions the MDP from a path state to the goal state at the same cell location.

- **Transition function:** $\mathcal{T} : \mathcal{S}_p \times \mathcal{A} \rightarrow \mathcal{S}$ maps path state and action pairs to other path states and goal states. Since goal states are terminal, the MDP has no transitions out of a goal state.

- **Rewards:** We learn two functions, r_{p_θ} corresponding to path rewards, and r_{g_θ} corresponding to goal rewards.

Approximate value iteration with inferred goals:

Algorithm 3 depicts our modified approximate value iteration, unconstrained on s_{goal}. Unlike algorithm 1, we do not hold the $V(s_{goal})$ fixed at 0 to enforce goal directed behavior. Instead, we use r_{g_θ} to learn a policy that induces a multimodal distribution over potential goal states. The inputs to algorithm 3 are the learned rewards r_{g_θ} and r_{p_θ}. We initialize $V(s)$ to $-\infty$ for all path states $\mathcal{S}_p$. This is because we want the MDP to end up at some goal state within the N step finite horizon. Since the goal states are terminal, the MDP receives the goal rewards only once. We thus hold $V(s)$ fixed to $r_{g_\theta}(s)$ for all goal states $\mathcal{S}_g$. We then iteratively update the state-action log partition function $Q^{(n)}(s, a)$ and the state log partition function $V^{(n)}(s)$ for path states $\mathcal{S}_p$ over N steps. At each step, the MaxEnt policy is obtained by taking the ratio of the exponent of $Q^{(n)}(s, a)$ and $V^{(n)}(s)$.

Policy propagation with inferred goals:

Algorithm 4 depicts policy propagation independent of s_{goal}. This is almost identical to algorithm 2. The only difference is, we do not set the goal state SVFs to 0, as in line 4 of algorithm 2. This is because we use the goal SVFs to train the reward model for r_{g_θ}, using equation (3). We use a frame of reference centered at the agent's location at the time of prediction. Thus, s_{init} is always the path state at the center of the grid.

Algorithm 3. Approx. value iteration (inferred goals)

Inputs: r_{g_θ}, r_{p_θ}

$V^{(N)}(s) \leftarrow -\infty,\ \forall s \in \mathcal{S}_p$

for $n = N, ..., 2, 1$ **do**

  $V^{(n)}(s) \leftarrow r_{g_\theta}(s),\ \forall s \in \mathcal{S}_g$

  $Q^{(n)}(s,a) = r_{p_\theta}(s) + V^{(n)}(s'),\ \forall s \in \mathcal{S}_p,\ s' = \mathcal{T}(s,a)$

  $V^{(n-1)}(s) = \text{logsumexp}_a\ Q^{(n)}(s,a),\ \forall s \in \mathcal{S}_p$

  $\pi_\theta^{(n)}(a|s) = \exp\left(Q^{(n)}(s,a) - V^{(n)}(s)\right)$

end for

Algorithm 4. Policy propagation (inferred goals)

Inputs: π_θ, s_{init}

$D^{(1)}(s) \leftarrow 0,\ \forall s \in \mathcal{S}$

$D^{(1)}(s_{init}) \leftarrow 1$

for $n = 1, 2..., N$ **do**

  $D^{(n+1)}(s) = \sum_{s',a} \pi_\theta^{(n)}(a|s') D^{(n)}(s'), \quad \mathcal{T}(s',a) = s$

end for

$D(s) = \sum_n D^{(n)}(s)$

4.3.2 Reward model

We define a reward model consisting purely of convolutional and pooling layers. This allows us to learn a mapping from local patches of the scene to path and goal rewards. The equivariance of the convolutional layers allows the reward model to be transferred to novel scenes with a different configuration of scene elements. Figure 4.3 shows our reward model. It consists of three sets of convolutional layers.

CNN_{feat} serves as a scene feature extractor, operating on the birds eye view

representation I of the static scene around the agent:

$$\phi_I = \text{CNN}_{feat}\left(I\right). \tag{5}$$

The spatial dimensions of the scene features ϕ_I equal the size of the 2-D grid corresponding to our state space $\mathcal{S}$. In addition to scene features, we want our goal and path rewards to depend on the past motion of the agent. Thus, similar to [192], we concatenate the scene features with feature maps encoding the agent's motion, and the locations of the grid cells:

$$\phi_M = \left[|v|, x, y\right]. \tag{6}$$

Here, $|v|$ is the speed of the agent. This value is replicated over the entire feature map. x and y are the locations of each grid cell in the agent-centric frame of reference, with the origin at the agent's current location and the x-axis aligned along the agent's current direction of motion. CNN_p and CNN_g map the scene and motion features to path and goal rewards respectively:

$$r_{p_\theta} = \text{CNN}_p\left(\phi_I, \phi_M\right). \tag{7}$$

$$r_{g_\theta} = \text{CNN}_g\left(\phi_I, \phi_M\right). \tag{8}$$

Implementation details:

We represent the scene as a 200×200 bird's eye view image around the agent. CNN_{feat} consists of the first two ImageNet pretrained blocks of ResNet34 [63]. This downsamples the spatial dimension of the feature maps to 50×50. This is followed by a 2×2 convolutional layer with depth 32 and stride 2, to aggregate context at each cell location. This gives 32 scene feature maps over a 25×25 grid. CNN_p and CNN_g have identical architectures consisting of two 1×1 convolutional layers. The first layer has depth 32, and the second layer has depth 1 to give a single path or goal reward value

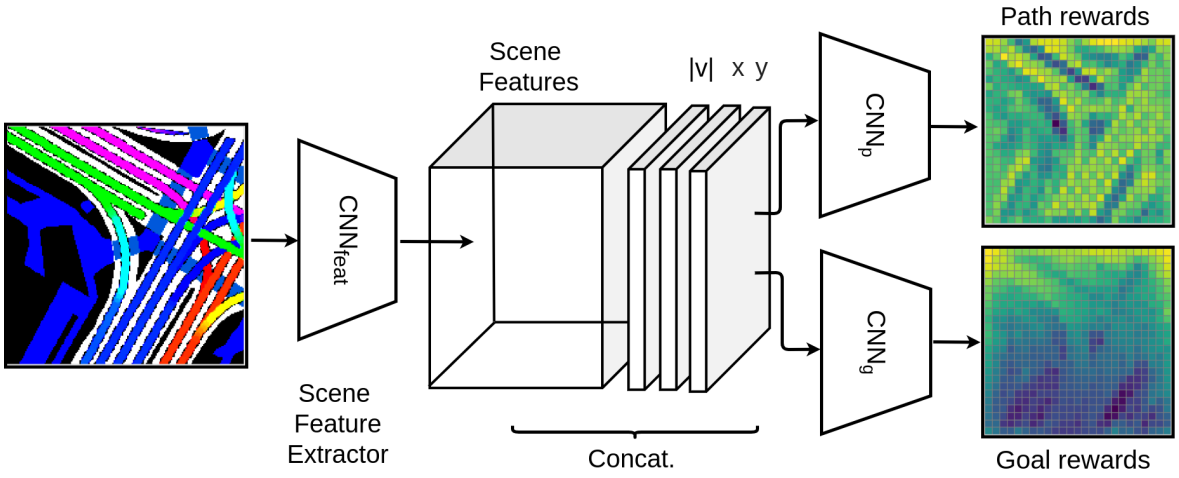


Figure 4.3. Reward model: CNN_{feat} extracts features from the static scene. We concatenate these with feature maps capturing the agent's motion. CNN_p and CNN_g learn path and goal rewards from the features.

at each cell. We apply the log-sigmoid activation at the outputs of CNN_p and CNN_g to restrict reward values between $-\infty$ and 0.

Training:

The reward model is trained to maximize the log-likelihood $\mathcal{L}_\theta$ of agent paths in the train set shown in equation (2), with gradients given by equation (3). The state visitation frequencies D_θ for both path and goal states are obtained using algorithms 3 and 4. We use Adam [82] with learning rate 0.0001 to train the reward model.

4.3.3 Trajectories conditioned on plans

Consider the optimal MaxEnt policy π_θ^* obtained using algorithm 3 for the converged reward model. Consider state sequences or *plans* sampled from π_θ^*, with the i^{th} plan given by

$$s^{(i)} = \left[s_1^{(i)}, s_2^{(i)}, \ldots, s_N^{(i)} \right]. \tag{9}$$

We expect the sampled plans to end at a diverse set of goal states, and explore various paths to these goals. Additionally, each plan $S^{(i)}$ can be expected to conform to the

underlying scene and model the agent's sequential decision making. However, the plans by themselves do not capture the dynamics of the agent's motion. A fast moving agent can make more progress along a plan compared to a slow moving agent over a fixed prediction horizon T_f. The dynamics of the agent's motion can be estimated using a snippet of their most recent track history, over time T_h,

$$x = [x_{-T_h}, \ldots, x_1, x_0], \tag{10}$$

where the x_t's correspond to past location, velocity, acceleration and yaw-rate of the agent, with the subscript t representing time and $t = 0$ the prediction instant.

We thus seek a model that, for each sampled plan $s^{(i)}$, and track history x, generates a continuous valued trajectory $y^{(i)}$ over a prediction horizon T_f,

$$y^{(i)} = \left[y_1^{(i)}, y_2^{(i)}, \ldots, y_{T_f}^{(i)}\right], \tag{11}$$

where y_t is the future location of the agent at time t. We propose a trajectory generator modeled as a recurrent neural network encoder-decoder, equipped with soft attention [6]. The trajectory generator has the following components.

- **Motion encoder:** We encode the track history x using a GRU encoder, where the state of the GRU at time t is given by

$$h_{m_t} = \mathrm{GRU}_m\left(h_{m_{t-1}}, e_x\left(x_t\right)\right). \tag{12}$$

  Here $e_x(\cdot)$ is a fully connected embedding layer. The GRU state at the prediction instant, h_{m_0}, can be expected to encode the motion of the agent.

- **Plan encoder:** The plan encoder (Fig. 4.4) encodes local scene features, nearby agent states and location co-ordinates along sampled state sequences. The plan

encoding serves as an inductive bias for the decoder to output trajectories that conform to the paths and goals inferred by the policy. For scene features, we use the outputs ϕ_I of CNN_{feat} from the reward model (equation 5). For surrounding agent states, we populate their grid locations with the agents' velocity, acceleration and yaw-rate. For each state $s_n^{(i)}$ in a sampled plan $s^{(i)}$, we embed the scene features, agent states and location co-ordinates at the grid cell corresponding to $s_n^{(i)}$, using fully connected layers and concatenate the outputs to give the state encoding $\phi_s\left(s_n^{(i)}\right)$. We use a bidirectional GRU (BiGRU) encoder to aggregate the state encodings over the entire plan. The state of the BiGRU at step n is given by

$$h_{s_n}^{(i)} = \text{BiGRU}_s\left(h_{s_{n-1}}^{(i)}, h_{s_{n+1}}^{(i)}, \phi_s\left(s_n^{(i)}\right)\right). \tag{13}$$

- **Attention based decoder:** We use a GRU decoder equipped with a soft attention module to generate the output trajectories $y^{(i)}$. Our core idea is to allow the decoder to attend to specific states of the sampled plan $s^{(i)}$ as it generates trajectories along the plan. Thus, the decoder can attend to just the first few states of sampled plans, as it generates the future trajectories for a slow moving agent. On the other hand, it can attend to later states while generating a fast moving agent's trajectories.

  We initialize the state of the decoder using the final state of the motion encoder,

$$h_{dec_1} = h_{m_0}. \tag{14}$$

  This provides the decoder a representation of the agent's motion. The decoder state is then updated over the prediction horizon, with the outputs at each time-step giving the predicted locations.

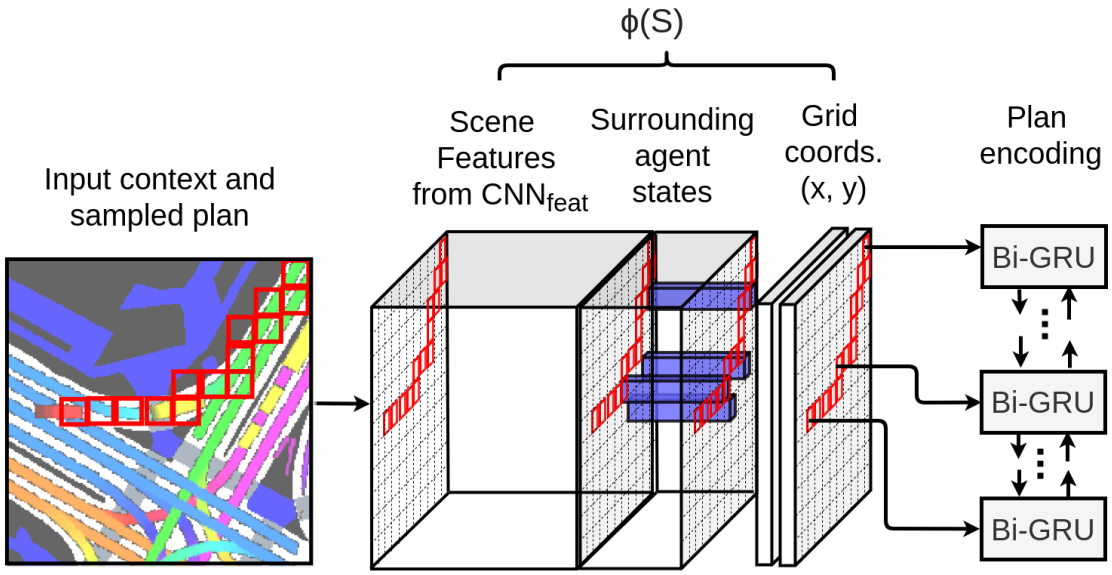


Figure 4.4. Plan encoder: For each state in a sampled plan, we encode the scene features, surrounding agent states and the location co-ordinates of the grid cell and term it $\phi_S(s)$. This is then fed into bidirectional GRU to encode the the entire sampled plan. Our GRU decoder generates output trajectories by attending to the plan encoding.

$$h^{(i)}_{dec_t} = \text{GRU}_{dec}\left(h^{(i)}_{dec_{t-1}}, \text{Att}\left(h^{(i)}_{dec_{t-1}}, h^{(i)}_{s_{1:N}}\right)\right), \tag{15}$$

$$y^{(i)}_t = o_y(h^{(i)}_{dec_t}), \tag{16}$$

where $\text{Att}(\cdot)$ is the attention module and $o_y(\cdot)$ is a fully connected layer operating on the decoder states.

Sampling and clustering:

The trajectory generator outputs a trajectory conditioned on each sampled plan. This allows us to indefinitely sample trajectories from our model. Since the MaxEnt policy induces a multimodal distribution over path and goal states, the sampled trajectories also represent a multimodal predictive distribution. However, sampling by itself can be inefficient, with several sampled state sequences and trajectories being identical or very

similar. In order to provide downstream path planners with a succinct representation of the trajectory distribution, we cluster the sampled trajectories using the K-means algorithm to output a set of K predicted trajectories. The number of clusters K can be varied as required by the downstream path planner, without having to re-train the model.

Implementation details:

As per the standard benchmarks for both datasets, we use track history of 3.2 seconds and a prediction horizon of 4.8 seconds SDD [150], while a track history of 2 seconds and a prediction horizon of 6 seconds for NuScenes [21]. We assume an agent centric frame of reference with the x-axis aligned along the agent's direction of motion at $t = 0$. We use a 32 sized state vector for each of the GRUs. The motion encoder uses an embedding layer of size 16, while the plan encoder uses embedding layers of size 16, 32 and 16 for grid locations, scene features and surrounding agent states respectively. Our attention module is a multi-layer perceptron (MLP) with one hidden layer of size 32.

Training:

To train the model, we sample 200 plans and corresponding trajectories from the trajectory generator and cluster them to give K output trajectories $\left[y^{(1)}, y^{(2)}, \ldots, y^{(K)}\right]$. We use $K = 10$ for NuScenes and $K = 20$ for SDD. We minimize the minimum over K average displacement error (MinADE_K) over the training set.

$$\text{MinADE}_K = \min_{i \in \{1,\ldots,K\}} \frac{1}{T_f} \sum_{t=1}^{T_f} \left\| y_t^{GT} - y_t^{(i)} \right\|_2, \tag{17}$$

where y^{GT} is the ground truth future trajectory of the agent. The MinADE_K loss has been used in prior work for training models for multimodal trajectory forecasting [32,60,154]. For a model generating multiple trajectories, it avoids penalizing plausible future trajectories that do not correspond to the ground truth. To speed up convergence, we pre-train

the model to minimize the average displacement error between y^{GT} and the trajectory predicted by the model conditioned on the ground truth plan of the agent $y^{S_{GT}}$. We use Adam, with learning rate 0.0001 for training the trajectory generator

4.4 Experimental Evaluation

4.4.1 Datasets

Stanford drone dataset: The Stanford drone dataset (SDD) [150] consists of trajectories of pedestrians, bicyclists, skateboarders and vehicles captured using drones at 60 different scenes on the Stanford university campus. The dataset provides bird's eye view images of the scenes, and locations of tracked agents in the scene's pixel co-ordinates. The dataset contains a diverse set of scene elements like roads, sidewalks, walkways, buildings, parking lots, terrain and foliage. The roads and walkways have different configurations, including roundabouts and four-way intersections. We use the dataset split defined in the TrajNet benchmark [153] and used in prior work [14,154,195], for defining our train, validation and test sets. The dataset is split based on scenes. Thus, the train, validation and test sets all have different scenes from the 60 total scenes. This allows us to evaluate our model on unknown scenes where it hasn't seen prior trajectory data. Note that we consider *all trajectories* in the train, validation and test scenes of SDD as per [14, 16, 154, 195]. Subsequent work [105, 106] has reported results on a subset of trajectories primarily consisting of pedestrians. We report results on this split separately.

NuScenes: The NuScenes dataset [21] comprises 1000 scenes, each of which is a 20 second record, capturing complex urban traffic over a wide variety of road layouts and lane structures. The dataset was captured using vehicle mounted camera and lidar sensors while driving through Boston and Singapore, and contains hand annotated vehicle detection boxes and tracks at a 2 Hz. In particular, we train and evaluate our model using the official benchmark split for the NuScenes prediction challenge consisting of vehicle trajectories.

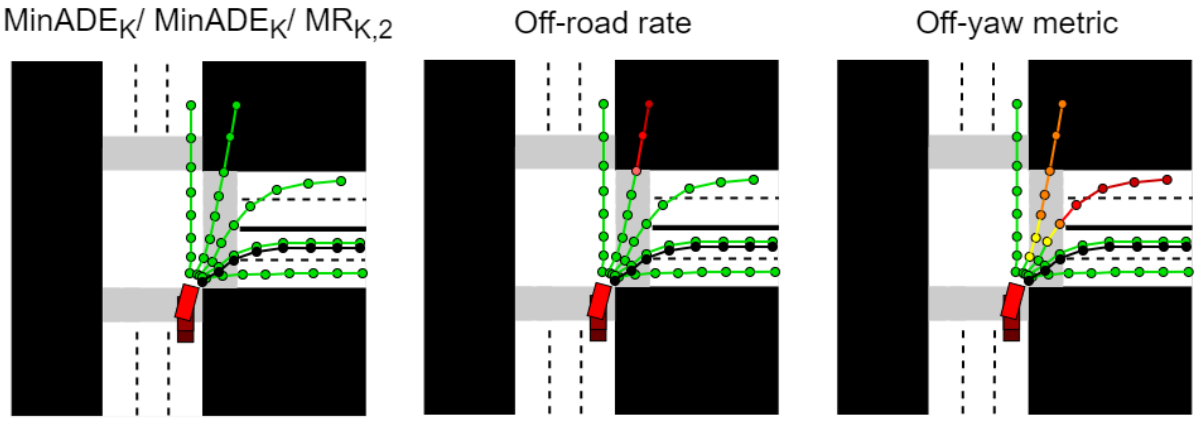


Figure 4.5. Sample quality metrics. MinADE$_K$, MinFDE$_K$ and miss rate fail to penalize a diverse set of trajectories that don't conform to the scene (left). The off-road rate (middle) and off-yaw (right) metrics address this by penalizing predicted points that fall off the drivable area or onto oncoming traffic. Warm colors indicate higher errors.

In addition to trajectories, NuScenes provides high definition bird's eye view maps of the scene, including drivable area masks, cross-walks, side-walks and lane center-lines along with their connectivity and directions. We use a 50m $\times$ 50m crop of the HD map around the vehicle of interest as the scene representation for our model. It extends 40m along the agent's direction of motion, 10m behind and $\pm$ 25m laterally.

4.4.2 Metrics

Deviation from ground-truth: For evaluating a trajectory forecasting model, we need a metric for how much the forecasts deviate from the ground truth future trajectory. However, since our model generates forecasts from a multimodal distribution, we need a metric that does not penalize plausible trajectories generated by the model that don't correspond to the ground truth. Thus, we use the minimum of K average displacement error (MinADE$_K$), final displacement error (MinFDE$_K$) and miss rate within 2 meters (MR$_{K,2}$) as metrics, as utilized in prior work on multimodal trajectory forecasting [14, 27, 60, 90, 148, 154, 195]. MinADE$_K$ (eq. 18) computes the average prediction error in terms of L2 norm between the ground truth future trajectory, and the forecast trajectory closest to it. MinFDE$_K$

is similar to MinADE_K, but only considers the prediction error for the final predicted location. Finally, a set of K predictions is considered a missed prediction if none of the K trajectories are within 2 meters of the ground truth over the entire prediction horizon. $\text{MR}_{K,2}$ computes the fraction of missed predictions in the test set.

Sample quality metrics: While MinADE_K, MinFDE_K and $\text{MR}_{K,2}$ avoid penalizing plausible future trajectories that don't conform to the ground truth, they also do not penalize implausible future trajectories as long as one of the K trajectories is close to the ground truth. Thus a model that generates a very diverse set of K trajectories by random guessing can achieve low MinADE_K, MinFDE_K and $\text{MR}_{K,2}$ values, even if the trajectories do not conform to the underlying scene. Thus, while these metrics serve as good measures of the 'recall' of the model for the multimodal predictive distribution, they serve as poor measures for the model's 'precision'. We refer readers to Rhinehart *et al.* [147] for a detailed discussion on the diversity-precision trade-off. To evaluate the precision of trajectories generated by our model, we additionally report results on two recently proposed sample quality metrics.

- **Off-road rate:** The off-road rate metric proposed by Niedoba *et al.* [117] computes the fraction of all predicted points that fall outside the road. For the NuScenes dataset, we use the drivable area mask to compute off-road rate. For SDD, we hand label the bird's eye view images in the test set, assigning each pixel as a path or an obstacle. Paths include roads, sidewalks, walkways etc., while obstacles include buildings, terrain, parked cars and road dividers.

- **Off-yaw metric:** For vehicles moving through city streets, the off-road rate metric fails to penalize predictions that fall onto oncoming traffic or illegal lanes. We thus additionally report the off-yaw metric proposed by Greer *et al.* [59], for the NuScenes dataset. The off-yaw metric computes the deviation between the direction of motion

Table 4.1. Results on SDD test set for split used in [154]

Model	MinADE_5	MinADE_{20}	MinFDE_5	MinFDE_{20}	Off-road rate
S-GAN [60]	-	27.25	-	41.44	-
Desire [90]	19.25	-	34.05	-	-
MATF [195]	-	22.59	-	33.53	-
SoPhie [154]	-	16.27	-	29.38	-
CF-VAE [14]	-	12.60	-	22.30	-
HBA-flow [16]	-	**10.80**	-	19.80	-
P2T (ours)	**15.90**	10.97	**30.48**	**18.40**	0.06

Table 4.2. Results on SDD test set for split used in [106]

Model	MinADE_5	MinADE_{20}	MinFDE_5	MinFDE_{20}	Off-road rate
PECNet [106]	12.79	9.96	29.58	15.88	-
Y-Net [105]	**11.49**	**7.85**	**20.23**	**11.85**	-
P2T (ours)	12.81	8.76	23.95	14.08	0.06

of the nearest lane and the yaw of predicted points. Similar to Greer *et al.*, we only penalize deviations above 45° to avoid penalizing lane changes.

Figure 4.5 illustrates how the off-road rate and off-yaw rate can penalize a set of diverse but imprecise forecasts that MinADE_K, MinFDE_K and $\text{MR}_{K,2}$ metrics fail to penalize.

4.4.3 Comparison with the state of the art

We compare our model with prior and concurrently developed models that represent the state of the art for the Stanford drone and NuScenes.

SDD: Table 4.1 reports results on SDD based on the dataset split used in [154]. While most prior works have reported MinADE_K and MinFDE_K for K=20, Desire [90] has results reported for K=5. We report metrics for both values of K here for our models. Note that the error values are in pixels in the bird's eye view image co-ordinates. We also report off-road rate values on SDD for our models based on per pixel path/obstacle labels

Table 4.3. Results on NuScenes test set for the prediction challenge

Model	MinADE_5	MinADE_{10}	$\text{MR}_{5,2}$	$\text{MR}_{10,2}$	Off-road rate	Off-yaw metric
Physics oracle [129]	3.70	3.70	0.88	0.88	0.12	-
CoverNet [129]	2.62	1.92	0.76	0.64	0.13	-
MTP [32]	2.44	1.57	0.70	0.55	0.11	0.11
M-SCOUT [24]	1.92	1.92	0.78	0.78	0.10	-
Trajectron++ [155]	1.88	1.51	0.70	0.57	0.25	-
SG-Net [179]	1.86	1.40	0.67	0.52	0.04	-
MHA-JAM [109]	1.81	1.24	**0.59**	**0.46**	0.07	-
cxx [101]	1.63	1.29	0.69	0.60	0.08	-
Multipath [26]	1.63	1.50	0.75	0.74	0.38	0.37
P2T (Ours)	**1.45**	**1.16**	0.64	**0.46**	**0.03**	**0.04**

for the SDD test set. Our model achieves the lowest MinFDE_K values, while only being closely outperformed by the HBA-Flow model on MinADE_K.

Table 4.2 reports results on the dataset split used by Mangalam *et al.* [105, 106]. This uses a subset of trajectories in SDD, primarily consisting of pedestrians. Our model outperforms PECNet [106]. However, the recently proposed Y-Net [105] achieves lower MinADE_K and MinFDE_K values. Similar to our models, Y-Net also conditions trajectories on goals and intermediate waypoints of agents in the scene, suggesting the importance of modeling the static scene for trajectory forecasts.

NuScenes: Table 4.3 reports results on the NuScenes prediction benchmark [2]. We compare our models with the physics oracle and CoverNet [129] baselines released with the benchmark, and the winning entries of the NuScenes prediction challenge, cxx [101], MHA-JAM [109] and Trajectron++ [155]. Additionally, we also consider the simple yet effective MTP [32] and Multipath [26] models as implemented and reported on NuScenes by Greer *et al.* [59]. Since NuScenes requires a single set of trajectories to evaluate metrics for K=5 and K=10, we merge the clustered trajectories for K=5 and K=10. To remove duplicates, we discard trajectories from the set of 10 closest to each trajectory in the

[2]URL: https://eval.ai/web/challenges/challenge-page/591/leaderboard/1659, results as of May 26, 2021

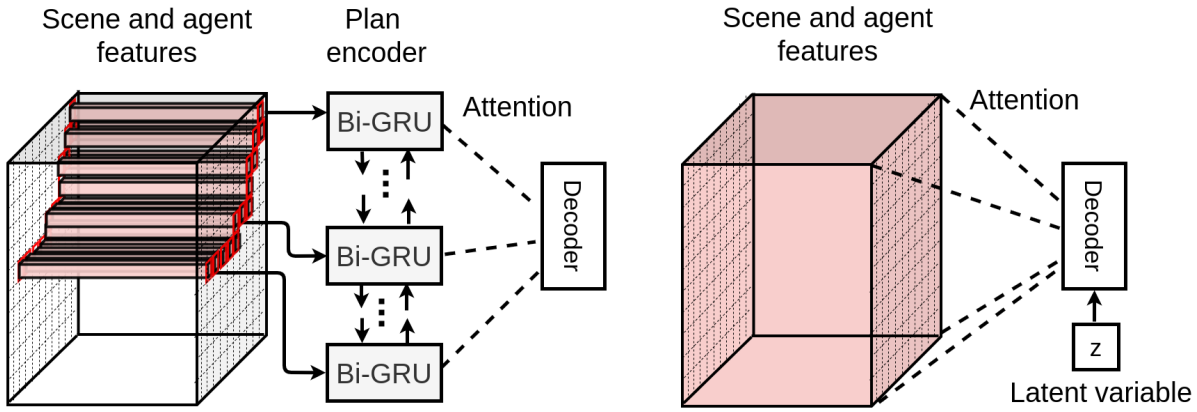


Figure 4.6. Ablation of grid based plans: Models with (left) and without (right) the plan encoder and grid based policy. Without the grid based plan, the trajectory decoder attends to all features within the grid

set of 5 in terms of displacement error. The set of 5 trajectories is nominally assigned a higher score than the set of 10 trajectories. The benchmark does not include results for the off-yaw metric. However, we report the metric for our models and those from [59].

Our model achieves state of the art results on almost metrics on the NuScenes benchmark. In particular, it achieves significantly lower off-road rate and off-yaw metrics compared to previous methods, while still maintaining low MinADE_K and miss rate values. The low MinADE_K and miss rates suggest that our model generates a diverse set of trajectories. The low off-road and off-yaw metrics suggest that conditioning trajectories on plans sampled from the MaxEnt policy lead to more scene compliant trajectories. We investigate this further in section 4.4.4.

4.4.4 Ablations

We additionally report results for the following ablations of our model.

- **Grid-based plans:** To analyze the effect of conditioning trajectories on the grid based plans, we consider an ablation of our model without the MaxEnt policy and plan encoder (Figure 4.6). In this case, the trajectory decoder attends to all features in the grid, rather than just those along the sampled state sequence. To

Table 4.4. Ablations on SDD

Model	CNN_{feat}	Reward layers	Grid-based plans	Trajectory generator	MinADE_5	MinADE_{20}	MinFDE_5	MinFDE_{20}	Offroad rate
LVM	✓			✓	18.28	12.17	36.71	20.98	0.11
P2T_{CS}	✓	✓	✓		21.70	16.10	38.25	25.22	0.09
P2T_{BC}	✓		✓	✓	15.93	11.56	**30.29**	19.51	**0.06**
P2T_{IRL}	✓	✓	✓	✓	**15.90**	**10.97**	30.48	**18.40**	**0.06**

Table 4.5. Ablations on NuScenes

Model	CNN_{feat}	Reward layers	Grid-based plans	Trajectory generator	MinADE_5	MinADE_{10}	$\text{MR}_{5,2}$	$\text{MR}_{10,2}$	Offroad rate	Off-yaw metric
LVM	✓			✓	1.77	1.27	0.80	0.63	0.10	0.12
P2T_{CS}	✓	✓	✓		4.18	4.05	0.93	0.92	**0.02**	0.07
P2T_{BC}	✓		✓	✓	1.47	**1.15**	0.67	0.49	0.04	0.07
P2T_{IRL}	✓	✓	✓	✓	**1.45**	1.16	**0.64**	**0.46**	0.03	**0.04**

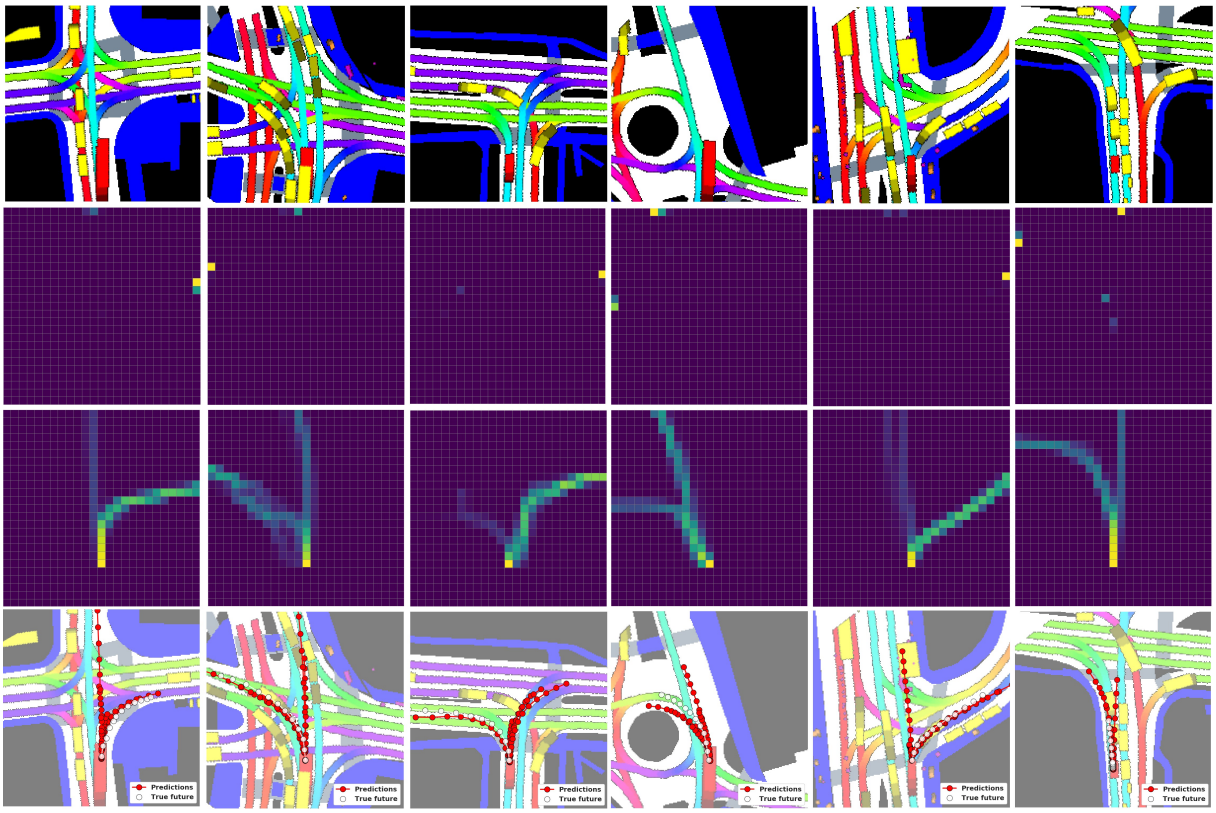


Figure 4.7. Qualitative examples from NuScenes. From top to bottom: Inputs, goal SVFs, path SVFs and predictions

keep memory usage tractable, we maxpool the features using a 2×2 kernel before attention layers. In order to sample a diverse set of trajectories, we additionally condition the trajectory decoder with a latent variable z sampled from a univariate Gaussian distribution. We refer to this ablation as the latent variable model (LVM).

- **Trajectory generator:** Next, we consider a model without the trajectory generator. To output continuous valued trajectories along sampled plans, we fit a smoothing spline along the sampled grid locations and propagate a constant speed trajectory along the spline using the target agent's velocity at the prediction instant. We refer to this model as P2T$_{CS}$.

- **Reward Layers:** Finally, we consider an ablation without the reward layers to

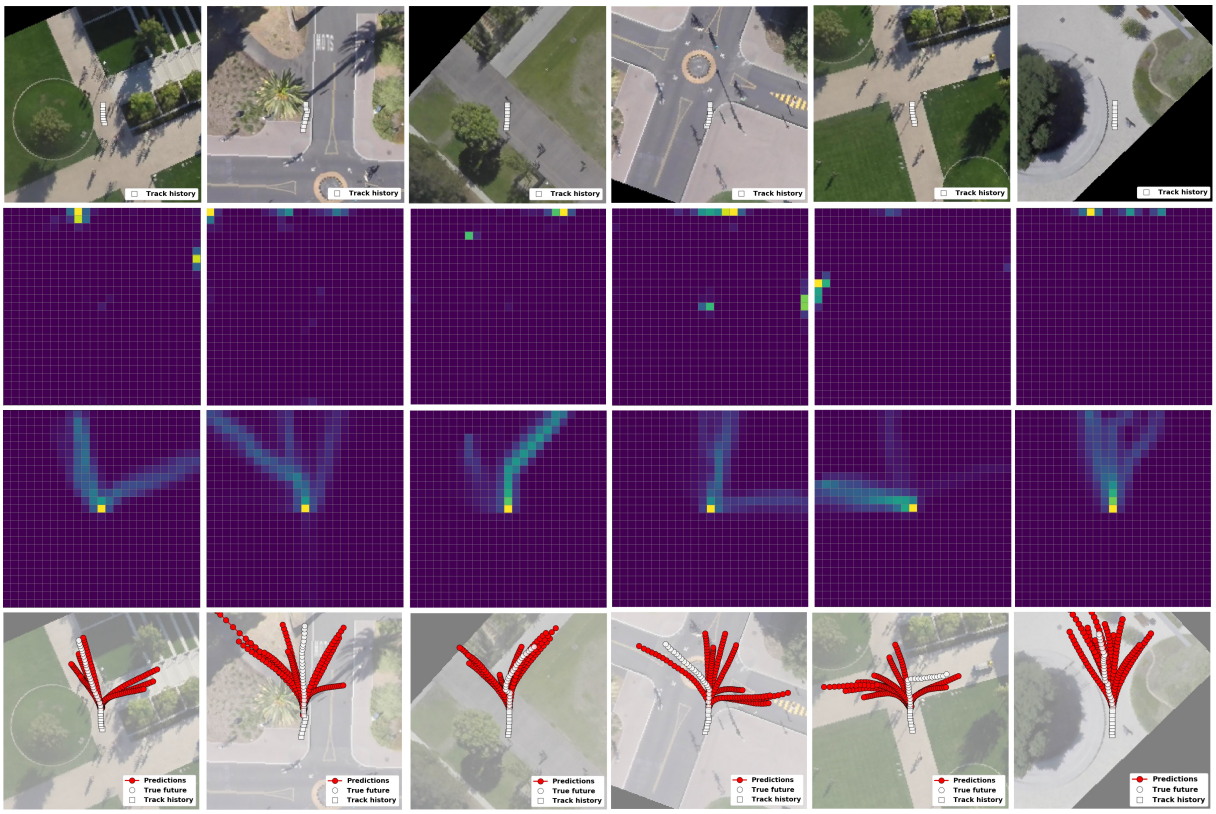


Figure 4.8. Qualitative examples from SDD. From top to bottom: Inputs, goal SVFs, path SVFs and predictions

analyze the usefulness of using IRL. Instead of learning the reward, we learn a behavior cloning policy that directly maps the scene and motion features to action probabilities at each grid cell. We refer to this model as P2T$_{BC}$.

Tables 4.4 and 4.5 report results for ablation studies on SDD and NuScenes respectively. We note that for both datasets, our complete proposed model (P2T$_{IRL}$) outperforms the LVM across all metrics. In particular, the high off-road and off-yaw metrics for the LVM compared to the other three models suggest that the LVM generates more trajectories that veer off the road or violate lane direction. This shows that the inductive bias due to grid based plans leads to trajectories that are more scene compliant. Conversely, P2T$_{CS}$ achieves comparable offroad and off-yaw metrics as P2T$_{IRL}$. However it does poorly in terms of the MinADE, MinFDE and miss rate metrics. Thus, although

Table 4.6. Inference time

Component	Time
Reward model	2 ms
Solve for MaxEnt policy (Algorithm 3)	28 ms
Sample MaxEnt Policy	28 ms
Trajectory Generator	12 ms
Clustering	9 ms
Total	79 ms

its trajectories are scene compliant, they deviate significantly from the ground truth, suggesting the limitation of the constant speed model compared to the attention based GRU decoder for modeling agent dynamics. Finally, P2T$_{IRL}$ slightly outperforms the behavior cloning model P2T$_{BC}$ on most metrics, with the difference being more prominent for SDD than for NuScenes.

4.4.5 Runtime

In table 4.6 we provide inference times for each component of the model. Inference is performed using an NVIDIA GeForce GTX 1080 Ti GPU. We also implement algorithms 1 and 2 using vectorized operations on the GPU. For each prediction instance, we sample 1000 state sequences from MaxEnt policy, generating 1000 trajectories, which are finally clustered to output K trajectories. The runtimes reported here are for K=10. We note inference can be performed in 79 ms (or 12Hz) for the complete proposed model which will allow for real-time deployment, given access to rasterized birds eye view scene representations and past tracks of agents.

4.4.6 Qualitative examples

Figures 4.7 and 4.8 show qualitative examples from the NuScenes and Stanford drone datasets. We show the input scene and past tracks of agents, heat maps for goal and path state visitation frequencies for the MaxEnt policy and the final set of 10 clustered

trajectories from the trajectory generator. We note that the MaxEnt policy explores plausible path and goal states in the 2-D grid for a variety of scene configurations. For the Nuscenes dataset, this corresponds to reachable lanes for the target agent. Note that the policy accurately infers which lanes correspond to the direction of motion rather than oncoming traffic. For SDD, the policy shows a preference for paths and roads while avoiding terrain or obstacles. We also note that the path and goal SVFs are multimodal. Finally, the predicted trajectories closely map to the states explored by the policy, leading to a diverse set of scene compliant predictions over a variety of scene configurations.

4.5 Conclusions

We introduced an approach to forecast trajectories of pedestrians and vehicles in unknown scenes conditioned on plans sampled from a grid based MaxEnt IRL policy. We reformulated MaxEnt IRL to learn a policy that can jointly infer goals and paths of agents on a coarse 2-D grid defined over the scene. We showed that our policy infers plausible goals of agents and paths to these goals that conform to the underlying scene. Additionally, we showed that our policy induces a multi-modal distribution over path and goal states. Next, we introduced an attention based trajectory generator that outputs continuous valued trajectories conditioned on state sequences sampled from our MaxEnt policy. Trajectories sampled from our trajectory generator are diverse and conform to the scene, outperforming prior approaches on the TrajNet benchmark split of the Stanford drone dataset and the NuScenes prediction benchmark. With an inference time of 79 ms, the proposed model can readily be deployed in conjunction with on board detectors [141], trackers [135, 140] and HD Maps for autonomous driving.

Acknowledgements

Chapter 4, in part, is a reprint of the material as it appears in: "Trajectory Forecasts in Unknown Enviroments Conditioned on Grid-based Plans," Nachiket Deo, and Mohan M. Trivedi, arXiv:2001.00735 (2020). The dissertation author was the primary investigator and author of this paper.

Chapter 5

Trajectory Prediction Conditioned on Lane-Graph Traversals

5.1 Introduction

High definition (HD) maps of driving scenes provide a succinct representation of the road topology and traffic rules, and have been a critical component of recent trajectory prediction models as well as public autonomous driving datasets [21, 27]. The first few works that utilized HD maps for prediction [25, 26, 32] encoded HD maps using a rasterized bird's eye view (BEV) image and convolutional layers. This was the approach we used in the previous chapter for encoding the scene in P2T. While this approach exploits the expressive power of modern CNN architectures, it has a few drawbacks:

- **Computational inefficiency:** HD map elements are originally stored using polylines for lanes and polygons for cross-walks, sidewalks and stop-lines. Rasterization of map elements into a BEV image leads to a dense representation which then needs to be encoded by the CNN backbone. Additionally, square receptive fields of CNNs are an inefficent way to aggregate relevant scene context for prediction, which tends to be along lanes.

- **Large state space:** Rasterization and encoding with CNN layers leads to a grid representation of the state space explored by the policy in P2T. While this generalizes

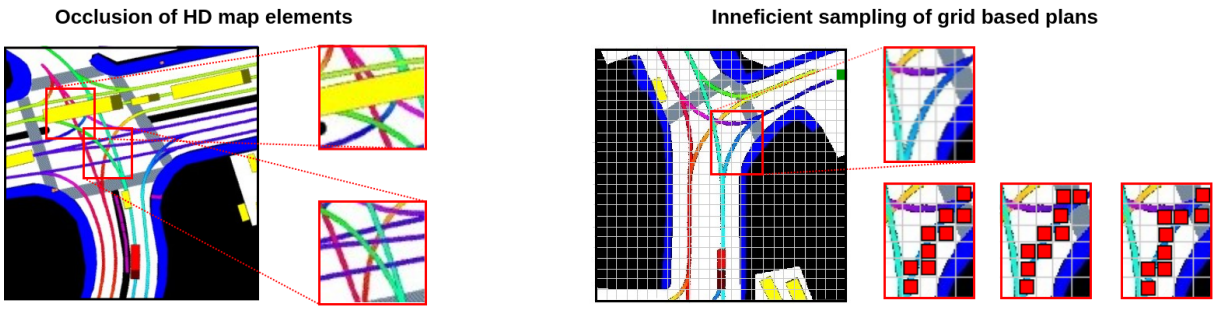


Figure 5.1. Drawbacks of rasterized HD maps: Rasterization of HD map elements in the bird's eye view can lead to occlusion of scene elements (*left*). Encoding rasterized HD maps with CNN layers yields a grid representation of the state space for sampling plans as described in the previous chapter. Sampling from a grid based policy can lead to redundant samples (*right*).

well to novel scene layouts, it leads to a large state space. This in turn leads to slower inference while solving for and sampling from the grid-based policy.

- **Occlusion:** Rasterization of the HD map can lead to occlusion of scene elements, especially when multiple lanes cross each other at intersections. Figure 5.1 shows examples of occluded HD map elements in the rasterized BEV representation.

- **Sample inefficiency:** Sampling plans from the grid-based policy can often lead to redundant samples. Fig 5.1 shows an example of a lane centerline with the grid state space overlayed on it. Distinct paths explored by the policy all correspond to the same behavior of following the lane.

As an alternative to rasterized HD maps, the recently proposed VectorNet [53] and LaneGCN [95] models directly encode structured HD maps, representing lane polylines as nodes of a graph. VectorNet aggregates context using attention [175], while LaneGCN proposes a dilated variant of graph convolution [83] to aggregate context along lanes. These approaches achieve state-of-the-art performance using lightweight encoders with fewer parameters than rasterization-based approaches. However, the above methods encode the entire scene into a single context vector as shown in Fig.5.2. The context vector is then used by a multimodal prediction header [26, 32] to output multiple plausible future

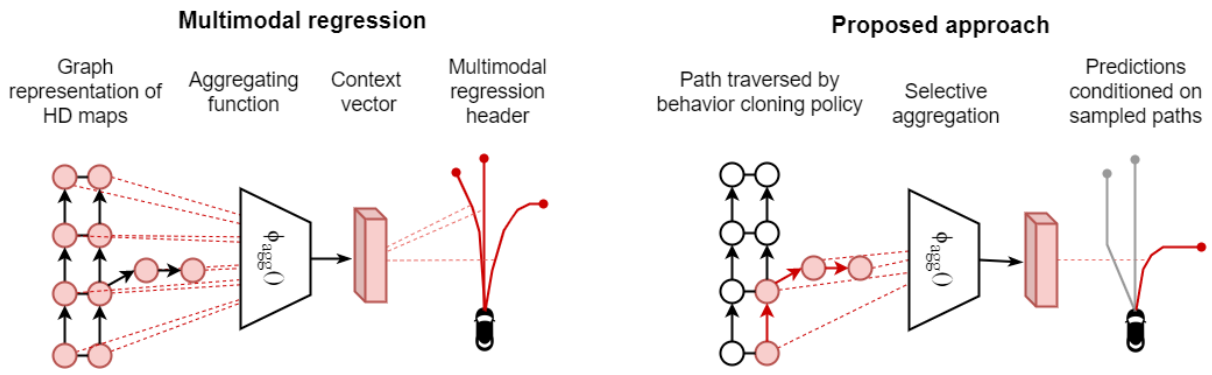


Figure 5.2. Overview of our approach. We encode HD maps and agent tracks using a graph representation of the scene. However, instead of aggregating the entire scene context into a single vector and learning a one-to-many mapping to multiple trajectories, we condition our predictions on selectively aggregated context based on paths traversed in the graph by a discrete policy.

trajectories. The prediction header thus needs to learn a complex mapping, from the entire scene context to multiple future trajectories, often leading to predictions that go off the road or violate traffic rules. In particular, the prediction header needs to account for both *lateral* or *route* variability (e.g. will the driver change lane, will they turn right etc.) as well as *longitudinal* variability (e.g. will the driver accelerate, brake, maintain speed).

5.1.1 Contributions

Our core insight is that the graph structure of the scene can additionally be leveraged to explicitly model the lateral or route variability in trajectories. We propose a novel approach for trajectory prediction termed Prediction via Graph-based Policy (PGP). Our approach relies on two key ideas.

1. **Predictions conditioned on graph traversals:** Instead of using a grid based policy to sample routes in the scene, we represent the HD map as a graph, where nodes represent lane centerlines and edges represent connectivity of the lanes. We learn a discrete policy that explores the graph, such that sampled traversals represent plausible routes that the agent of interest could follow. We then selectively aggregate part of the scene along sampled traversals for each prediction, as shown in Fig. 5.2.

By more directly selecting the subset of the graph that is used for each prediction, we lessen the representational demands on the output decoder. Additionally, the probabilistic policy leads to a diverse set of sampled paths and captures the lateral variability of the multimodal trajectory distribution. Finally, the graph representation leads to a more compact state space, a lightweight map encoder and efficient sampling compared to a grid representation of the scene.

2. **Latent variable for longitudinal variability:** To account for longitudinal variability of trajectories, we additionally condition our predictions with a sampled latent variable. This allows our model to predict distinct motion profiles even for identical path traversals. We show through our experiments that this translates to greater longitudinal variability of predictions.

We summarize our main contributions on multimodal motion prediction using HD maps:

- A novel method combining discrete policy roll-outs with a lane-graph subset decoder.
- State-of-the-art performance on the nuScenes motion prediction challenge.
- Extensive ablations demonstrating the ability to capture lateral (route) and longitudinal (motion) variations.

5.2 Related Research

5.2.1 Graph representation of HD maps

Most self-driving cars have access to HD vector maps, which include detailed geometric information about objects such as lanes, crosswalks, stop signs, and more. VectorNet [53] encodes the scene using a hierarchical representation of map objects and agent trajectories. Each component is represented as a sequence of vectors, which are then processed by a local graph network. The resulting features are aggregated via global

attention layers. LaneGCN [95] extracts a lane graph from the HD map, and uses a graph convolutional network to compute lane features. These features are combined with both agent and lane features in a fusion network. Both methods utilize the entire graph for making predictions, relying on the header to identify the most relevant features.

5.2.2 Multimodal trajectory prediction

Researchers have proposed a variety of ways to model the multiple possible future trajectories that vehicles may take. One approach is to model the output as a probability distribution over trajectories, using either regression [32], ordinal regression [26], or classification [129]. Another approach models the output as a spatial-temporal occupancy grid [190]. Sampling methods use stochastic policy roll outs [147, 148] or latent variable models that map a latent variable sampled from a simple distribution to a predicted trajectory. Latent variable models are trained as GANs [60, 195], CVAEs [90, 155], or directly using the winner-takes-all regression loss [103]. These models must learn a one-to-many mapping from the entire input context (except the random variable) to multiple trajectories, and can lead to predictions that are not scene compliant.

5.2.3 Goal-conditioned trajectory prediction

Rather than learning a one to many mapping from the entire context to multiple future trajectories, methods such as TnT [194], LaneRCNN [189], and PECNet [105] condition each prediction on goals of the driver. Conditioning predictions on future goals makes intuitive sense and helps leverage the HD map by restricting goals to be near the lanes. However, one limitation is that over moderate time horizons, there can be multiple paths that reach a given goal location. Additionally, certain plausible goal locations might be unreachable due to constraints in the scene that are not local to the goal location, e.g., a barrier that blocks a turn lane. In contrast, our method conditions on paths traversed in a lane graph, which ensures that the inferred goal is reachable. Furthermore, the traversed

path provides a stronger inductive bias than just the goal location.

A similar stream of work conditions on candidate lane centerlines as goals (e.g., WIMP [79], GoalNet [191], CXX [101]). While the lane centerline provides more local context than just the goal, accounting for lane changes can be difficult. Additionally, routes need to be deterministically chosen, with multiple trajectories predicted along the selected route. Our approach allows for probabilistic sampling of both routes and motion profiles. In scenes with just a single plausible route, our model can use its prediction budget of K trajectories purely for different plausible motion profiles.

Closest to this work is P2T [40] from the previous chapter. P2t predicts trajectories conditioned on paths explored by an IRL policy over a grid defined over the scene. However, we use a rasterized BEV image for the scene in P2T, which leads to inefficient encoders and loss of connectivity information due to occlusions. Additionally, P2T cannot generate different motion profiles along a sampled path.

5.3 Formulation

We predict trajectories of vehicles of interest, conditioned on their past trajectory, the past trajectories of nearby vehicles and pedestrians, and the HD map of the scene. We represent the scene and predict trajectories in the bird's eye view and use an agent-centric frame of reference aligned along the agent's instantaneous direction of motion.

5.3.1 Trajectory representation

We assume access to past trajectories of agents in the scene obtained from on-board detectors and multi-object trackers. We represent the past trajectory of agent i as a sequence of motion state vectors $s^i_{-t_h:0} = [s^i_{-t_h}, ..., s^i_{-1}, s^i_0]$ over the past t_h time steps. Each $s^i_t = [x^i_t, y^i_t, v^i_t, a^i_t, \omega^i_t, \mathcal{I}^i]$, where x^i_t, y^i_t are the BEV location co-ordinates, v^i_t, a^i_t and ω^i_t are the speed, acceleration and yaw-rate of the agent at time t, and $\mathcal{I}^i$ is an indicator with value 1 for pedestrians and 0 for a vehicles. We nominally assign the index 0 to the target

vehicle, and timestamp 0 to the time of prediction.

5.3.2 Representing HD maps as lane graphs

Nodes: We represent the HD map as a directed graph $\mathcal{G}(V, E)$. The network of lane centerlines captures both, the direction of traffic flow, and the legal routes that each driver can follow. We seek to use both as inductive biases for our model. We thus use lane centerlines as nodes (V) in our graph. We consider all lane centerlines within a fixed area around the target vehicle. To ensure that each node represents a lane segment of a similar length, we divide longer lane centerlines into smaller snippets of a fixed length, and discretize them to a set of N poses. Each snippet corresponds to a node in our graph, with a node v represented by a sequence of feature vectors $f^v_{1:N} = [f^v_1, ..., f^v_N]$. Here each $f^v_n = [x^v_n, y^v_n, \theta^v_n, \mathcal{I}^v_n]$, where x^v_n, y^v_n and θ^v_n are the location and yaw of the n^{th} pose of v and $\mathcal{I}^v_n$ is a 2-D binary vector indicating whether the pose lies on a stop line or crosswalk. Thus, our node features capture both the geometry as well as traffic control elements along lane centerlines.

Edges: We constrain edges (E) in the lane graph such that any traversed path through the graph corresponds to a legal route that a vehicle can take in the scene. We consider two types of edges. Successor edges (E_{suc}) connect nodes to the next node along a lane. A given node can have multiple successors if a lane branches out at an intersection. Similarly, multiple nodes can have the same successor if two or more lanes merge. To account for lane changes, we additionally define proximal edges (E_{prox}) between neighboring lane nodes if they are within a distance threshold of each other and their directions of motion are within a yaw threshold. The yaw threshold ensures that proximal edges are not erroneously assigned in intersections where multiple lanes cross each other.

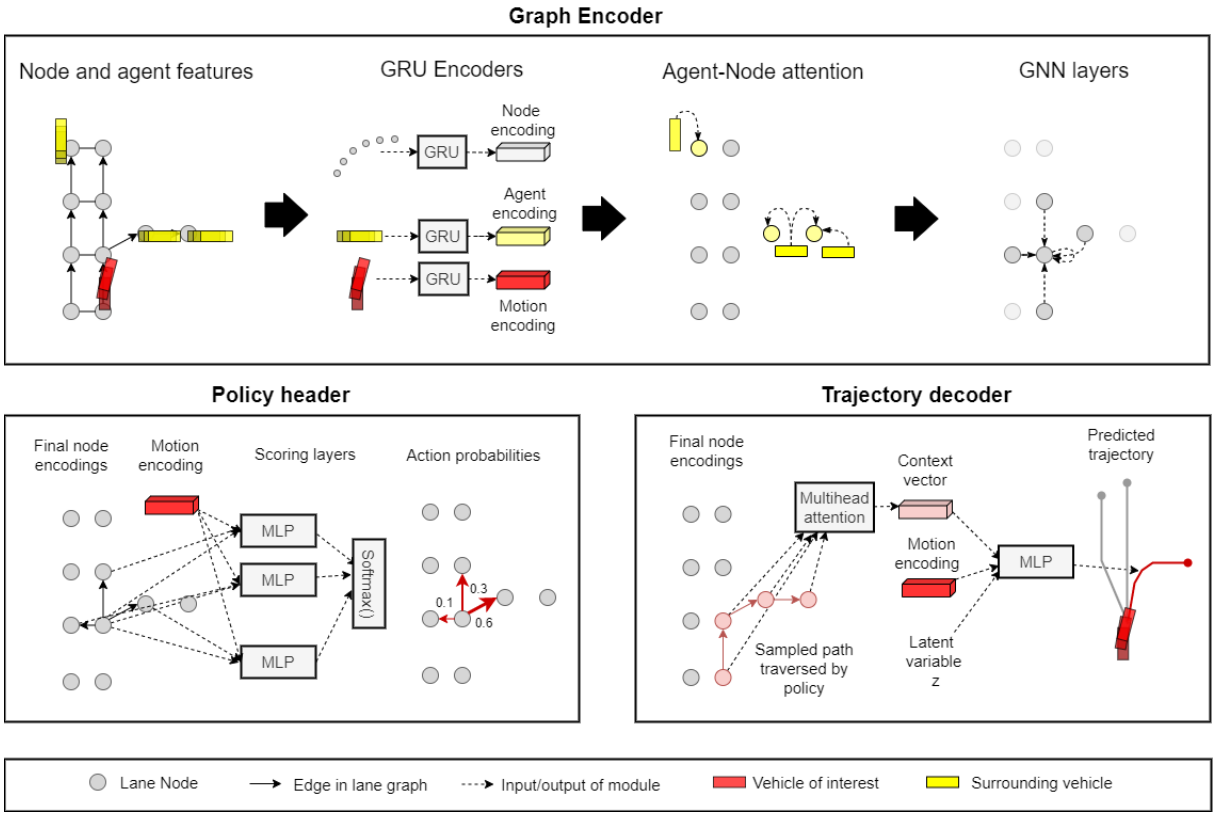


Figure 5.3. PGP: PGP consists of three modules trained end-to-end. The graph encoder (top) encodes agent and map context as node encodings of a directed lane-graph. The policy header (bottom-left) learns a discrete policy for sampled graph traversals. The trajectory decoder (bottom-right) predicts trajectories by selectively attending to node encodings along paths traversed by the policy and a sampled latent variable.

5.3.3 Output representation

To account for multimodality of the distribution of future trajectories, we output a set of K trajectories $[\tau^1_{1:t_f}, \tau^2_{1:t_f}, ..., \tau^K_{1:t_f}]$ for the target vehicle consisting of future x-y locations over a prediction horizon of t_f time steps. Each of the K trajectories represents a mode of the predictive distribution, ideally corresponding to different plausible routes or different motion profiles along the same route.

5.4 Proposed Model

Fig. 5.3 provides an overview of our model. It consists of three interacting modules trained end-to-end. The *graph encoder* (Sec. 5.4.1) forms the backbone of our model. It outputs learned representations for each node of the lane graph, incorporating the HD map as well as surrounding agent context. The *policy header* (Sec. 5.4.2) outputs a discrete probability distribution over outgoing edges at each node, allowing us to sample paths in the graph. Finally, our attention based *trajectory decoder* (Sec. 5.4.3) outputs trajectories conditioned on paths traversed by the policy and a sampled latent variable.

5.4.1 Encoding scene and agent context

Inspired by the simplicity and effectiveness of graph based encoders for trajectory prediction [53, 95], we seek to encode all agent features and map features as node encodings of our lane graph $\mathcal{G}(V, E)$.

GRU encoders. Both, agent trajectories and lane polylines form sequences of features with a well defined order. We first independently encode both sets of features using gated recurrent unit (GRU) encoders. We use three GRU encoders for encoding the target vehicle trajectory $s^0_{-t_h:0}$, surrounding vehicle trajectories $s^i_{-t_h:0}$ and node features $f^v_{1:N}$. These output the motion encoding h_{motion}, agent encodings h^i_{agent} and initial node encodings h^v_{node} respectively.

Agent-node attention. Drivers co-operate with other drivers and pedestrians to navigate through traffic. Thus, surrounding agents serve as a useful cue for trajectory prediction. Of particular interest are agents that might interact with the target vehicle's route. We thus update node encodings with nearby agent encodings using scaled dot product attention [175]. We only consider agents within a distance threshold of each lane node to update the node encoding. This allows our trajectory decoder (Sec 5.4.3) to selectively focus on agents that might interact with specific routes that the target vehicle might take.

We obtain keys and values by linearly projecting encodings h^i_{agent} of nearby agents, and the query by linearly projecting h^v_{node}. Finally, the updated node encoding is obtained by concatenating the output of the attention layer with the original node encoding.

GNN layers. With the node encodings updated with nearby agent features, we exploit the graph structure to aggregate local context from neighboring nodes using graph neural network (GNN) layers. We experiment with graph convolution (GCN) [83] and graph attention (GAT) [176] layers. For the GNN layers, we treat both successor and proximal edges as equivalent and bidirectional. This allows us to aggregate context along all directions around each node. The outputs of the GNN layers serve as the final node encodings learned by the graph encoder.

5.4.2 Discrete policy for graph traversal

Every path in our directed lane graph corresponds to a plausible route for the target vehicle. However, not every route is equally likely. For example, the past motion of the target vehicle approaching an intersection might indicate that the driver is preparing to make a turn rather than go straight. A slow moving lane make it likelier for the target vehicle to change lane rather than maintain lane.

We seek to learn a policy π_{route} for graph traversal such that sampled roll-outs of the policy correspond to likely routes that the target vehicle would take in the future. We represent our policy as a discrete probability distribution over outgoing edges at each node. We additionally include edges from every node to an *end* state to allow π_{route} to terminate at a goal location. The edge probabilities are output by the policy header shown in Fig. 5.3. The policy header uses an MLP with shared weights to output a scalar score for each edge (u, v) given by

$$\text{score}(u, v) = \text{MLP}\left(\text{concat}(h_{motion}, h^u_{node}, h^v_{node}, \mathbb{1}_{(u,v)\in E_{suc}})\right). \tag{5.1}$$

The scoring function thus takes into account the motion of the target vehicle as well as local scene and agent context at each edge. We then normalize the scores using a softmax layer for all outgoing edges at each node to output the policy for graph traversal,

$$\pi_{route}(v|u) = \text{softmax}(\{\text{score}(u, v)|(u, v) \in E\}). \tag{5.2}$$

We train the policy header using behavior cloning. For each prediction instance, we use the ground truth future trajectory to determine which nodes were visited by the vehicle. We can naively assign each pose in the future trajectory to the closest node in the graph. However, this can lead to erroneous assignment of nodes in intersections, where multiple lanes intersect. We thus only consider lane nodes whose direction of motion is within a yaw threshold of the target agent's pose. An edge (u, v) is treated as visited if both nodes u and v are visited. We use negative log likelihood of the edge probabilities for all edges visited by the ground truth trajectory (E_{gt}), as the loss function for training the graph traversal policy, given by

$$\mathcal{L}_{BC} = \sum_{(u,v)\in E_{gt}} -\log(\pi_{route}(v|u)). \tag{5.3}$$

5.4.3 Decoding trajectories conditioned on traversals

Sampling roll-outs of π_{route} yields plausible future routes for the target vehicle. We posit that the most relevant context for predicting trajectories is along these routes and propose a trajectory decoder that selectively aggregates context along the sampled routes.

Given a sequence of nodes $[v_1, v_2, ..., v_M]$ corresponding to a sampled policy roll-out, our trajectory decoder uses multi-head attention [175] to aggregate map and agent context over the node sequence. We linearly project the target vehicle's motion encoding to obtain the query, while we linearly project the node features $[h_{node}^{v_1}, h_{node}^{v_2}, ..., h_{node}^{v_M}]$ to obtain keys and values for computing attention. The multi-head attention layer outputs a context

vector $\mathcal{C}$ encoding the route. Each distinct policy roll-out yields a distinct context vector, allowing us to predict trajectories along a diverse set of routes.

Diversity in routes alone does not account for the multimodality of future trajectories. Drivers can brake, accelerate and follow different motion profiles along a planned route. To allow the model to output distinct motion profiles, we additionally condition our predictions with a sampled latent vector z. Unlike routes, vehicle velocities and accelerations vary on a continuum. We thus sample z from a continuous distribution. We use the multivariate standard normal distribution for simplicity.

Finally, to sample a trajectory $\tau^k_{1:t_f}$ from our model, we sample a roll-out of π_{route} and obtain $\mathcal{C}_k$, we sample z_k from the latent distribution and concatenate both with h_{motion} and pass them through an MLP to output $\tau^k_{1:t_f}$ the future locations over t_f timesteps,

$$\tau^k_{1:t_f} = \text{MLP}(\text{concat}(h_{motion}, \mathcal{C}_k, z_k)). \tag{5.4}$$

The sampling process can often be redundant, yielding similar or repeated trajectories. However our light-weight encoder and decoder heads allows us to sample a large number of trajectories in parallel. To obtain a final set of K modes of the trajectory distribution, we use K-means clustering and output the cluster centers as our final set of K predictions $[\tau^1_{1:t_f}, \tau^2_{1:t_f}, ..., \tau^K_{1:t_f}]$. We train our decoder using the winner takes all average displacement error with respect to the ground truth trajectory (τ^{gt}) in order to not penalize the diverse plausible trajectories output by our model,

$$\mathcal{L}_{reg} = \min_k \frac{1}{t_f} \sum_{t=1}^{t_f} \|\tau^k_t - \tau^{gt}_t\|_2. \tag{5.5}$$

We train our model end-to-end using a multi-task loss combining losses from Eq. 5.3 and Eq. 5.5,

$$\mathcal{L} = \mathcal{L}_{BC} + \mathcal{L}_{reg}. \tag{5.6}$$

5.5 Experimental Evaluation

5.5.1 Experimental settings

We evaluate our method on nuScenes [21], a self-driving car dataset collected in Boston and Singapore. nuScenes contains 1000 scenes, each 20 seconds, with ground truth annotations and HD maps. Vehicles have manually-annotated 3D bounding boxes, which are published at 2 Hz. The prediction task is to use the past 2 seconds of object history and the HD map to predict the next 6 seconds. We use the standard split from the nuScenes software kit [119].

5.5.2 Metrics

To evaluate our model, we use the standard metrics on the nuScenes leaderboard [119]. The minimum average displacement error (ADE) over the top K predictions (MinADE_K). The miss rate ($\text{MissRate}_{K,2}$) only penalizes predictions that are further than 2 m from the ground truth. The offroad rate measures the fraction of predictions that are off the road. Since all examples in nuScenes are on the road, this should be zero. Additionally, we report metrics measuring sample diversity of a set of K predictions. To measure lateral diversity, we report the average number of distinct final lanes reached, and the variance of final heading angle of the target vehicle (σ^2_{yaw}) for the set of K trajectories. To measure longitudinal diversity, we report the variance of average speeds (σ^2_{speed}) and accelerations (σ^2_{acc}) for the set of K trajectories.

5.5.3 Comparison to the state of the art

We report our results on the standard benchmark split of nuScenes in table 5.1, comparing with the top performing entries on the nuScenes leaderboard. We achieve state of the art results on almost all metrics, significantly outperforming the previous best entry P2T [40] on the MinADE_K and MissRate metrics, while achieving comparable off-road

Table 5.1. Comparison to the state of the art on nuScenes

Model	MinADE_5	MinADE_{10}	$\text{MissRate}_{5,2}$	$\text{MissRate}_{10,2}$	Offroad rate
CoverNet [129]	1.96	1.48	0.67	-	-
Trajectron++ [155]	1.88	1.51	0.70	0.57	0.25
SG-Net [179]	1.86	1.40	0.67	0.52	0.04
MHA-JAM [109]	1.81	1.24	**0.59**	0.46	0.07
CXX [101]	1.63	1.29	0.69	0.60	0.08
P2T [40]	1.45	1.16	0.64	0.46	**0.03**
PGP (Ours)	**1.30**	**1.00**	0.61	**0.37**	**0.03**

Table 5.2. Encoder ablations

Graph structure		Agent-node attention	GNN layers	MinADE_K		$\text{MissRate}_{K,2}$		Offroad rate
E_{suc}	E_{prox}			K=5	K=10	K=5	K=10	
✓				1.35	1.03	0.64	0.41	0.04
✓	✓			1.33	1.01	0.63	0.38	0.03
✓	✓	✓		**1.30**	**1.00**	**0.61**	**0.37**	**0.03**
✓	✓	✓	GCN × 1	1.31	1.01	0.62	0.39	0.04
✓	✓	✓	GCN × 2	1.31	1.01	0.61	0.39	0.04
✓	✓	✓	GAT × 1	**1.30**	**1.00**	0.62	0.38	0.03
✓	✓	✓	GAT × 2	1.31	1.01	**0.61**	**0.37**	**0.03**

rate. This suggests that our model achieves better coverage of the modes of the trajectory distribution, while still predicting trajectories that are scene-compliant.

Table 5.3. Decoder ablations

Decoder	MinADE_5	MinADE_{10}	$\text{MissRate}_{5,2}$	$\text{MissRate}_{10,2}$	Offroad rate
MTP [32]	1.59	1.12	**0.57**	0.48	0.08
Latent var (LV) only	1.38	1.08	0.65	0.43	0.05
Traversal only	1.37	1.10	0.65	0.44	0.04
Goals + LV	1.33	1.02	0.60	0.42	0.06
Traversals + LV	**1.31**	**1.01**	0.61	**0.37**	**0.03**

Table 5.4. Lateral diversity metrics (K=10)

Decoder	# distinct final lanes	σ^2_{yaw}
LV only	1.22	0.11
Traversals + LV	**1.41**	**0.13**

Table 5.5. Longitudinal diversity metrics (K=10)

Decoder	σ^2_{speed}	σ^2_{acc}
Traversal only	2.33	5.28
Traversals + LV	**4.07**	**6.65**

5.5.4 Encoder ablations

We analyze the effects of our graph structure and components of the graph encoder by performing ablations on the graph encoder reported in table 5.2. In particular we analyze the effect of including proximal edges, modeling surrounding agents with agent-node attention and finally aggregating local context using GCN [83] or GAT [176] layers. We get improvement across all metrics by adding proximal edges, and agent-node attention, suggesting the importance of modeling lane changes and agent context. Somewhat surprisingly, adding GNN layers gives ambiguous results with GCN layers achieving slightly worse results and GAT layers performing on par with the encoder without GNN layers. This could be because the multi-head attention layer aggregates context across the entire traversed path, making the GNNs redundant.

5.5.5 Decoder ablations

We next analyze the effect of our traversal and latent variable based decoder. We compare several decoders, all built on top of our proposed encoder with both types of edges, agent-node attention and 2 GAT layers. First, we consider the multimodal regression header from MTP [32]. Next we consider ablations of our decoder without the graph traversals and without the latent variable conditioning. Finally, we consider a model that

conditions predictions on sampled goals at different node locations, instead of traversals. Table 5.3 reports quantitative results while Fig. 5.4 shows qualitative examples comparing the decoders. We make the following observations.

MTP generally fares worse compared to the other decoders, particularly in terms of offroad rate. We note from Fig. 5.4 that while it generates a diverse set of trajectories, several veer off-road.

The decoders conditioned purely on the latent variable or purely on traversals both fare worse in terms of MinADE and MissRate compared to our decoder conditioned on both. From the sample diversity metrics (Tables 5.4 and 5.5) and qualitative examples (Fig.5.4) we observe that this is for different reasons. The 'LV only' decoder generates diverse motion profiles, but almost always predicts trajectories along a single route, leading to poor lateral diversity of trajectories. On the other hand, the 'Traversal only' decoder predicts trajectories over a variety of routes, but lacks diversity in terms of motion profiles.

Finally, the 'Goals + LV' decoder also fares worse compared to our 'Traversals + LV' decoder, again, especially in terms of off-road rate. Qualitatively, we observe that this is due to two types of errors. First, it tends to predict spurious goals which aren't reachable for the target vehicle (Fig.5.4 ③, ④), and second, while it predicts correct goals, it generates trajectories that don't follow accurate paths to those goals (Fig.5.4 ②,⑥).

5.6 Conclusions

We presented a novel method for multimodal trajectory prediction conditioned on paths traversed in a lane graph of the HD map by a discrete policy, and a sampled latent variable. Through experimental analysis and ablation studies using the publicly available nuScenes dataset, we showed that

- Selectively conditioning predictions on lane-graph traversals leads to trajectories that are (i) diverse in terms of routes, and (ii) precise and scene compliant with the

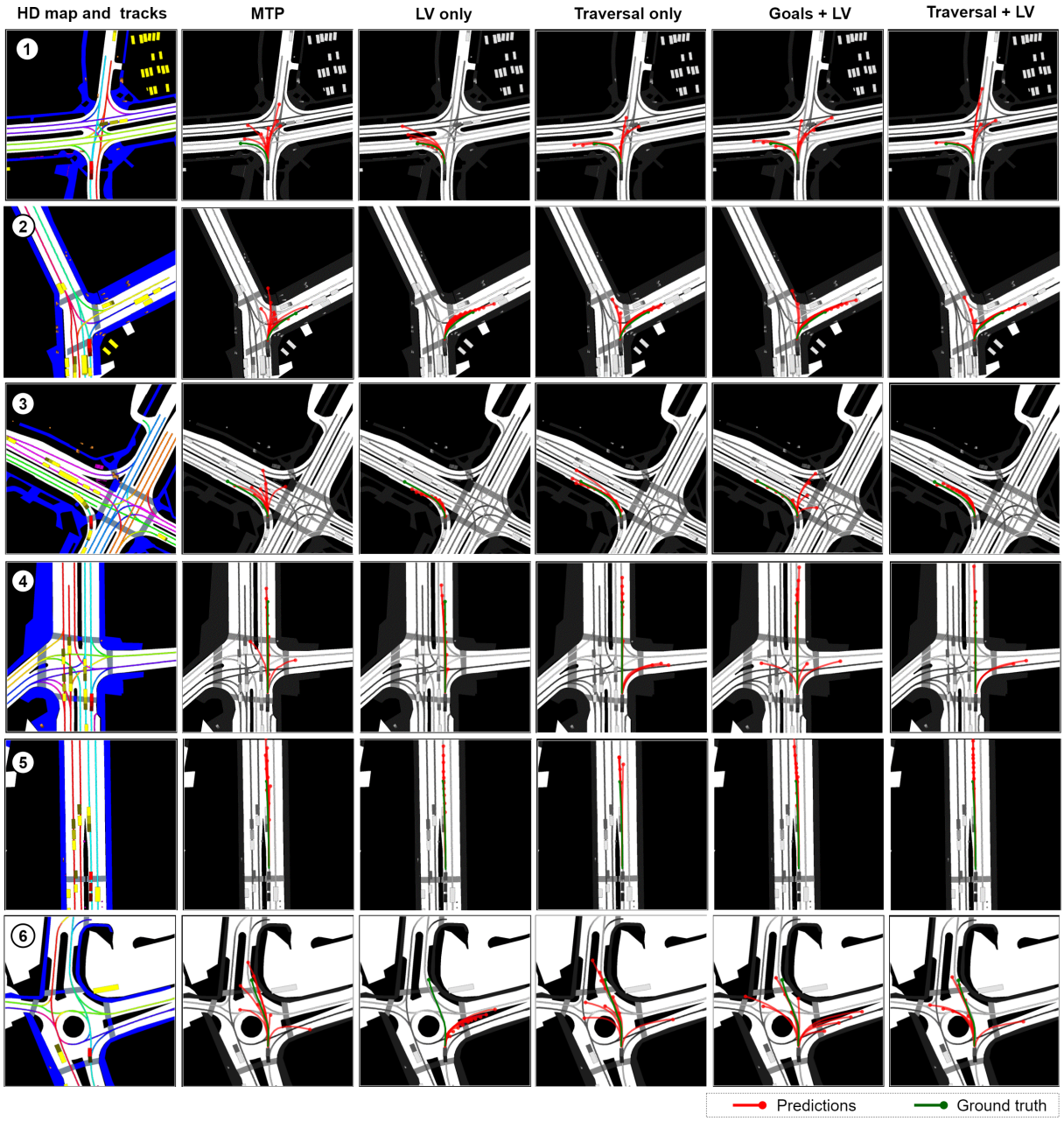


Figure 5.4. Qualitative comparison of decoders: MTP (*column 2*) predicts trajectories that often veer off-road (①-③,⑥). The decoder purely conditioned on latent variables (*column 3*) lacks lateral diversity and predicts trajectories along a single route, even missing the correct route in ⑥. The decoder conditioned purely on traversals (*column 4*) predicts diverse routes, but lacks longitudinal diversity (①,②,⑤). Finally, the decoder conditioned on goals rather than path traversals (*column 5*) predicts spurious goals that may not be reachable (③, ④). Our model (*column 6*) predicts scene-compliant trajectories over a diverse set of routes. In cases with few plausible routes (e.g.⑤), it uses its prediction budget of K trajectories to generate more longitudinal diversity.

lowest offroad-rates.

- Additionally conditioning predictions on sampled latent variables leads to trajectories that are diverse in terms of motion profiles.
- Both put together lead to state of the art results in terms of MinADE and MissRate metrics.

Acknowledgements

Chapter 5, in part, is a reprint of the material as it appears in: "Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals," Nachiket Deo, Eric Wolff and Oscar Beijbom, CoRL 2021. The dissertation author was the primary investigator and author of this paper.

5.A Appendix: Implementation details

Here, we provide the implementation details for the model architecture and experiments from Chapter 5. We implement our model using Pytorch [127]. Here, we provide details of our model architecture, ablations and training.

5.A.1 Map representation

The nuScenes map API provides lane polylines, their successors, and polygons for cross-walks and stop lines. We consider map elements within an area of [-50, 50] m laterally and [-20, 80] m longitudinally around the target vehicle. This ensures that most ground truth trajectories lie within the area of interest. We split longer lane centerlines into snippets of maximum length 20m, and discretize the polylines at a 1m resolution. Each snippet corresponds to a node in the graph. This ensures that each lane node represents a lane segment of similar length. The node resolution (20m) and pose resolution (1m) for the polylines were experimentally chosen. There is a trade-off associated with the

resolution of lane nodes: A finer resolution would provide a more informative set of inputs, but would lead to a graph with a greater number of nodes (and a greater number of poses per node) increasing encoder complexity.

5.A.2 GRU encoders

We embed both agent and node features using linear layers of size 16, followed by a leaky ReLU non-linearity. We use GRUs with depth 1 and hidden state dimension 32 on top of the embeddings for both the agent and node encoders.

5.A.3 Agent-node attention

We use scaled dot-product attention with a single attention head for the agent-node attention layers. We use 32×32 weight matrices for projecting the node and agent encodings for obtaining the queries, and keys and values respectively. The outputs of the attention layer are concatenated with the original node encodings and passed through a linear layer of size 32, followed by a leaky ReLU non-linearity to obtain updated node encodings of the same size as the original node encodings.

5.A.4 GNN layers

We use Pytorch geometric[1] for implementing the GCN and GAT layers of our model. For GCN layers, we use the layer-wise propagation rule from [83]. Our adjacency matrix includes both successor and proximal edges (treated as bidirectional), as well as self loops. The outputs at each node have the same dimension, 32, as the inputs. For GAT layers, we use the layer-wise propagation rule from [176]. We use a single attention head, with the outputs again having the same dimension as the inputs.

[1] https://github.com/rusty1s/pytorch_geometric

5.A.5 Policy header

The policy header is implemented as an MLP with 2 hidden layers of size 32 each and a scalar output. The input to the policy header for each edge is a vector of size 98, consisting of the source node encoding, destination node encoding and motion encoding of the target agent each of size 32, and a one-hot encoding for the edge type of size 2.

5.A.6 Trajectory decoder

We aggregate context along nodes traversed by the policy using a multi-head scaled dot-product attention layer. The attention layer has 32 parallel attention heads, and outputs a context vector $\mathcal{C}$ of size 128. We model the latent variable as a multivariate standard normal distribution. $z \sim \mathcal{N}(0, \mathbf{I})$, where $\mathbf{I}$ is a 5×5 identity matrix. We output a trajectory for each sampled $\mathcal{C}_k$, z_k and h_{motion} using an MLP with a hidden layer of size 128, and output of size 24 (x and y co-ordinates over the prediction horizon of 6 seconds at 2 Hz). We sample 200 trajectories from the model and cluster to obtain K=10 trajectories during training to compute the winner takes all regression loss $\mathcal{L}_{reg}$.

5.A.7 Training

We train the model using Adam, with learning rate 1e-4, and a batch size of 32. For the first few epochs of training, since π_{route} does not produce meaningful traversals, we use the ground truth traversal for sampling trajectories and computing $\mathcal{L}_{reg}$. We pre-train the model using the ground truth traversal for 100 epochs. We then finetune using paths sampled from π_{route} for 100 epochs. We train our model using an AWS "p3-8xlarge" instance with 4 NVIDIA Tesla V100 GPUs. Each pre-training epoch takes roughly 1 minute and each finetuning epoch takes roughly 5 minutes for nuScenes.

5.A.8 Ranking Clustered Trajectories

The nuScenes leaderboard[2] requires a single set of ranked or scored predictions for computing the MinADE_k and MissRate metrics for $k = 1$, 5 and 10. We rank our set of 10 clustered trajectories based on Ward's merging cost[3]. We obtain the two clusters with the minimum merging cost. The trajectory corresponding to the smaller of the two clusters is assigned rank 10. The two clusters are then merged, with the merged cluster assuming the identity of the larger cluster. This process is then repeated to assign ranks 9 through 1. Using Ward's merging cost ensures that the top k trajectories cover a diverse set of modes for all values of k.

5.A.9 Decoder ablation details

MTP: For the MTP header, we first aggregate context over the entire graph using a multi-head scaled dot-product attention layer identical to our trajectory decoder, with 32 parallel attention heads and an output context vector $\mathcal{C}$ of size 128. We then use two fully connected layers of size 240 and 10 respectively to output K=10 trajectories, and K probabilities.

LV only: For the LV only decoder, similar to the MTP header, we first aggregate context over the entire graph using a multi-head attention layer with 32 attention heads and output $\mathcal{C}$ of size 128. The decoder then outputs trajectories conditioned on $\mathcal{C}$, h_{motion} and a sample z_k of the latent variable using the final MLP layer.

Traversal only: The traversal only decoder is identical to the trajectory decoder of our complete model, except for the final MLP layer, which outputs trajectories conditioned only on $\mathcal{C}_k$ and h_{motion} and not on the sampled latent variable z_k.

Goals + LV: The Goals + LV decoder consists of two output headers: A goal prediction header that outputs a scalar score at each node normalized using a softmax layer to give

[2]https://eval.ai/web/challenges/challenge-page/591/leaderboard/1659
[3]https://en.wikipedia.org/wiki/Ward%27s_method

goal probabilities, and a trajectory decoder that outputs goal conditioned trajectories. We model the goal prediction header using an MLP with 2 hidden layers, each of size 32, and a scalar output. The input to the goal prediction header at each node is obtained by concatenating h_{node} and h_{motion}. The trajectory decoder consists of a multi-head attention layer with 32 heads that aggregates context over the entire graph to output a context vector $\mathcal{C}$ of size 128. $\mathcal{C}$ is concatenated with h_{motion}, a sampled latent vector z_k and the node encoding of a sampled goal $h_{node}^{u_k}$ and passed through an MLP with a hidden layer of size 128, and output size 24 corresponding to a goal conditioned trajectory.

Part II

Predicting Driver Behavior during Control Transitions

Chapter 6

Predicting Take-Over Readiness of Drivers using Vision Sensors

6.1 Introduction

The overarching goal of autonomous vehicle research is the development of fully automated systems capable of driving in any traffic scenario. The occupants of such a vehicle would then be mere passengers, without access to controls. However, to safely develop the technology to achieve this goal, there needs to be shared control between the vehicle and a human driver. This can be seen in the 5 levels of vehicle automation defined by the Society of Automotive Engineers (SAE) [145], with levels 2 to 4 corresponding to some form of shared control. Conditionally autonomous vehicles (level 3), can operate autonomously under specific traffic scenarios like lane keeping on freeways while maintaining a desired speed. Such vehicles are now commercially available [1, 2]. However, a human occupant, behind the wheel, is expected to monitor the automated system and be prepared for *take-over requests*. These are cases where control needs to be transferred from the vehicle to the human during failure modes of the system. Continuous estimation of this occupant's take-over readiness is thus critical for safe and timely transfer of control. In the remainder of this chapter, we use the term 'driver' in the context of conditionally autonomous vehicles to refer to the occupant responsible for taking over control.

Prior research [11, 94, 96, 97, 99, 181] has addressed the closely related problem

of estimating *driver distraction* under manual driving conditions. Driver distraction has been defined as the diversion of the driver's attention away from activities critical for safe driving toward a competing activity, which may result in insufficient or no attention to activities critical for safe driving [144]. Conditionally autonomous driving raises the possibility of drivers engaging in secondary activities unobserved during manual driving, as well as more freely engaging in previously known secondary activities. While sophisticated computer vision algorithms have been proposed for driver activity analysis [18,34,46,51,52,62,91,92,112,122–124,136,138,139,164–166,168,169,173,177,178,184,186], relatively few works [20,54,110,146,188] have addressed the problem of mapping driver activity to take-over readiness.

This could be attributed to two main challenges. First, there is a lack of naturalistic driving datasets observing driver activity in conditionally autonomous vehicles. A comprehensive naturalistic driving dataset capturing a large range of driver behaviors would allow for data-driven approaches to map driver activity to take-over readiness. Second, defining the ground truth for take-over readiness is a challenging task. Data-driven approaches hinge on the availability of ground-truth of the quantity being estimated. While electroencephalogram (EEG) sensors allow for the most faithful representation of the driver's brain activity [76,98,171,193], they are too intrusive to be viable in commercial vehicles. Another approach used in recent studies [20] is to define take-over readiness based on take-over time and take-over quality in experimental trials with take-over requests issued to drivers performing secondary activities. However, the nature of the task restricts such experiments to simulator settings.

6.1.1 Contributions

In this chapter, we propose a data-driven approach to estimate the take-over readiness of drivers in conditionally autonomous vehicles, based purely on the outputs of non-intrusive sensors observing the driver. Figure 6.1 summarizes our approach. Our

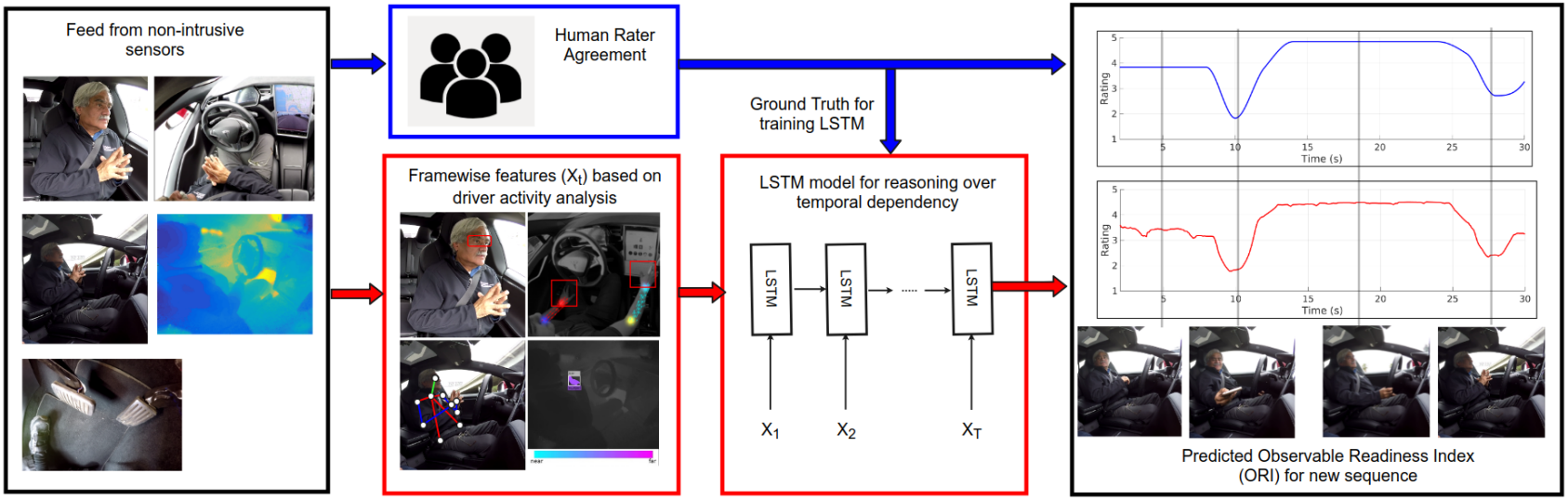


Figure 6.1. Overview of our approach: We wish to continuously estimate the driver's readiness to take-over control from an autonomous vehicle based on feed from vision and depth sensors capturing the driver's complete state. We define a continuous ground truth value for take-over readiness of the driver based on ratings provided by multiple human raters observing sensor feed. We term this the 'Observable Readiness Index (ORI)'.We process the sensor feed frame-by-frame using models for driver activity analysis and propose an LSTM model to learn the temporal dependencies in the frame-wise features. Our model continuously estimates the ORI of the driver.

main contributions are as follows:

1. **Naturalistic dataset with autonomous driving:** We capture a 2 hour 10 min dataset of drivers behind the wheel of a commercially available conditionally autonomous vehicle. This is captured using multiple high resolution cameras and depth sensors observing the driver. We use this data to train and evaluate our models. To the best of our knowledge, this is the first study evaluating take-over readiness of drivers using a naturalistic driving dataset from a conditionally autonomous vehicle.

2. **Human ratings for take-over readiness:** The goal of this work is to continuously estimate the take-over readiness of drivers using vision sensors. To test the feasibility of this approach, we collect ratings from multiple human raters viewing the sensor feed and analyze inter-rater agreement. Our experiments show a high consistency in the trend of the ratings across raters, rather than their absolute value. We normalize for rater bias using a percentile based approach. The mean value of the normalized ratings, averaged across raters, is then treated as the ground truth for our models. We term this the Observable Readiness Index (ORI).

3. **LSTM model for estimating take-over readiness:** We process the sensor streams frame by frame to analyze the drivers gaze [177], pose [23], hand [139, 186] and foot activity, giving a holistic representation of the driver's state. We propose a Long Short Term Memory (LSTM) network to model the temporal dependency of the frame-wise representations. The LSTM continuously outputs the driver's ORI based on 2 seconds of past activity. Additionally, the model can recognize key-frames from the sensor streams that are most predictive of the driver's take-over readiness.

6.2 Related Research

6.2.1 Driver behavior analysis

Driver behavior analysis based on vision sensors has been extensively addressed in prior research [18, 34, 51, 52, 91, 112, 122–124, 136, 138, 139, 164, 166, 168, 169, 173, 177, 178, 184, 186]. A large body of literature [51, 52, 91, 164, 166, 173, 177, 178] has focused on the driver's gaze estimation, being a useful cue for estimating the driver's attention. While early works [91, 165, 166] have relied on head pose estimation for determining the driver's gaze, most recent approaches use a combination of head and eye cues [46, 51, 52, 164, 173]. Recent work [177, 178] employing convolutional neural networks (CNNs) has allowed for generalizable estimation of driver gaze zones, across drivers and small variations in camera placement. Driver hand activity has also been the subject of a large number of research studies, being closely linked to the motor readiness of the driver. Common challenges involved in detecting the driver's hands such as fast varying illumination conditions, self occlusions, truncation have been outlined in [33]. Many approaches have been proposed for detection, tracking and gesture analysis of the driver's hands while addressing some of these challenges [18, 34, 112, 122–124, 136]. Recent CNN models proposed by Yuen *et al.* [186] and Rangesh *et al.* [139] allow for accurate localization of the driver's hands in image co-ordinates and in 3-D respectively, and allow for further analysis such as hand activity classification and detection of objects held by the driver. A few works have also addressed driver foot activity analysis [138, 168, 169], being a complimentary cue to hand activity, for estimation of the driver's motor readiness. There has also been significant research that builds upon cues from driver gaze, hand and foot analysis for making higher level inferences such as driver activity recognition [10, 19, 120], driver intent prediction [72, 73, 107, 121] and driver distraction detection [11, 94, 96, 97, 99, 181].

6.2.2 Driver distraction estimation

While very little literature exists on estimating the take-over readiness of drivers in autonomous vehicles, prior work [11, 94, 96, 97, 99, 181] has addressed the closely related problem of driver distraction estimation in manually driven vehicles. Driver distraction estimation poses the same key challenges as take-over readiness estimation: defining a ground-truth metric for the quantity being estimated, and proposing models that map driver behavior to this metric. Here, we review the approaches used in prior work for addressing these challenges. Bergasa *et al.* [11] extract face and gaze features such as PERCLOS [41], eye closure duration, blink frequency and face position and map them to the driver's vigilance level based on fuzzy rule expressions. Liang *et al.* [96, 97] and Liu *et al.* [99] define the ground-truth for driver distraction as a binary variable, determined based on experimental conditions. The driver is considered distracted for trials involving the driver performing a secondary activity and not distracted for baseline trials, not involving secondary activities. Binary classifiers trained on features capturing the driver's head and eye activity and driving performance to detect driver distraction, with support vector machines (SVMs) used in [97], Bayesian networks used in [96] and extreme learning machines (ELMs) used in [99]. Wollmer *et al.* [181] use subjective ratings provided by the drivers to define the ground truth distraction levels. They train an LSTM for classifying the ground truth distraction level using features based on driving performance and head pose of the driver. Li and Busso [94] use ratings provided by observers viewing the driver, rather than self evaluation by drivers. This allows for ratings that vary with time, and also allows for multiple raters to assign the rating. We use a similar approach for defining the ground-truth value for take-over readiness.

6.2.3 Take-over time and quality studies

A few recent studies [20, 54, 110, 146, 188] have addressed the case of conditionally autonomous driving under simulator settings. Gold *et al.* [54] analyzed reaction times and take-over quality of drivers prompted with a take-over request 5 seconds and 7 seconds before a hazardous event, showing that drivers achieve faster reaction times for the 5 second interval, but poorer quality of take-overs. Mok *et al.* [110] conducted a study with drivers performing a distracting secondary activity, prompted with take-over requests 2 sec, 5 sec and 8 sec before a hazardous event. Their results showed comparable take-over quality for the 5 and 8 second settings, while being considerably poorer for the 2 second setting. However the drivers reported greater trust in the automation for the 8 second setting. Rezvani *et al.* [146] also analyzed take-over quality and trust in the automation, with the focus being on user interface design for issuing take-over requests, rather than time before take-overs. Zeeb *et al.* [188] analyzed the relationship between driver gaze activity and take-over time and quality, showing that drivers preferring less frequent, longer glances at the secondary activity achieved slower take-overs, more prone to collisions, compared to drivers that preferred more frequent, but shorter glances. More recently, Braunagel *et al.* [20] presented a model for classifying take-over readiness of drivers based on driver gaze and encodings of the driver's secondary activity and situation complexity. They defined the ground-truth for take-over readiness as a binary value, based on the quality of the performed take-overs. Our work differs from this approach on two counts; we evaluate our models using naturalistic driving data, and we train our models on more detailed descriptors of the driver's behavior.

6.3 Experimental Setup

We use our testbed **LISA-T** [134], built on top of a Tesla Model S, for our experiments. The vehicle has automatic cruise control and auto-steer capabilities for

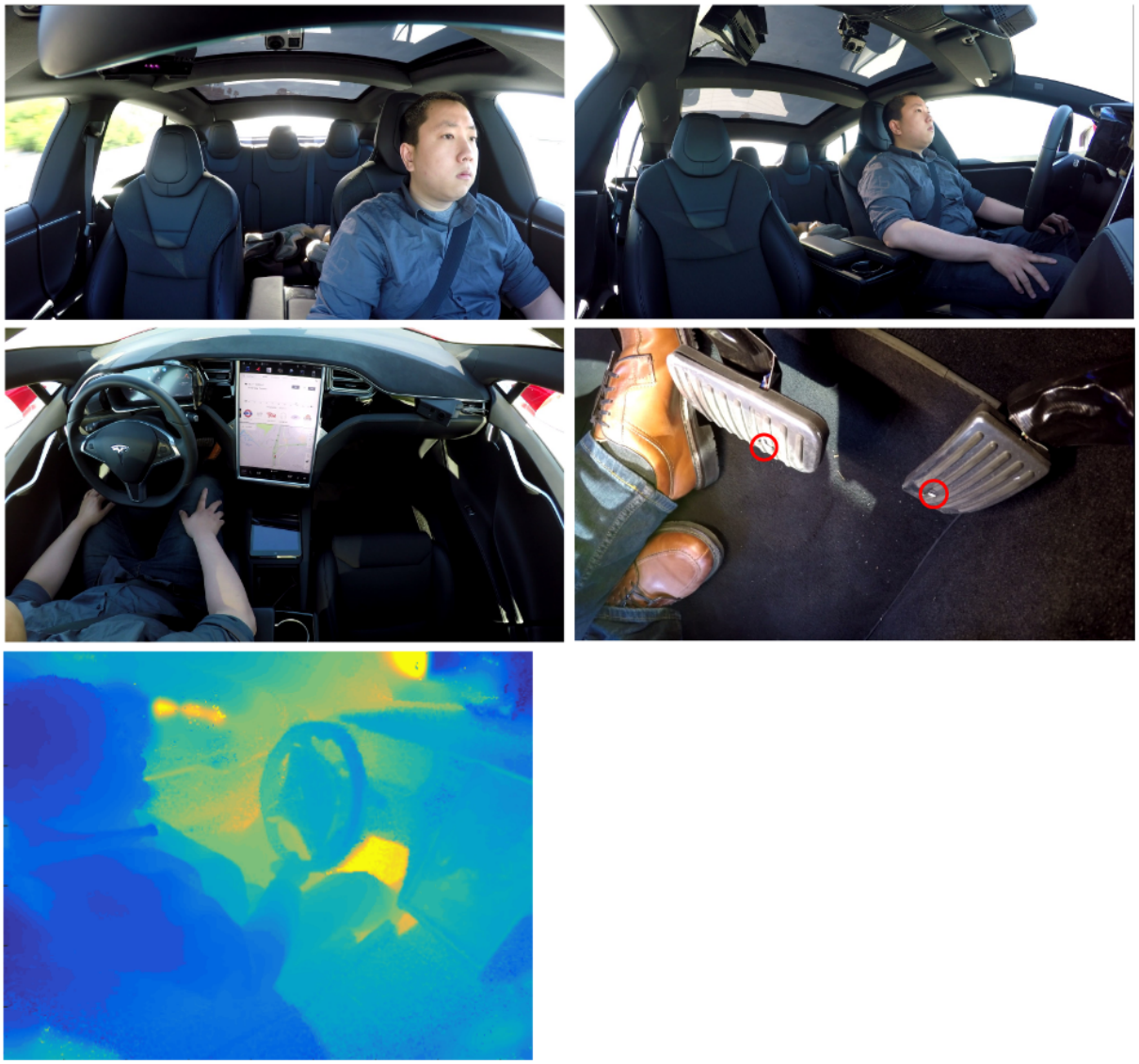


Figure 6.2. Experimental setup: Our testbed is equipped with 4 high resolution cameras, a depth sensor and infrared sensors for foot pedals. This figure shows views used for driver face and gaze analysis *(top-left)*, hand activity analysis *(middle-left)*, pose analysis *(top-right)*, foot analysis *(middle-right)* with IR sensor locations, depth sensor output *(bottom)*

freeway driving. The testbed is equipped with a suite of non-intrusive sensors monitoring the driver's state. Four high resolution cameras observe the driver's face, hands, body pose and feet. These allow for computer vision algorithms to analyze the driver's gaze activity, hand activity, objects held and body posture. Additionally, a depth sensor observing the driver allows for analysis of the 3-D distances of the driver's hands from the vehicle controls. Figure 6.2 shows the camera views and depth sensor outputs. The testbed is also equipped with infrared sensors on its brake and gas pedals to measure the distance of the driver's foot from each pedal. All sensors are synchronized and record at a frame rate of 30 frames per second.

6.4 Human ratings for observable driver readiness

We collect subjective ratings provided by multiple human observers viewing video feed from our testbed, where the raters assign a value for the driver's readiness to take-over. The human ratings serve as a sanity check for our approach based on non-intrusive sensors. A high degree of agreement across raters would point to a measure for driver take-over readiness that could be estimated purely based on visual cues. Secondly, the ratings address the problem of ground-truth generation. The average rating provided by the raters can be used as the ground truth for training and evaluating a machine learning algorithm to estimate this measure of take-over readiness.

6.4.1 Protocol for collecting ratings

We chose a pool of 5 raters with driving experience and working knowledge of the Tesla autopilot system for assigning the subjective ratings. The raters were shown multiple 30 second video clips of drivers with the car driving in autonomous mode. Figure 6.3 shows the interface used. The raters were shown synchronized feed from the pose camera and foot camera. These two views were chosen in order to capture the complete state of the driver with the minimum number of views, so as to not overwhelm the raters.

Figure 6.3. Interface for collecting ratings: The raters observe video feed from the pose and foot cameras and assign a rating for each 2 second segment of video.

The raters were given the following prompt: *"You will now be shown video clips of drivers behind the wheel of a vehicle operating in autonomous mode in freeway traffic. Rate on a scale of 1 to 5, with 1 being low and 5 being high, the readiness of the driver to take-over control from the vehicle for each 2 second video segment."* We chose a discrete rating scale rather than a continuous scale to minimize rater confusion and expedite the rating process. We used discrete, fixed length time steps rather than flexible time boundaries for the same reason. The raters were allowed to replay the segments as well as update their previous ratings.

To prevent rater fatigue, in a single session, a rater rated a maximum of 25 clips, each 30 seconds long. To account for the learning effect on raters, the first two clips in every session were used to prime raters for the task. The ratings for these two clips were discarded. The two priming clips were chosen such that one clip had primarily vigilant behavior, typically rated high by the raters, and one clip had the driver performing a distracting secondary activity, typically rated low.

6.4.2 Dataset Description

The complete dataset includes 260 clips, each 30 seconds long, amounting to 2 hours and 10 minutes of data. Among these clips, 20 were rated by the entire pool of 5 raters. We refer to this subset as the *common set*. We use the common set to analyze the agreement across raters and to normalize for rater bias. The remaining 240 clips were rated by 2 different raters from the rater pool. We refer to this set as the *expansion set*. We use both, the common set and the expansion set, for training and evaluating our models.

The entire dataset involves naturalistic drives with 11 drivers, with the car operating in autonomous mode on Californian multi-lane freeways. 7 drivers were male, while 4 were female. 5 drivers were in the age group of 20 to 30 years, 3 in the age group of 30 to 40, 1 in the age group of 40 to 50, and 2 drivers were over 50 years of age. All drivers were familiarized with the Tesla auto-pilot functionality prior to data collection. The data includes a wide range of driver behavior including vigilant driving, talking to a co-passenger, gesturing, operating the infotainment unit, drinking a beverage and interacting with a cell-phone or tablet. Table 6.1 lists the observed secondary activities in our data, in terms of the nomenclature used in the SHRP2 NEST database [125]. Figure 6.4 shows example frames from the pose view.

6.4.3 Normalization of ratings

One source of noise in the assigned ratings is rater bias. Raters can be strict or lax, and can use a varying range of values. We normalize for rater bias using a percentile based approach. We use the common set for normalization of the ratings. We pool and sort ratings provided by each rater on the common set to obtain rater specific look-up tables. We then pool and sort ratings of all raters to obtain a combined look-up table. To normalize a specific rater's ratings, we find the percentile range of the assigned value in the rater's look-up table. We then replace it with the average of all values in that percentile

Figure 6.4. Examples frames from the dataset (showing pose view): The dataset driver behaviors such as vigilant driving, talking to a co-passenger, gesturing, operating the infotainment unit, drinking a beverage and interacting with a cell-phone or tablet

range in the combined look-up table. This percentile based lookup can be applied to the entire dataset, including the expansion set.

6.4.4 Observable Readiness Index

We average the normalized ratings across raters to give a single rating for each 2 second interval in the clips. To obtain a continuously varying value representing the driver's take-over readiness, we interpolate these values over all frames using bi-cubic interpolation. We term this normalized and interpolated value the Observable Readiness Index (ORI), since it was assigned based purely on observable cues.

6.4.5 Inter-rater agreement analysis

We use intraclass correlation co-efficients (ICCs) as formulated by McGraw *et. al.* [108], to evaluate inter-rater agreement. We model the human ratings as a two-way

Table 6.1. Secondary activities in collected dataset

Secondary task based on SHRP2 NEST nomenclature [125]	Present in collected data?
Talking/Singing to Self	✓
Talking/Singing to Passenger(s)	✓
Dancing	✗
Reading	✓
Writing	✗
Holding object	✓
Manipulating object	✓
Reaching for object	✓
Talking/listening on handheld phone	✓
Adjusting steering wheel buttons	✓
Adjusting/monitoring center stack controls	✓
Adjusting/monitoring other devices	✓
Looking for object internal to vehicle	✓
Looking at object external to vehicle	✓
Eating/drinking	✓
Grooming (combing hair, removing glasses)	✓

random-effect model without interaction, assuming n observations and k raters. Under this model, the rating x_{ij} assigned by rater j to clip i can be expanded as,

$$x_{ij} = \mu + r_i + c_j + e_{ij}, \tag{6.1}$$

where, μ is the global average rating, r_i's are the deviations based on the content of the rated clips, and c_j's are the deviations due to rater bias. The r_i's and c_j's are independent, with mean 0 and variance σ_r^2 and σ_c^2 respectively. And finally, e_{ij} is the normally distributed measurement error with zero mean and variance σ_e^2

We report the following ICC values for the normalized and unnormalized ratings, as defined in [108]:

- *ICC(C,1):* This is given by the expression

$$ICC(C,1) = \frac{\sigma_r^2}{\sigma_r^2 + \sigma_e^2}, \tag{6.2}$$

and can be interpreted as the degree of consistency of the rating values. This is independent of the rater bias, and has a high value if the trend of ratings across raters is consistent.

- *ICC(A,1):* This is given by the expression

$$ICC(A,1) = \frac{\sigma_r^2}{\sigma_r^2 + \sigma_c^2 + \sigma_e^2}. \tag{6.3}$$

This is the degree of absolute agreement of rater values, and has a high value only if the raters are in agreement in terms of the actual value of the ratings.

- *ICC(A,k):* This is given by the expression

$$ICC(A,k) = \frac{\sigma_r^2}{\sigma_r^2 + \frac{\sigma_c^2 + \sigma_e^2}{k}}. \tag{6.4}$$

This can be interpreted as the reliability of the average rating provided by k different raters. In our case, $k = 5$ for the common set, and $k = 2$ for the expansion set.

All ICC values are bounded between 0 and 1. The σ values are estimated using two-way analysis of variances (ANOVA). Koo and Li [85] prescribe that ICC values less than 0.5, between 0.5 and 0.75, between 0.75 and 0.9, and greater than 0.90 are indicative of poor, moderate, good, and excellent reliability, respectively.

Table 6.2 shows the ICC values for the common and expansion sets, with and without normalization. As expected, the ICC(C,1) values are higher than the ICC(A,1) values due to the rater bias term σ_c^2 in the denominator for ICC(A,1). However, we note that normalization considerably improves the ICC(A,1) values for both the common and

Table 6.2. Rater agreement analysis based on intra-class correlation co-efficients (ICC)

Dataset	Normalization	ICC(C,1)	ICC(A,1)	ICC(A,k)
Common Set		0.584	0.415	0.780
	✓	0.580	0.582	**0.874**
Expansion Set		0.608	0.517	0.682
	✓	0.637	0.627	**0.772**

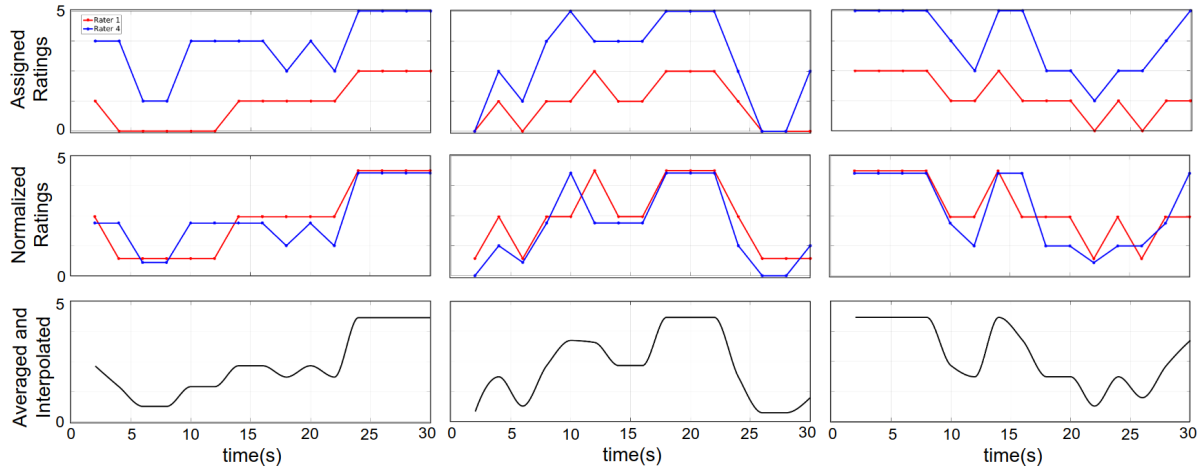


Figure 6.5. Example ratings: Assigned (*top*), normalized, (*middle*) and averaged and interpolated (*bottom*) ratings provided by two raters for 3 sequences from the expansion set. The percentile based normalization scheme removes rater bias while retaining the trend of the ratings. Finally averaging and interpolating gives the continuously varying ORI for the sequences

expansion sets, without affecting the ICC(C,1) values. This shows that the normalization maintains the trend (σ_r^2) of the ratings while reducing rater bias (σ_c^2). Finally, the last column shows the ICC(A,k) values, which represent the reliability of the average rating provided by all raters. As expected, this value is higher for the common set, rated by 5 raters as compared to the expansion set rated by 2 raters. However, both sets after normalization have an ICC(A,k) rating that falls within the range indicative of 'good reliability' of the ORI values as prescribed by Koo and Li [85].

6.4.6 Qualitative analysis of ratings

Figure 6.5 shows 3 examples of ratings from the expansion set. The top row shows the ratings assigned by raters 1 and 4. We observe that rater 1 is strict and rarely assigns a rating greater than 3. On the other hand, rater 4 is much more liberal, primarily assigning ratings from 3 to 5. However, we can observe the similarity in the trend of the ratings assigned by the two raters. The middle row shows the ratings after applying the percentile based normalization. We observe that normalization reassigns the ratings to a similar range of values while maintaining their trend, thereby removing rater bias. Finally, the bottom row shows the continuously varying ORI, obtained after averaging and interpolation of the normalized ratings.

6.5 Model for Estimating ORI

We wish to continuously estimate the driver's ORI based on recent feed from the sensors observing the driver. We pose this as a regression problem. At each instant, we treat the past two seconds of sensor feed as the input and the ORI value as the output.

The raw sensor feed has high dimensionality due to the multiple high resolution cameras and depth sensors. To reduce the input dimensionality, we use existing models for driver behavior analysis [23, 139, 177, 186] based on convolutional neural networks (CNNs). These models perform frame-wise analysis of the camera and depth sensor feed to extract features corresponding to driver gaze, hand activity, foot activity and pose. Figure 6.6 summarizes the inputs and outputs of the frame-wise feature extraction models. Since all sensor streams are synchronized, the features can be concatenated to obtain a frame-by-frame representation of the driver's state.

While the CNNs extract a holistic frame-wise representation, driver activity spans across multiple frames. We thus need a model that can reason about the temporal dependencies of the features to estimate the ORI value. Due to their effectiveness at

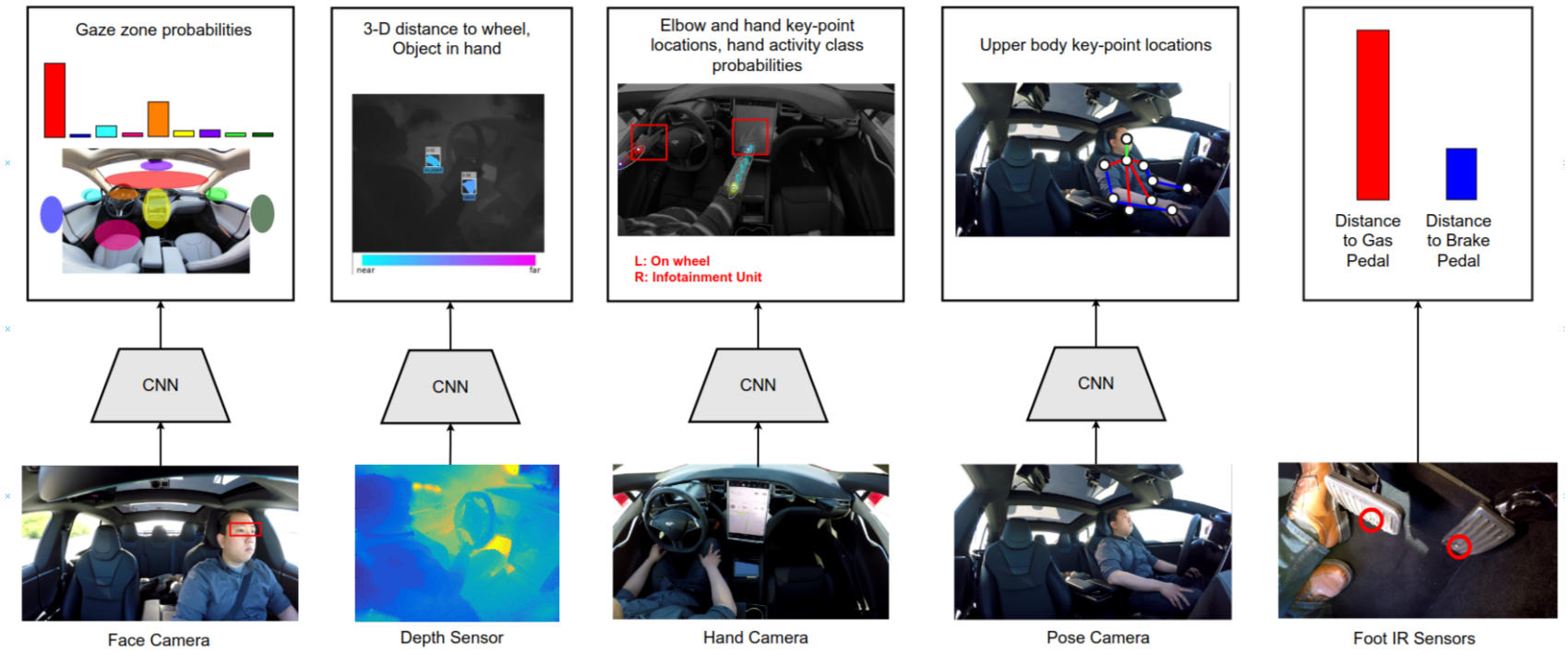


Figure 6.6. Frame-wise features capturing driver state: We extract frame-wise features capturing driver's gaze, hand activity, pose and foot activity from the synchronized feed of our cameras and depth sensors. Existing convolutional neural network (CNN) based approaches [23, 139, 178, 186] are used for extracting these frame-wise features.

modeling long-term temporal dependencies in various sequence modeling tasks [43, 56, 57, 102, 187], we propose a model based on Long Short-Term Memory (LSTM) networks [66] for estimating ORI.

6.5.1 Frame-wise feature extraction

Gaze Analysis: We use the model proposed by Vora *et al.* [177] for gaze analysis. The inputs to the model are frames from the face camera. The face detector described in [184] is used for accurate localization of the driver's eye region. The cropped image of the driver's eye region is then passed through a CNN classifier, which outputs the driver's gaze zone. We consider 9 different gaze zones: {*forward, left shoulder, left mirror, lap, speedometer, infotainment unit, rear-view mirror, right mirror, right shoulder*} shown in Figure 6.6. The CNN uses a softmax output layer, giving frame-wise probabilities for each gaze zone. We use this 9 dimensional vector to represent driver gaze for each frame.

Hand Analysis (Camera-based): We use the model proposed by Yuen and Trivedi [186], for hand analysis based on feed from the hand camera. The model localizes the elbow and wrist joints of the driver using part affinity fields [23]. The model is trained for robustness under rapidly changing lighting conditions and variability in hand posture during naturalistic drives. Additionally, the model classifies hand activity for each hand using a CNN classifier. The CNN uses as input a cropped image near the localized wrist joints and outputs a hand-activity class. Four activity classes are considered for the left hand: {*on lap, in air (gesturing), hovering wheel, on wheel*}. Six classes are considered for the right hand: {*on lap, in air (gesturing), hovering wheel, on wheel, interacting with infotainment unit, on cup-holder*}. We obtain an 18 dimensional feature vector corresponding to the x and y co-ordinates of the 4 joints, the 4 dimensional output of the left hand activity classier and the 6 dimensional output of the right hand activity classifier for each frame of the hand camera.

Hand Analysis (Depth-based): We use HandyNet, proposed by Rangesh and Trivedi

[139], for hand analysis using the depth sensor. The model performs instance segmentation on each frame of the depth sensor to output the locations of the driver's hands. This allows for computation of the distance of the driver's hands from the vehicle controls in 3-D. Additionally, the model also classifies the object held by the driver, if any. We consider 4 different object classes: {*no object, cell-phone/tablet, beverage/food, other item*}. Thus, for each frame of the depth sensor, we obtain a 5 dimensional feature vector, corresponding to the distance to the wheel, and the 4 dimensional output of the hand object classifier.

Pose Analysis: The driver's body posture can be a useful additional cue for estimating their readiness to take over. We capture the driver's pose for each frame by using OpenPose [23] proposed by Cao *et al.* The model is capable of localizing 18 body key-point locations. For our task, we only consider the the 10 upper body key-points visible in the pose camera view, as shown in Figure 6.6. The x and y co-ordinates of the 10 body key-points give a 20 dimensional feature vector for each frame of the pose camera.

Foot Analysis: To obtain a representation of the driver's foot activity, we use the outputs of the IR pedal sensors. These give the distance of the driver's foot to the gas and brake pedal for each frame.

6.5.2 Correlation of extracted features with ORI

Figure 6.7 shows the frame-wise correlation of features with the ORI values. We note that in general, hand features seem to have higher correlation with ORI compared to gaze or foot features. In particular, the presence of a hand-held phone or tablet has a strong negative correlation with the rated ORI values. On the other hand, the driver's hands being free (without holding any object) has a high positive correlation with ORI. In terms of activity classes, gesturing and interacting with the infotainment unit are negatively correlated with ORI, whereas the hands hovering or being on the wheel are positively correlated. Although the gaze features have a lower correlation with ORI as compared to hand features, we note that the gaze zones corresponding to sensing the vehicle's

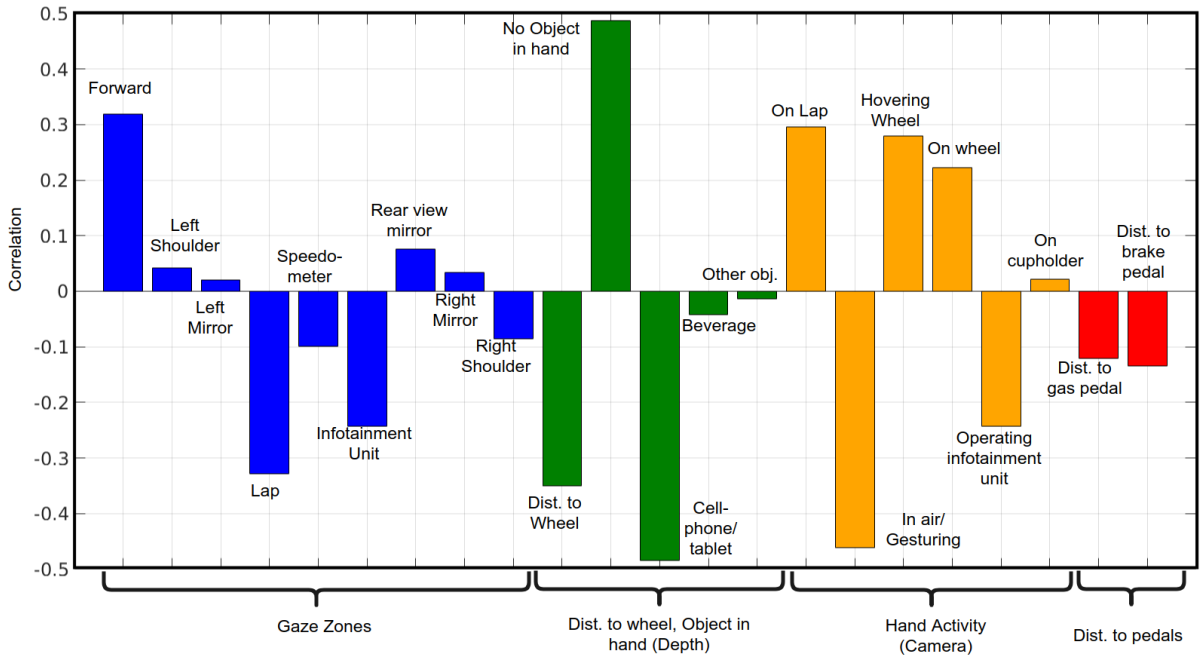
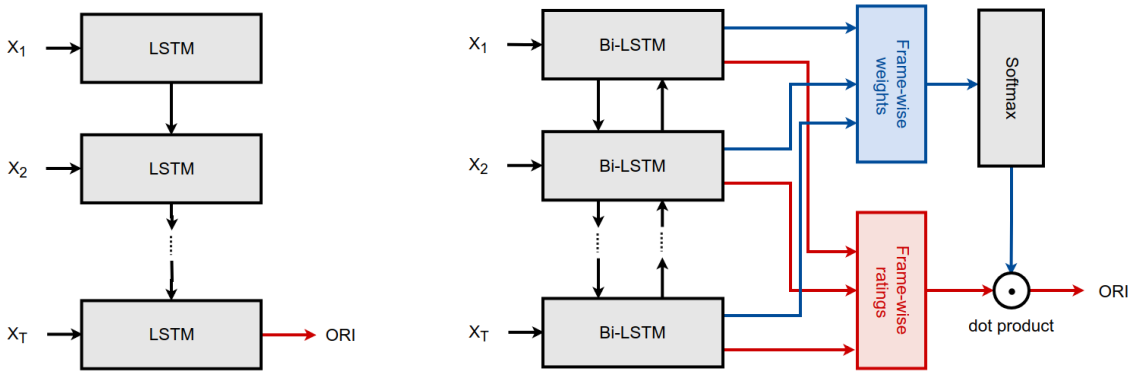


Figure 6.7. Feature correlation: Frame-wise correlation of gaze, hand and foot features with ORI ratings

surroundings such as looking forward or viewing the mirrors are correlated positively with ORI. On the other hand, gaze zones corresponding to driver distraction such as looking at the lap (or hand-held device), infotainment unit or speedometer have negative correlation. The only exception to this seems to be the right shoulder gaze zone, which is negatively correlated with ORI. This could be because the driver may look to the right to interact with the co-passenger in the front seat, which can be seen as a distracting activity. Finally, we note that the distances of the hands and feet from the vehicle controls are negatively correlated with the rated ORI, which is reasonable since these affect the motor-readiness.

6.5.3 Proposed LSTM model

In its simplest form, we can define an LSTM model for processing the sequence of frame-wise features as shown in Figure 6.8a. An LSTM is a recurrent neural network. It updates its state at each time-step based on its state at the previous time-step and the frame-wise features at the current time-step. The final state of the LSTM can be



(a) Vanilla LSTM **(b)** LSTM with key-frame weighting

Figure 6.8. Models: LSTM models used for estimating ORI

expected to encode the driver's activity spanning over 2 seconds of context by processing the temporal dependencies of the frame-wise features. The final state of the LSTM is then processed by an output layer to give the scalar valued ORI.

Key-frame weighting

While the LSTM reasons over the complete 2 second sequence of frame-wise features, driver activity in certain parts of the sequence can be more strongly predictive of their readiness to take over. For example, a driver may for a split second, interact with their phone or the infotainment unit, while otherwise appearing vigilant in terms of hand, eye and foot activity. While this activity may not span 2 seconds, it is indicative of driver distraction and low readiness to take over. Human raters viewing the driver would take this into consideration and weight this part of the video clip more strongly for assigning their ratings. We thus need a model capable of assigning greater weight to key-frames in the sequence of features, to determine the final rating.

We propose the model shown in Figure 6.8b for key-frame weighting. The LSTM outputs two values at each time step, the rating for that frame and the weight for that frame. The frame-wise weights are normalized using a softmax layer. A dot product of the normalized weights and the frame-wise ratings gives the ORI value for the sequence.

This allows the model to assign a greater weight to key-frames in the sequence of features. We use a bidirectional LSTM [58] instead of a unidirectional LSTM. This allows for information to flow from the entire 2 second sequence to LSTM states at each time step. This allows the model to relatively weigh the current frame with respect to the other frames in the sequence in order to determine the frame-wise weight. Note that we do not provide explicit supervision in terms of key-frames in the sequence, but incorporate a mechanism in the model to learn them, via the dot product between frame-wise ratings and frame-wise weights. The model is trained end-to-end to minimize the error between the predicted rating and the assigned rating. Our approach can be considered similar to the attention mechanism [6] in LSTM encoder-decoder models, with an attention based decoder applied for a single step.

Implementation details

We use bi-directional LSTMs with a 32 dimensional state vector for both the forward and backward LSTMs. We use a sigmoid activation for the frame-wise ratings, giving a predicted ORI value between 0 and 1. The ground truth ORI ratings ranging from 1 to 5 are shifted and scaled to map them to the 0 to 1 range. We train the model by minimizing mean absolute error on the train set. We use Adam [82] with learning rate 0.001 to train the model. The models are trained using Pytorch [127].

6.6 Experimental Evaluation

For our experiments, we split our dataset of 260 video clips, into a train set, validation set and a test set. The train set consists of 172 video clips from 22 different drives, the validation set consists of 32 video clips from 5 drives and the test set consists of 56 clips from 6 drives. The train, validation and test sets were selected from different drives. The test set includes data from 6 different drivers, 3 of whom were not present in the train or validation sets. Since each clip is 30 seconds long, with the sensors recording

at 30 fps, this gives us about 154,000 training samples, 28,000 validation samples and 50,000 test samples. We train our models to minimize the mean absolute error between the predicted and assigned ORI values on the train set. We select the model with the minimum mean absolute error on the validation set for reporting results on the test set.

6.6.1 Metrics and baselines

We report the mean absolute error (MAE) between the predicted and assigned ORI values for the test set. We compare the following models:

- *Frame-wise SVM baseline:* We use a linear SVM trained purely using the frame-wise features as the simplest baseline. This allows us to evaluate the effect of using temporal context for estimating ORI.
- *Vanilla LSTM:* We report results using a vanilla LSTM as shown in Figure 6.8a.
- *LSTM with key-frame weighting:* We additionally consider a model with key-frame weights, but with unidirectional LSTMs
- *Bi-LSTM with key-frame weighting:* Finally, we report results using our complete model with bidirectional LSTMs and key-frame weights shown in Figure 6.8b.

For a fair comparison with our proposed model using bidirectional LSTMs, we use LSTMs with 64 dimensional states for the unidirectional LSTMs, while using LSTMs with 32 dimensional states for the forward and backward components of the bidirectional LSTM. Additionally we perform ablations with the feature streams to analyze the importance of gaze, hand, pose and foot features for estimating the driver's ORI.

6.6.2 Results

Table 6.3 shows the MAE values for the three models compared, trained using one or more feature streams. We note that the LSTM models achieve a significantly lower

MAE than the frame-wise baseline for most combinations of feature streams, showing that modeling temporal context is useful for estimating the ORI. We also note that key-frame weighting consistently leads to lower errors for different features, compared to the vanilla LSTM, with the Bi-LSTM outperforming the uni-directional LSTM. The top four rows of Table 6.3 shows the MAE values for models trained on a single feature stream of gaze, hand, pose or foot features. All models achieve an MAE of within 1 point of the average rating assigned by the human raters on the 5 point scale. This shows that each feature stream has useful cues for estimating ORI. Comparing the features, we see that the hand features seem to have the most useful cues for estimating ORI, closely followed by the gaze features. The last three rows of the table show the MAE values achieved by combining the feature streams. We note that combining the hand and gaze features significantly lowers the MAE compared to single feature models. The pose and foot features lead to further improvements, giving the lowest MAE value of 0.449.

6.6.3 Inference time

The results reported in section 6.6.2 were generated offline at the data capture rate of 30 Hz. However, we report the inference times for all feature extractors and the LSTM model here for completeness, since the proposed model would need to be deployed real time in conditionally autonomous vehicles. Table 6.4 shows the average inference times in ms for the components of our model, obtained using an Nvidia GTX 1080 Ti system. We note that the camera based hand [186] and gaze analysis [178] models run faster than the data capture rate. However, the rate determining step for gaze analysis is the face detector [184] used for localizing the driver's eyes, whereas the rate determining step for hand analysis is the depth based CNN [139]. The LSTM model for estimating ORI is a lightweight model with very little additional run time. Thus, the models could be deployed at about 15 Hz, which is half the data capture rate.

Table 6.3. Mean absolute error (MAE) of predicted ORI values with respect to assigned values

Features used				Frame SVM	Vanilla LSTM	LSTM keyframe wts	Bi-LSTM keyframe wts
Gaze	Hand	Pose	Foot				
✓				0.779	0.621	**0.578**	0.581
	✓			0.639	**0.571**	0.573	0.572
		✓		0.836	0.855	0.831	**0.823**
			✓	**0.986**	1.018	1.043	1.001
✓	✓			0.611	0.470	0.463	**0.457**
✓	✓	✓		0.602	0.468	0.463	**0.456**
✓	✓	✓	✓	0.699	0.467	0.456	**0.449**

Table 6.4. Average inference times for components of our model

Component	Run time
Face detector	62 ms
Gaze analysis	6 ms
Hand analysis (Camera based)	25 ms
Hand analysis (Depth based)	55 ms
Pose analysis	45 ms
Foot analysis	-
LSTM for estimating ORI	3 ms

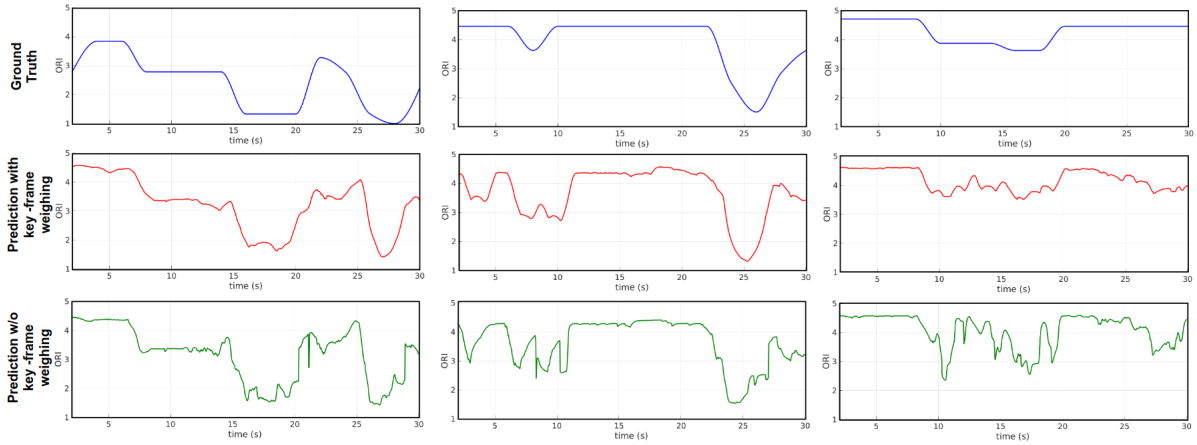


Figure 6.9. Effect of key-frame weighting model: Three example clips with ground truth ORI (top), ORI predicted by vanilla LSTM (bottom), ORI predicted with key-frame weighting (middle). Key-frame weighting allows the model to focus on the most relevant frames in the sequence and generate a smoother, more reliable rating, compared to the noisier, more reactive vanilla LSTM.

6.7 Qualitative analysis

6.7.1 Effect of key-frame weighting model

We qualitatively analyze the ratings produced by a vanilla LSTM and our proposed model. Figure 6.9 shows ratings for three example clips. The top row (blue) shows the ground truth ratings, provided by the raters. The middle row (red) shows ratings predicted by the model with key-frame weighting. The bottom row (green) shows ratings predicted by a vanilla LSTM model. We observe that while both models manage to predict the trend of the ground truth ratings, the vanilla LSTM produces a noisy estimate, rapidly varying in short intervals. On the other hand the model with key-frame weighting produces a much smoother rating. This can be explained by the sparsity of key-frames in the sequence. Since the model primarily relies on a subset of the most important sections of the sequence to assign an ORI rating, it leads to a smoother, more reliable output throughout the sequence compared to the vanilla LSTM, which is more reactive.

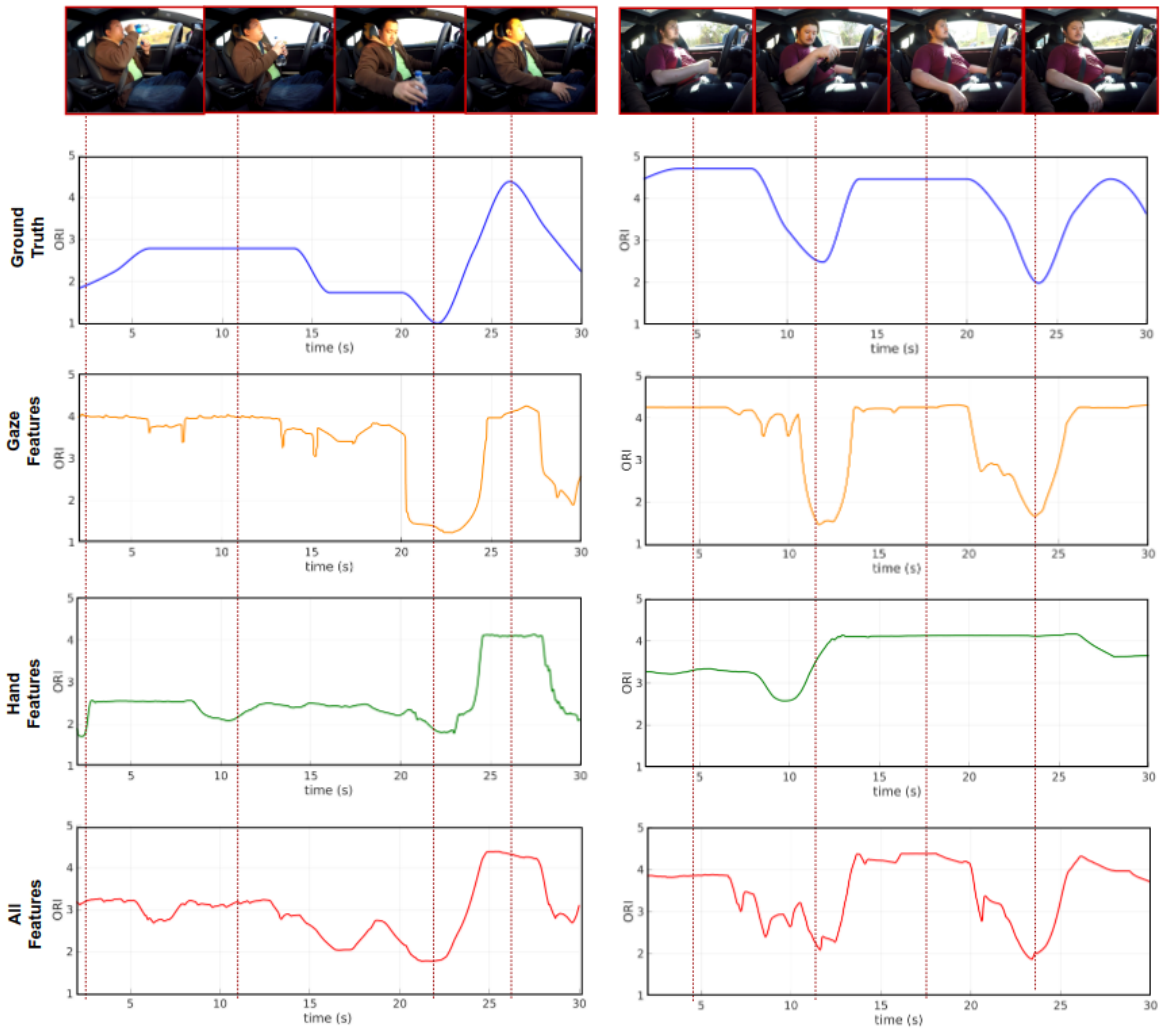


Figure 6.10. Importance of gaze and hand cues: This figure shows two example clips with ratings predicted based purely on gaze cues, hand cues and all features combined. The first example shows a case where the gaze features fail to correctly predict the ORI, where the hand features can be used to correct the error. The second example shows a failure case of hand features, that could be corrected based on the gaze features. The model trained on the combined feature streams correctly predicts the ground truth rating for both cases.

6.7.2 Effect of feature streams

We also compare the ratings generated by models trained purely using the gaze or hand features, with those generated by the combined feature stream model to analyze the importance of each cue for estimating the ORI. Figure 6.10 shows the ratings for two example clips. The top row (blue) shows the ground truth ratings, provided by the raters. The second (yellow) and third (green) rows show ratings predicted based purely on gaze and hand features respectively. Finally, the bottom row (red) shows the ratings predicted by the model trained using all feature streams. Additionally, we also show snapshots from the pose camera at different instants to provide context.

The first example shows the driver drinking water from a bottle. For the interval from 0-20 seconds in the clip, the driver has the bottle in their hand. However, they have their eyes on the road. We note that the gaze model overestimates the ORI value for this interval, while the hand model underestimate the ORI. The rating from the combined model closely matches the ground truth value. At 22 seconds, the driver has a bottle in their hand while looking away from the road to put it away. All models assign a low value for this instant. At 26 seconds, the driver is attentive, with their hands close to the wheel. All models correctly assign a high rating at this instant.

The second example shows the driver viewing the screen of a hand held device, and later viewing the screen of the infotainment unit. We see that both events correspond to dips in the ORI ratings assigned by the raters. We observe that the hand model correctly assigns a low rating for the first case, due to the presence of a hand held device. However, it fails to capture the dip in ORI due to the driver glancing at the infotainment unit. The gaze and combined models, on the other hand correctly predict both dips in the ORI value.

6.8 Conclusions

In this chapter, we proposed an approach to characterize the observable take-over readiness of drivers in autonomous vehicles, and a machine learning model to estimate it.

We collected subjective ratings for driver take-over readiness from observers viewing the sensor feed. Analysis of the assigned ratings in terms of intra-class correlation coefficients showed high consistency in the ratings assigned by the raters, suggesting that human raters agree upon an observable measure for the driver's take-over readiness. We normalized the ratings for rater bias and averaged across raters to extract this measure, which we termed the observable readiness index (ORI).

We presented a model for estimating the ORI, based on CNNs for extracting frame-wise representations of the driver's gaze, hand , pose and foot activity and an LSTM for learning their temporal dependencies. Our best model achieved an MAE of 0.449 on the five point rating scale, showing that the ORI can be reliably estimated. Ablation of feature streams showed the usefulness of hand, gaze, pose and foot activity analysis for estimating the ORI. Finally, we proposed a modification to vanilla LSTMs, to allow for detecting key-frames in the input sequence most predictive of the driver's take-over readiness. Our experiments showed that this leads to lower MAE of the predicted ORI, and leads to a smoother, more reliable prediction.

Acknowledgements

Chapter 6, in full, is a reprint of the material as it appears in: "Looking at the driver/rider in autonomous vehicles to predict take-over readiness," Nachiket Deo and Mohan M. Trivedi, IEEE Transactions on Intelligent Vehicles 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Predicting Take-over Time for Autonomous Driving with Real-World Data

7.1 Introduction

In this chapter, we address *take-over time* (TOT), defined as the interval of time between a take-over request (TOR) and the human driver assuming control. More specifically, we define the assumption of control as the completion of three behaviors:

1. **Hands-on-wheel:** hand(s) return to the vehicle's steering control.
2. **Foot-on-pedal:** foot returns (from floorboard or hovering) to make contact with any driving pedal.
3. **Eyes-on-road:** gaze is directed forward, toward the active driving scene.

We work with the assumption that these three cues are *necessary* to consider the driver both attentive to the scene and in control of the vehicle. We do note that the three cues may not be *sufficient* to consider the driver attentive and in control. This would additionally depend on factors such as the driver's situational awareness and the corrective or stabilizing maneuver performed post TOR. We limit the scope of this chapter to predicting the time taken for the above three cues, as a first step towards analysis of control transitions using

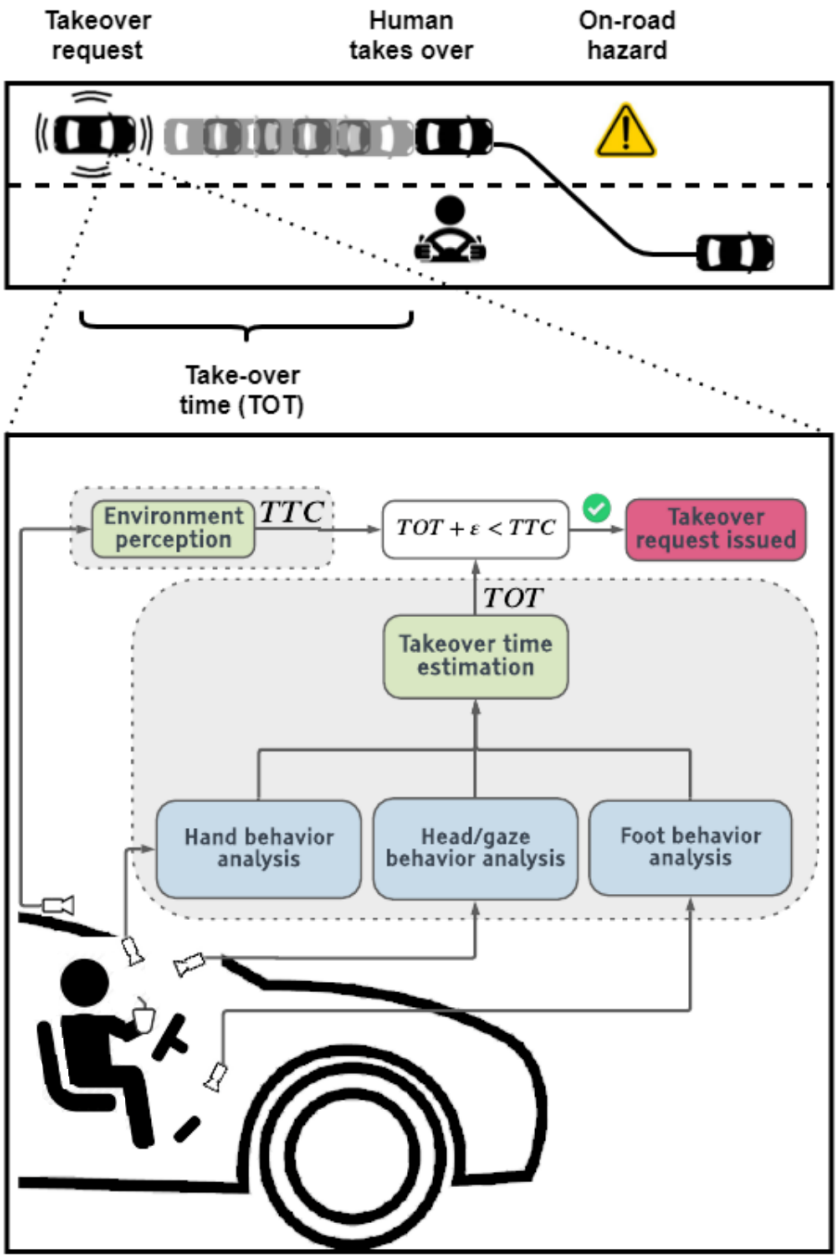


Figure 7.1. Role of take-over time (TOT) prediction: We propose a model for predicting TOT during control transitions based on driver behavior. The proposed model can be used in conjunction with time-to-collision estimation to determine whether to issue a take-over request and transfer control to the human, or to deploy active safety measures for collision avoidance.

real-world autonomous driving data. Analysis of situational awareness and corrective maneuvers will be addressed in future work.

As depicted in Fig. 7.1, the transition of control from an autonomous agent to the human driver should be a function of both the surrounding scene and the state of the driver. The surrounding scene can be concisely expressed using a metric such as time-to-collision (TTC), whereas the state of the driver can be captured by the predicted TOT. Combined, this forms a criterion for safe control transitions:

$$TOT + \varepsilon < TTC, \tag{7.1}$$

where ε is a marginal allowance that represents the time it takes for the human driver to gain situational awareness and perform a corrective maneuver. A system that takes the state of the driver into account can decide between handing over control if the driver is ready, versus coming to a safe and smooth halt if not. While there are many approaches to accurately estimate TTC, TOT prediction (especially in the real world) remains unexplored.

7.1.1 Contributions

In this chapter, we present a long short-term memory (LSTM) model for predicting TOT based on driver behavior prior to the TOR. We train and evaluate our model using a real world dataset of control transitions captured using a commercially available conditionally autonomous vehicle. Our contributions are as follows:

1. **TOT prediction with limited real-world data:** Capturing real-world takeover events in autonomous vehicles is expensive and time-consuming. Thus generating a large enough dataset for training machine learning models can be a challenge. To address this, we propose a data-augmentation scheme to increase the number of training samples by an order of magnitude. Additionally we use transfer learning, and pre-train our TOT prediction models to estimate the driver's observable take-over

readiness index (ORI) [39].

2. **Multimodal TOT prediction:** There is inherent uncertainty in predicting the future. The driver could perform multiple plausible sequences of actions after the issued TOR. To model this, we extend the model proposed in [133] to output a multimodal distribution over TOT.

3. **Extensive evaluation:** We present a more extensive set of ablation experiments, particularly focused on the above two contributions. We also present additional qualitative analysis of TOT estimates beyond [133].

7.2 Related Research

7.2.1 Vision based driver behavior analysis

A large body of literature has addressed driver behavior analysis using in-cabin vision sensors. The most commonly addressed task is driver gaze estimation [51, 52, 74, 91, 114, 115, 143, 164, 166, 173, 177, 178], since the driver's gaze closely relates to their attention to driving and non-driving tasks. Early works relied on head pose estimation [91, 114, 165, 166] or a combination of head and eye features [46, 51, 52, 164, 173] for estimating the driver's gaze. More recent work [74, 115, 143, 177, 178] uses convolutional neural networks (CNNs) to directly map regions around the driver's eyes to gaze zones. In this work, we use the CNN model proposed by Vora *et al.* [178] driver gaze analysis.

Driver hand and foot activity has also been the subject of prior work, being useful cues to gauge the driver's motor readiness. Several approaches have been proposed for detection, tracking and gesture analysis of the driver's hands [18, 33, 34, 112, 122–124, 136] using in vehicle cameras and depth sensors. Recently proposed CNN models [139, 186] accurately localize the driver's hands in image co-ordinates and in 3-D respectively, and further classify hand-activity and held objects. We build upon the model proposed by Yuen *et al.* [186] in this work, for driver hand analysis. Relatively few works have addressed

the driver's foot activity [138, 168, 169]. However, we believe this is a significant cue for TOT estimation, especially since we estimate the foot-on-pedal time after the TOR. We use the model proposed by Rangesh *et al.* [137] for driver foot activity analysis.

There has also been significant research that builds upon cues from driver gaze, hand and foot analysis for driver activity recognition [10,19,120,151,152], driver intent or behavior prediction [44, 47, 72, 73, 107, 121, 160] and driver distraction detection [11, 94, 96, 97, 99, 181]. Of particular interest is recent work [39], where the authors map driver gaze, hand and foot activity to the driver's observable take-over readiness index (ORI) obtained via subjective ratings assigned by multiple human observers. We use ORI estimation as a transfer learning task for pre-training our TOT prediction model.

7.2.2 Take-over time analysis in autonomous driving

Take-over time in partial and conditionally autonomous vehicles has been the subject of several recent studies [29, 42, 50, 54, 55, 70, 86, 111, 116, 128, 132]. The primary focus of these studies has been to analyze the effect of various human and environmental factors on take-over time and quality. The independent variables analyzed for their effect on TOT are as follows:

TOT budget (or time to collision): This corresponds to the time window between the TOR and the imminent collision or system boundary. Gold *et al.* [54] compare TOT and take-over quality for two different TOT budgets of 5s and 7s. They report longer TOTs for the 7s budget but better take-over quality. Mok *et al.* [111] report a similar finding while comparing TOT budgets of 2s, 5s, and 8s, with the 2s case corresponding to significantly worse take-over quality and collision rates.

Traffic density: Radlmayr *et al.* [132] and Gold *et al.* [55] analyze the effect of traffic density on TOT and take-over quality, with both studies reporting longer TOTs and worse take-over quality in situations involving high traffic density.

Driver age: Korber *et al.* [86] and Clark and Feng [29] analyze the effect of driver age on TOT by comparing a group of young drivers with a group of old drivers. Korber *et al.* [86] report similar TOTs, but different modus operandi – older drivers brake harder and more often leading to higher TTC. Clark and Feng [29] report lower TOTs for the young group for a TOT budget of 4.5s, and lower TOTs for the old group for a 7.5s TOT budget.

TOR modality: Petermeijer *et al.* [128] and Huang *et al.* [70] compare different modalities for issuing the TOR. Auditory and tactile TORs are considered in [128] while auditory, tactile and visual TORs and their combinations are considered in [70]. Both studies report the lowest TOTs for multimodal TORs. Dogan *et al.* [42] analyze the effect of providing the driver anticipatory information about the vehicle and traffic state prior to the TOR, but report similar TOTs with and without the anticipatory information.

Non-driving-related tasks (NDRTs): Several prior works [42, 50, 116, 132] have consistently reported worse take-over times or take-over quality when the driver is engaged in a NDRT prior to the take-over, whether the NDRT places visual, cognitive or motor-control based demand on the driver. In this chapter, we thus primarily focus on the effect of driver behavior and NDRTs on TOT. In particular, we map the observed NDRTs to feature descriptors of driver gaze, hand and foot activity using vision based models for driver behavior analysis and predict TOT based on these feature descriptors.

7.2.3 Take-over time prediction for autonomous driving

While the studies described in the previous section analyze take-over times under various experimental conditions, closest to our work are recently proposed machine learning models [12, 20, 48, 71, 100, 126] that *predict* TOT prior to the control transition.

Braunagel *et al.* [20] and Du *et al.* [48] propose binary classifiers that output whether or not the driver is ready to take-over. Gaze activity, NDRT label and a label for situation complexity are used as input features in [20], while gaze activity, heart rate variability, galvanic skin response, traffic density and TOT budget are used as inputs

in [48]. Pakdamanian *et al.* [126] propose a three class classifier over TOT intervals based on driver gaze activity, heart rate variability, galvanic skin response, NDRT label and vehicle signals. Lotz and Weissenberger [100] compare various classifiers over 4 TOT intervals trained using features capturing driver's head orientation and gaze activity, along with TTC. Hwang *et al.* [71] propose a regression model based on hidden Markov models that outputs TOT based on vehicle signals prior to the TOR. Finally, Berghofer *et al.* [12] propose a regression model for TOT prediction based on driver gaze activity and driver characteristics such as age, gender, sleepiness, attitude towards highly automated driving and previous experiences with automated driving.

Our work differs from previously proposed TOT prediction models on two counts. First, we use fine-grained descriptors of driver gaze, hand and foot activity obtained purely using non-intrusive vision sensors as inputs to our TOT prediction model. Second, we train and evaluate our models using a large *real-world* dataset of take-overs captured in a conditionally autonomous vehicle. Prior work on TOT prediction has been limited to the simulator setting [20, 48, 71, 100, 126]. Berghofer *et al.* [12] do use a real world dataset. However, they use a 'Wizard of Oz' setting where a safety driver with access to vehicle controls plays the role of the autonomous vehicle.

7.3 Dataset & Labels

7.3.1 Controlled Data Study (CDS)

To capture a diverse set of real-world take-overs, we conduct a large-scale study under controlled conditions. More specifically, we enlist a representative population of 89 subjects to drive a Tesla Model S testbed mounted with three driver-facing cameras that capture the gaze, hand, and foot activity of the driver. In this controlled data study (CDS), we required each subject to drive the testbed for approximately an hour in a pre-determined section of the roadway, under controlled traffic conditions. During the

drive, each test subject is asked to undertake a variety of distracting secondary activities while the autopilot is engaged, following which an auditory take-over request (TOR) is issued at random intervals. This initiates the control transition during which the driver is instructed to take control of the vehicle and resume the drive. Each such transition corresponds to one take-over event, and our CDS produces 1,375 take-over events in total.

7.3.2 Annotation

Automated video segmentation: Each driving session is first segmented into 30 second windows surrounding known take-over events, consisting of 20 seconds prior to the take-over request (TOR) and 10 seconds after the take-over event.

Event annotations: For each 30 second clip corresponding to a take-over event, we manually annotate the three times after the take-over request corresponding to when the driver's eyes are on the road, hands are on the wheel, and foot is on the pedal. We also label the secondary activity being performed by the driver during each take-over event, assigning one of 8 possible activity labels: (1) No secondary activity, (2) talking to co-passenger, (3) eyes closed, (4) texting, (5) phone call, (6) using infotainment unit, (7) counting change, (8) reading a book or magazine.

TOT statistics by secondary activity: Figure 7.2 shows the average times corresponding to eyes on road, hands on wheel and foot on pedal for each of the 8 secondary activities. It also shows the overall take-over time, which is the maximum of the three markers for each event. We note that texting, phone-calls, counting change and reading correspond to longer average take-over times, as compared to talking to the co-passenger or using the infotainment unit, which can be reasonably expected. Counter to intuition, the 'eyes closed behind the wheel' activity has low take-over times. This is mainly because the drivers are merely 'acting' to be asleep, since actual sleep could not have been achieved given the duration and nature of each trial. We also note that the 'hands on wheel' event

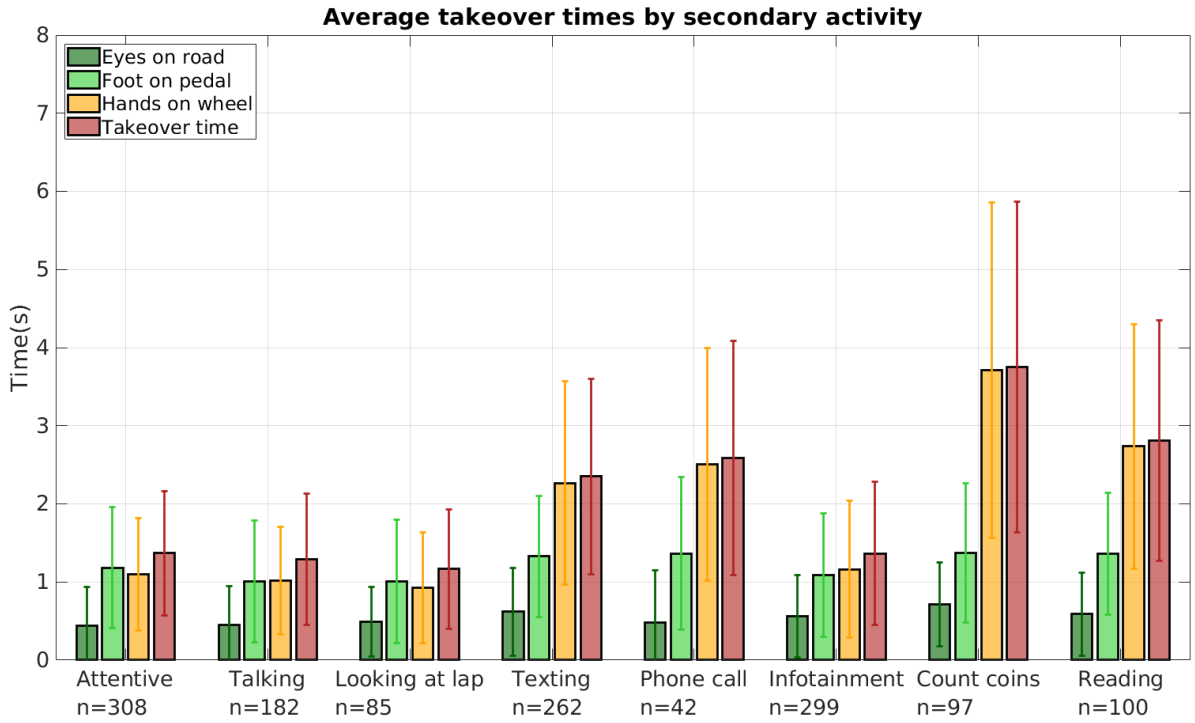


Figure 7.2. Take-over time statistics from the CDS: We plot the mean values (with error bars) of the different take-over related event timings for each secondary activity.

seems to take much longer on average, as compared to eyes on road or foot on pedal. This reinforces the need for driver hand analysis, which is also a key predictor of the driver's observable readiness index (see next section). Finally, we note that for the more distracting secondary activities (reading, texting, phone calls, counting change), even the foot on pedal times are longer compared to the other secondary activities, although the secondary activities do not involve the driver's feet. Thus, there seems to be a delay corresponding to the driver shifting attention from secondary activity to the primary activity of driving.

7.3.3 Data Augmentation

Takeover time data is very limited and expensive to capture and label. This is illustrated by the size of the CDS dataset (1,375 unique takeover events). We propose a new data augmentation scheme to increase the number of samples in the dataset by an order of magnitude in order to train our LSTM based TOT prediction models.

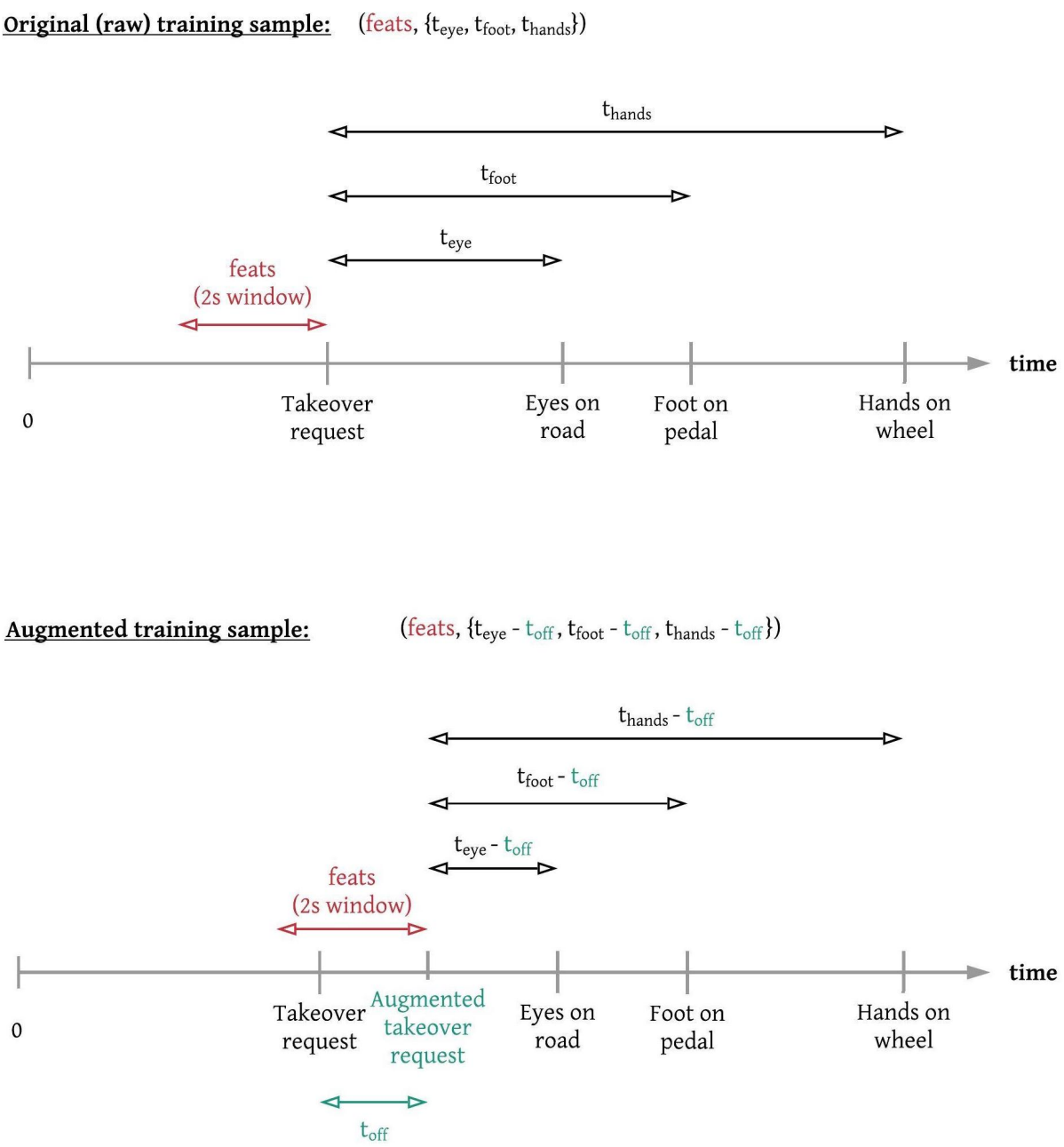


Figure 7.3. TOT dataset augmentation scheme: We increase the number of samples in our TOT prediction dataset by an order of magnitude by considering *augmented TORs* between the actual TOR and the first of the three takeover completion cues.

Figure 7.3 illustrates our data augmentation scheme. We term each take-over event in the CDS dataset a *raw sample*. Each raw sample has annotated timestamps corresponding to the take-over request (t_{tor}), as well as the time taken by the driver to get their eyes on the road (t_{eyes}), hands on the wheel (t_{hands}) and foot on the pedals (t_{foot}) after the TOR. We wish to learn a model that maps a 2 second window of driver activity prior to the TOR to the take-over times, $\{t_{eyes}, t_{hands}, t_{foot}\}$.

The raw samples alone are insufficient to train a machine learning model from scratch. We thus mine *augmented training samples* from each takeover event. An augmented training sample is characterized by an augmented TOR at time t_{off} after the actual t_{tor}. We use a 2 second window of driver activity before the augmented TOR as the input to the model while the corresponding takeover times are given by $\{t_{eyes} - t_{off}, t_{hands} - t_{off}, t_{foot} - t_{off}\}$. If the driver's hands, eyes, or foot are already in position at $t_{tor} + t_{off}$, the corresponding takeover time is set to 0.

An augmented training sample maps the driver's state at an intermediate timestamp during the takeover event, to their reaction times from that timestamp. While this doesn't correspond to an actual TOR, it still serves as useful data for training our TOT prediction model as we show in Section 7.5. Intuitively, the driver can be expected to be less and less distracted by a non-driving activity as t_{off} is increased, leading to shorter takeover times. Thus the augmented samples provide additional instances where the driver is increasingly prepared to takeover control from the vehicle.

We capture data at a frame rate of 30 Hz. Thus, t_{off} can be varied from 0 to the maximum of $\{t_{eyes}, t_{hands}, t_{foot}\}$ using increments of 1/30 seconds to yield multiple augmented samples per takeover event. With the proposed augmentation scheme, we get datasets vastly larger in size, as depicted in Table 7.1. The augmentation scheme is only applied to the training split. The validation and test splits are left untouched.

Table 7.1. Sizes of different takeover time prediction datasets.

Dataset	Number of samples
Raw dataset: CDS (R)	1,375
Augmented dataset: CDS (A)	47,461

7.4 Models for Predicting Takeover Times

It is important to preserve both the diverse and sequential nature of all features related to driver behavior while designing a holistic take-over time (TOT) prediction framework. High level tasks such as TOT prediction are influenced by low level driver behaviors. Figure 7.4 provides an overview of our proposed approach for estimating TOT. Our approach consists of two major components. The first component is a set of convolutional neural networks (CNNs) for extracting frame-wise descriptors of driver gaze, hand and foot activity from the raw camera feed. We describe these is greater detail in section 7.4.1. The second component is an LSTM model for estimating TOT based on a sequence of frame-wise features over a pre-defined time window. We describe the different variants of our LSTM based models in section 7.4.2.

7.4.1 Frame-wise feature extraction

Gaze activity: We use the model proposed by Vora *et al.* [177] for driver gaze analysis. The inputs to the model are frames from the face camera. We use a face detector [184] for localizing the driver's eyes. A cropped bounding box around the driver's eyes is passed through a CNN, which outputs the driver's gaze zone. We consider 8 gaze zones: {forward, left mirror, lap, speedometer, infotainment unit, rear-view mirror, right mirror, over the shoulder}. The CNN outputs frame-wise probabilities for each gaze zone. We use this 8 dimensional vector to represent driver's gaze features.

Hand activity: We use the model proposed by Yuen and Trivedi [185] for driver hand analysis. The model localizes the elbow and wrist joints of the driver using part affinity

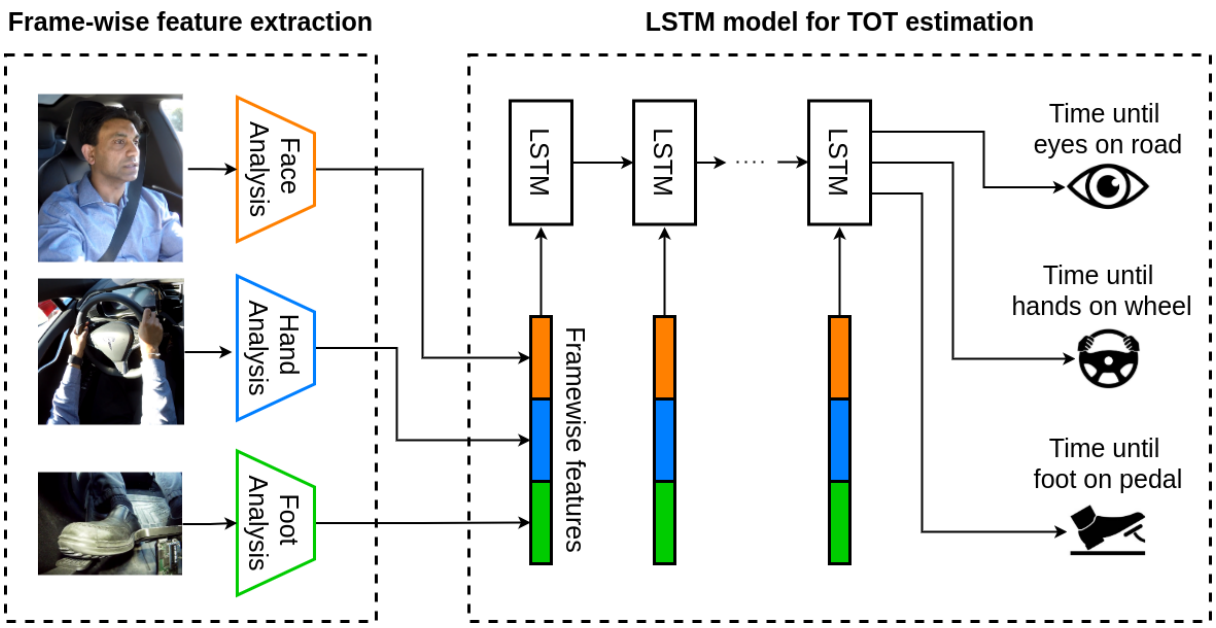


Figure 7.4. Overview of the proposed approach: We extract frame-wise descriptors of driver gaze, hand and foot activity. We propose an LSTM model for predicting TOT based on a sequence of the extracted features over a 2 second window.

fields [23]. A cropped bounding box around the driver's wrist is passed through a CNN to output probabilities corresponding to 6 hand activities for each hand: {on lap, in air, hovering over steering wheel, on steering wheel, on cupholder, interacting with infotainment unit}. We extend the model to additionally output hand-held object probabilities. We consider 7 object categories: {no-object, phone, tablet, food, beverage, book, other}. By running the models on images from a stereo camera pair, we also obtain 3-d coordinates for the driver's wrist locations and the steering wheel using triangulation. We then calculate the distance of each hand (wrist) of the driver to the steering wheel in 3-d. The hand activity probabilities, hand object probabilities and 3-d distance to steering wheel together form the hand activity features for each frame.

Foot activity: We use the model proposed by Rangesh and Trivedi [137] for driver foot analysis. Each frame from the foot camera feed is passed through a CNN to output probabilities over 5 foot activity classes: {away from pedal, on brake, on gas, hovering over brake, hovering over gas}. These probabilities represent the foot activity features.

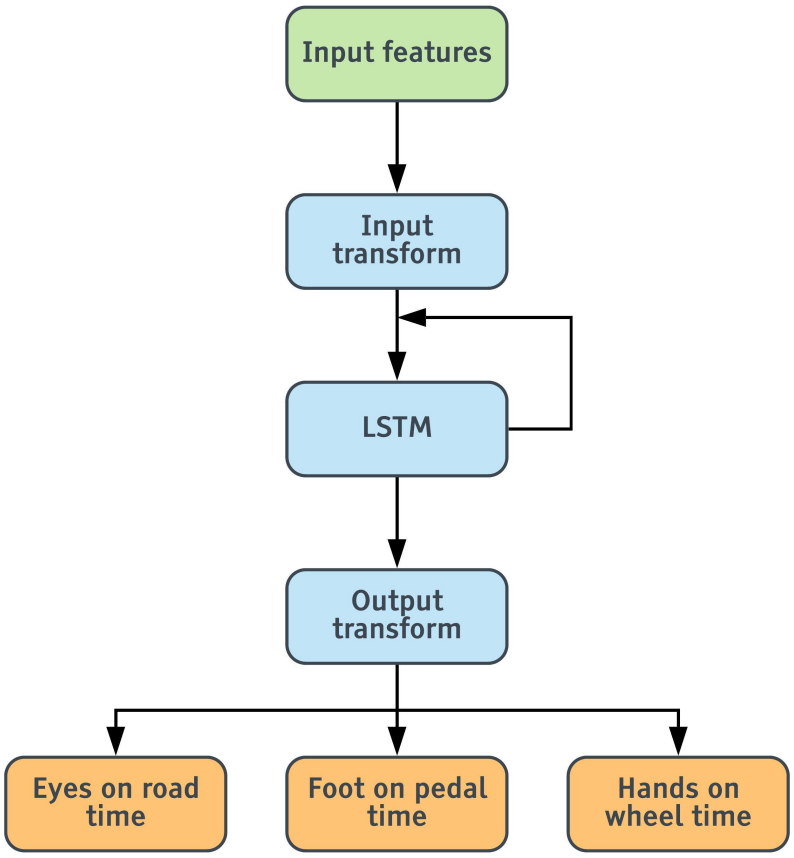


Figure 7.5. LSTMs: Baseline LSTM model architecture.

7.4.2 LSTM models for take-over time prediction

Baseline LSTM: This is the simplest (baseline) version of all TOT models. The input features are first transformed using a fully-connected (FC) layer of size 16 (plus non-linearity), which is then fed to an LSTM with a hidden state of size 32 at each timestep as shown in figure 7.5. The LSTM layer receives the transformed input features at each timestep and updates its internal representation known as the hidden state. In all our experiments, we choose a 2 second window of features as input to our models. After 2 seconds worth of inputs and updates, the hidden state of the LSTM after the latest timestep is passed through an output transformation (FC layer plus non-linearity) to

predict the three times of interest.

We apply a simple $L1$ loss to train this network. Let o_e, o_f, and o_h be the outputs produced by the model. Assuming t_e, t_f, and t_h are the target eyes on road time, foot on pedal time, and hands on wheel time respectively, the total loss is:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}|t_e^i - o_e^i| + \frac{1}{N}\sum_{i=1}^{N}|t_f^i - o_f^i| + \frac{1}{N}\sum_{i=1}^{N}|t_h^i - o_h^i|. \tag{7.2}$$

The entire model is trained using an Adam optimizer with a learning rate of 0.001 for 10 epochs.

Independent LSTMs: Figure 7.6 shows the independent LSTM model architecture. This model is the same as the baseline LSTM model, except for one major difference: each target output time has its own independent LSTM. The reasoning behind this is to accommodate different hidden state update rates for different driver behaviors, for example – eyes on road behavior is generally faster (short term) than hands on wheel behavior (mid/long term). Having multiple independent LSTMs allows each one to update at different rates, thereby capturing short/mid/long term behaviours separately.

Although each branch has its own LSTM cell, the input and output transformations are still shared between the three LSTMs as the feature inputs to the three branches are the same. This tends to reduce overfitting based on our experiments.

We use the identical loss (eq. 7.2) and optimizer settings as the baseline LSTM for training the independent LSTMs model.

LSTM with Multi-modal Outputs: This model shown in figure 7.7 is largely based on the baseline LSTM with one addition: *multi-modal outputs*. Instead of just producing one output for each of the three targets, we output $K(=3)$ outputs per target and their associated probabilities. We do this to model the inherent multi-modality and subjectiveness of takeover times. For example, given similar history of behavior, one driver may respond faster in taking control of the vehicle than another. Producing multiple

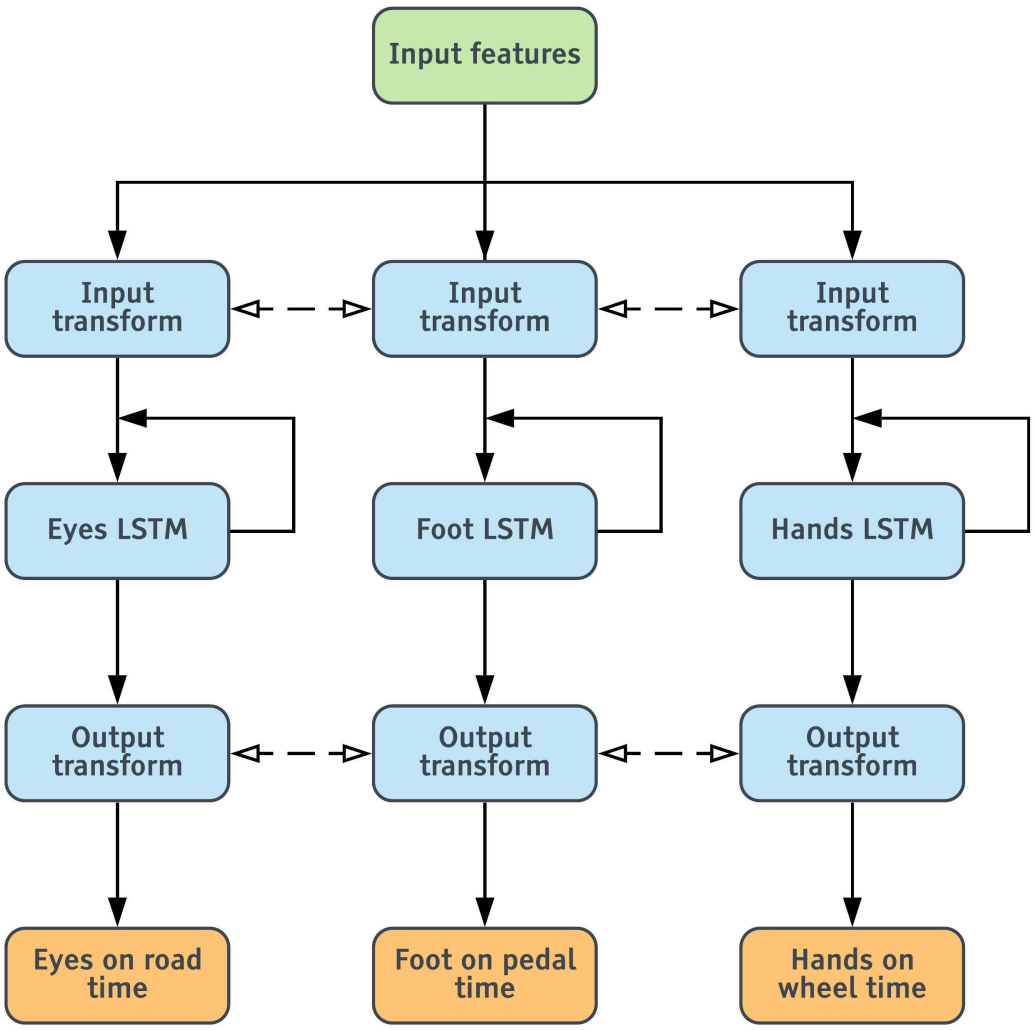


Figure 7.6. ID LSTMs: Independent LSTMs model architecture.

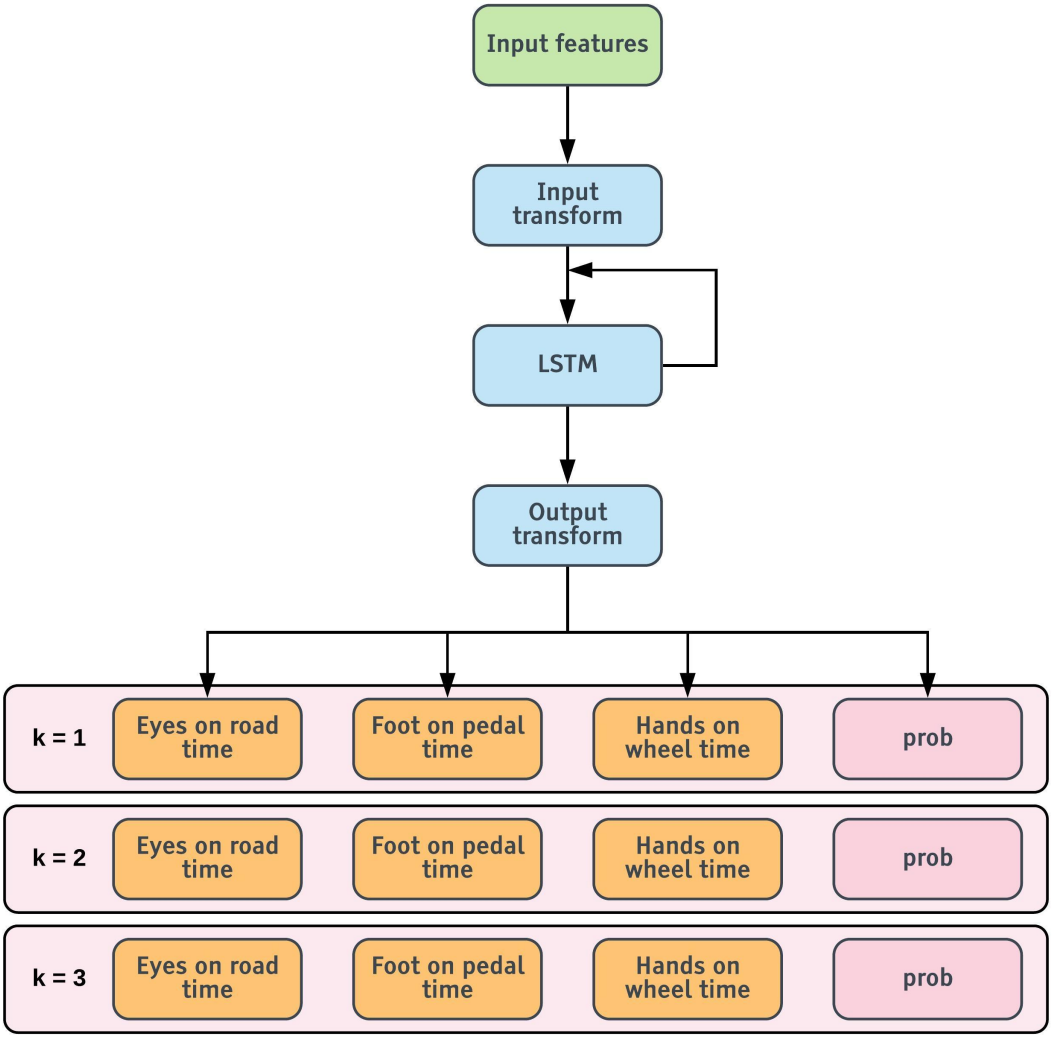


Figure 7.7. LSTMs + MM: LSTM with multi-modal outputs model architecture.

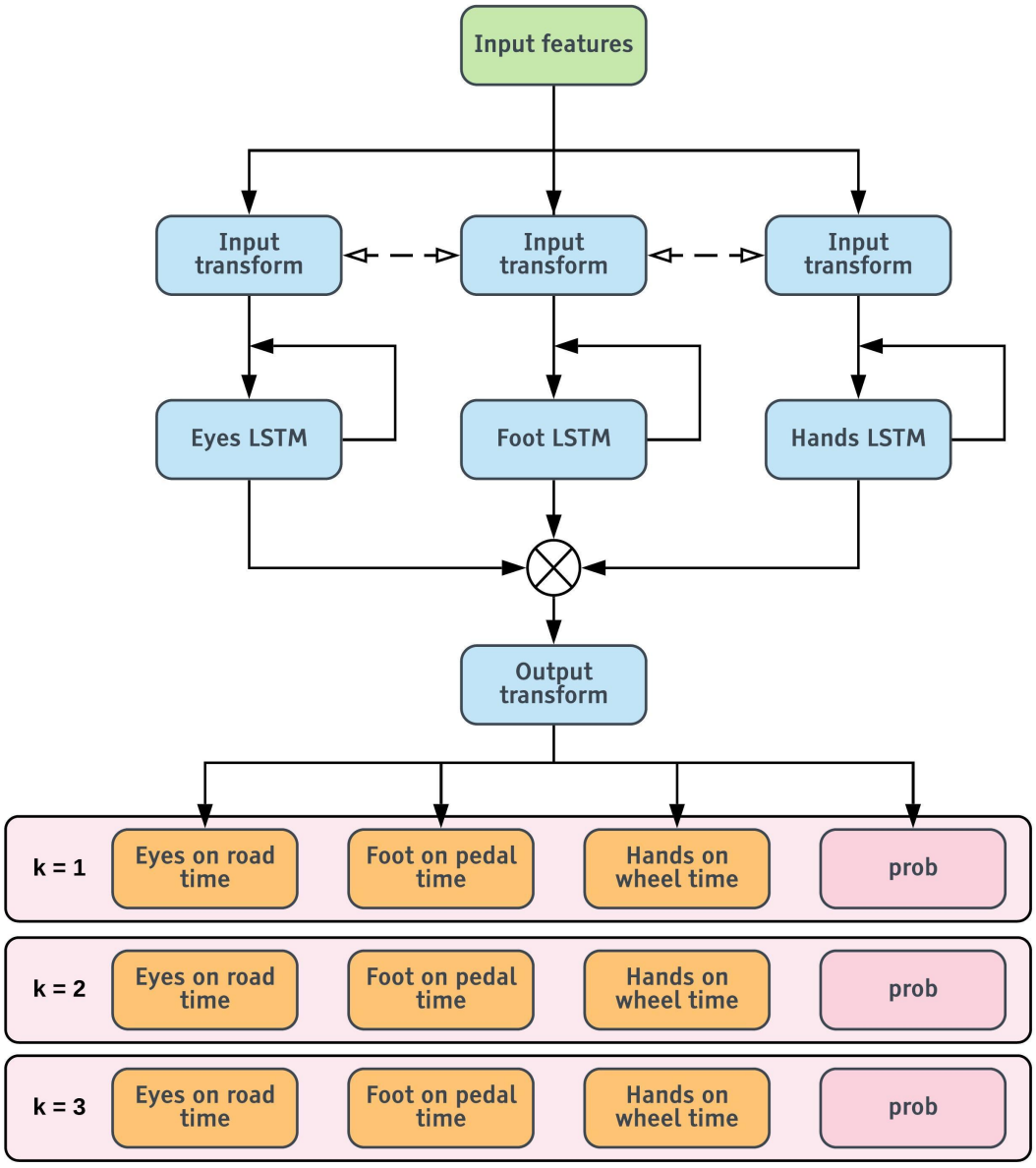


Figure 7.8. ID LSTMs + MM: Independent LSTMs with multi-modal outputs model architecture.

probable outputs (and their probabilities) could possibly address this ambiguity and provide more usable information to any downstream controller.

Unlike the previous models, this model is trained using a minimum of K loss, where $L1$ losses are only applied to the output modes closest to the ground truth target. Additionally, the output probabilities are refined using cross-entropy. Let $o_e(k)$, $o_f(k)$, $o_h(k)$ and $q(k)$ denote the k^{th} set of outputs and corresponding probability produced by the model. Assuming t_e, t_f, and t_h are the target eyes on road time, foot on pedal time, and hands on wheel time respectively, the total loss is:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \min_k \left(|t_e^i - o_e^i(k)| + |t_f^i - o_f^i(k)| + |t_h^i - o_h^i(k)|\right) - \lambda \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} p^i(k)\log(q^i(k)), \quad (7.3)$$

where $p^i(k)$ is a one-hot categorical probability distribution given by

$$p^i(k) = \mathbb{1}\left(\arg\min_l \left(|t_e^i - o_e^i(l)| + |t_f^i - o_f^i(l)| + |t_h^i - o_h^i(l)|\right) = k\right), \quad (7.4)$$

and λ is a coefficient used for relatively weighting the L1 and cross-entropy losses.

As before, the entire model is trained using an Adam optimizer with a learning rate of 0.001 for 10 epochs. We use $\lambda = 1$ for simplicity.

Independent LSTMs with Multi-modal Outputs: The final proposed model uses a combination of independent LSTMs and multi-modal outputs described before. One difference to the original independent LSTMs model is that we now concatenate the hidden states of all three LSTMs and transform them together to produce the target outputs. This is done because probabilities are assigned to the joint of all three target times, and thus need to be operated on together. We use the identical loss (equation 7.3) and optimizer settings as the LSTM with multi-modal outputs for training the independent LSTMs with

multi-modal outputs.

7.5 Experimental Evaluation

7.5.1 Comparison of LSTM models for TOT prediction

First, we conduct an experiment to assess the effects of different model architectures. All proposed models (from Section 7.4.2) were trained on CDS train set with augmented data, and then evaluated on the validation set. We use individual and overall mean absolute errors (MAEs) as metrics for comparison. Table 7.2 contains results from this experiment. In addition to the LSTM models, as a sanity check, we include a simple baseline that always predicts a constant value for all take-over time markers, corresponding to the maximum value for each marker from the train set. From these results, we note that all LSTM models considerably outperform the constant value baseline showing that there is a learnable signal in the data and the usefulness of using a machine learning model. We observe that the independent LSTMs model consistently outperforms other models. At first glance, the multi-modal models tend to perform worse than the ones without multi-modal outputs. To further analyze the source of these errors, we provide the *best-of-K* MAEs for these models in Table 7.2. The *best-of-K* MAEs simply mean that instead of choosing the most probable set of predictions for error calculation, we use the set that produces the least error i.e. assume perfect classification. The *best-of-K* numbers are vastly superior to the ones without multi-modal outputs. This indicates that in most cases, at least one of $K(= 3)$ sets of predictions is highly accurate. However, accurate probability assignment for these K modes (i.e. classification) remains error-prone. Nevertheless, we believe that having multiple probable outputs instead of one less accurate one could be beneficial for downstream controllers.

Table 7.2. TOT prediction errors for the CDS validation set comparing model architectures.

Model type (s)	Overall MAE (s)	Eyes on road MAE (s)	Foot on pedal MAE (s)	Hands on wheel MAE (s)	Takeover time MAE (s)
Constant prediction (Max over train set stats)	3.9271	2.4540	2.9880	6.3392	6.1969
LSTM[1]	0.5104	0.3353	0.5029	0.7126	0.8098
ID LSTMs[2]	**0.5073**	**0.3266**	**0.4841**	**0.7113**	**0.7912**
LSTM + MM[3]	0.5589	0.3582	0.5262	0.7921	0.8908
ID LSTMs + MM	0.5319	0.3415	0.5019	0.7524	0.8441
LSTM + MM (best of K)	0.3921	0.2393	0.4204	0.5167	0.6265
ID LSTMs + MM (best of K)	0.3911	0.2344	0.3875	0.5513	0.6586

[1] baseline LSTM model [2] Independent LSTMs [3] Multi-modal outputs (with $K = 3$ modes)

Table 7.3. Effect of data augmentation and ORI pretraining

Training dataset (s)	Overall MAE (s)	Eyes on road MAE (s)	Foot on pedal MAE (s)	Hands on wheel MAE (s)	Takeover time MAE (s)
CDS (R)[1]	0.5799	0.3676	0.5435	0.8285	0.8576
CDS (A)[2]	**0.5073**	0.3266	0.4841	0.7113	0.7912
ORI[3]→ CDS (A) (Fig. 7.9)	0.5184	**0.3246**	0.5182	**0.7054**	**0.7729**

[1] raw dataset [2] augmented dataset [3] ORI estimation dataset

7.5.2 Effect of data augmentation and transfer learning

Next, we conduct experiments to assess the effects of our data augmentation and transfer learning schemes. To isolate these effects, we use the same ID LSTMs model for all experiments. We compare the following training schemes:

- **CDS (R):** First, as a baseline, we train a model purely using the raw CDS data without augmentation.
- **CDS (A):** Next, we train a model using the augmented training dataset using the augmentation scheme described in section 7.3.3. The number of training samples for the raw and augmented datasets are shown in table 7.1
- **ORI $\rightarrow$ CDS (A):** Finally, we consider a model pre-trained to estimate the observable take-over readiness index (ORI) proposed in [39]. The ground truth ORI values are obtained via subjective ratings assigned by multiple human observers rating how ready a driver is to take-over control from the vehicle based on the past two seconds of video feed from the driver facing cameras. The ratings are normalized and averaged to account for rater bias as described in [39]. The process for transfer learning from ORI estimation to TOT prediction is shown in Fig 7.9.

Results from these experiments are presented in Table 7.3. As before, we use individual and overall mean absolute errors (MAEs) as metrics for comparison. From Table 7.3, we notice that training on the augmented dataset (as proposed in Section 7.3.3) consistently and considerably improves performance as compared to the raw dataset. We believe that doing so prevents overfitting, provides regularization, smooths the outputs of model, and adds new training samples that would be cumbersome or impossible to capture. Finally, we observe that training the model for observable readiness index (ORI) estimation [39], followed by transfer learning on TOT prediction improves some metrics.

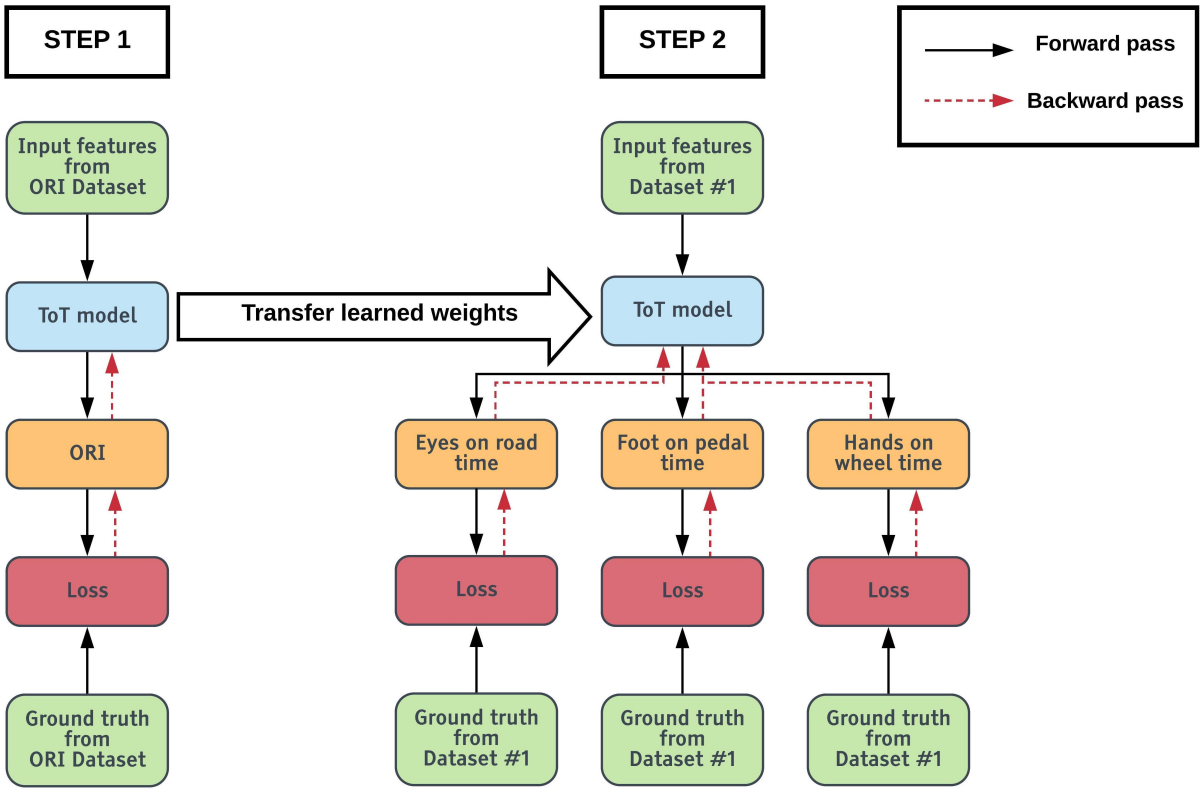


Figure 7.9. ORI pretraining: We use a transfer learning approach to first train a model for ORI estimation [39] (Step 1), and then refine the model's weights on the target TOT prediction task (Step 2).

This highlights the commonality between the two tasks - features from learning one task can improve performance in the other.

7.5.3 Effect of hand, gaze and foot activity features

Finally, we conduct an experiment to assess the relative importance of different input features and their combinations. To isolate effects from features, we train the same ID LSTMs model with different input feature combinations. We use individual and overall mean absolute errors (MAEs) as metrics for comparison. Table 7.4 contains results from this experiment. We notice that hand features are the most important, followed by foot and gaze features respectively. This might be because gaze dynamics are relatively predictable during takeovers as the first thing drivers tend to do is look at the road to assess the

Table 7.4. TOT prediction errors for different times of interest on the CDS validation set for a variety of feature combinations.

Features					Overall	Eyes on road	Foot on pedal	Hands on wheel	Takeover time
F[1]	G[2]	H[3]	S[4]	O[5]	MAE (s)	MAE (s)	MAE (s)	MAE (s)	MAE (s)
✓					0.5735	0.3587	0.5018	0.8599	0.8856
	✓				0.5811	0.3332	0.5690	0.8411	0.8837
		✓			0.5560	0.3729	0.5384	0.7565	0.9012
		✓	✓		0.5420	0.3783	0.5109	0.7369	0.8315
		✓		✓	0.5217	0.3702	0.4973	0.7177	0.8621
		✓	✓	✓	0.5182	0.3747	0.4857	0.7141	0.7983
	✓	✓		✓	0.5202	0.3244	0.5220	0.7163	0.7920
	✓	✓	✓	✓	0.5213	0.3299	0.5124	0.7215	0.7921
✓	✓	✓	✓		0.5384	**0.3222**	0.5059	0.7870	0.8475
✓	✓	✓		✓	0.5088	0.3277	0.5074	0.7144	0.7918
✓	✓	✓	✓	✓	**0.5073**	0.3266	**0.4841**	**0.7113**	**0.7912**

[1] **foot features:** probabilities for all 5 foot activities, namely - away from pedal, on break, on gas, hovering over break, and hovering over gas
[2] **gaze features:** probabilities for all 8 gaze zones, namely - front, speedometer, rearview, left mirror, right mirror, over the shoulder, infotainment, and eyes closed/looking down
[3] **hand features:** probabilities for all 6 hand activities (left and right hand), namely - on lap, in air, hovering over steering wheel, on steering wheel, cupholder, and infotainment
[4] **stereo hand features:** distance of left and right hand from the steering wheel
[5] **hand-object features:** probabilities for all 7 hand object categories (left and right hand), namely - no object, cellphone, tablet/iPad, food, beverage, reading, and others

situation, leading to less variance in eyes-on-road behavior. Next, we notice that adding more informative hand feature like 3D distances to the steering wheel and hand-object information improves the performance further. Hand-objects in particular seem to vastly improve the performance in general. This makes sense as hand-objects are the strongest cue related the secondary activities of drivers. Adding stereo hand features improves the results, but not by much. Adding foot features also tends to reduce the errors considerably, illustrating the importance of having a foot camera. In conclusion, one could get close to peak performance by utilizing 3 cameras - 1 foot, 1 hand, and 1 face camera respectively. Hand features are most informative, followed by foot and gaze features respectively.

Table 7.5. Prediction errors for different models on the takeover time test set.

Model type (s)	Overall MAE (s)	Eyes on road MAE (s)	Foot on pedal MAE (s)	Hands on wheel MAE (s)	Takeover time MAE (s)
Constant prediction (Max over train set stats)	4.0835	2.6790	3.1540	6.4175	6.2073
LSTM[1]	0.5242	**0.2365**	0.5007	0.8710	0.9457
ID LSTMs[2]	**0.5208**	0.2497	**0.4650**	**0.8055**	**0.9144**
LSTM + MM[3]	0.5339	0.2635	0.5265	0.8117	0.9307
ID LSTMs + MM	0.5526	0.2665	0.5180	0.8734	0.9418
ID LSTMs (75%[4])	0.5348	0.2557	0.5013	0.8474	0.9779
ID LSTMs (90%[5])	0.5282	0.2514	0.4851	0.8482	0.9424

[1] baseline LSTM model [2] Independent LSTMs [3] Multi-modal outputs (with $K = 3$ modes) [4] 75% of the dataset used for training [5] 90% of the dataset used for training

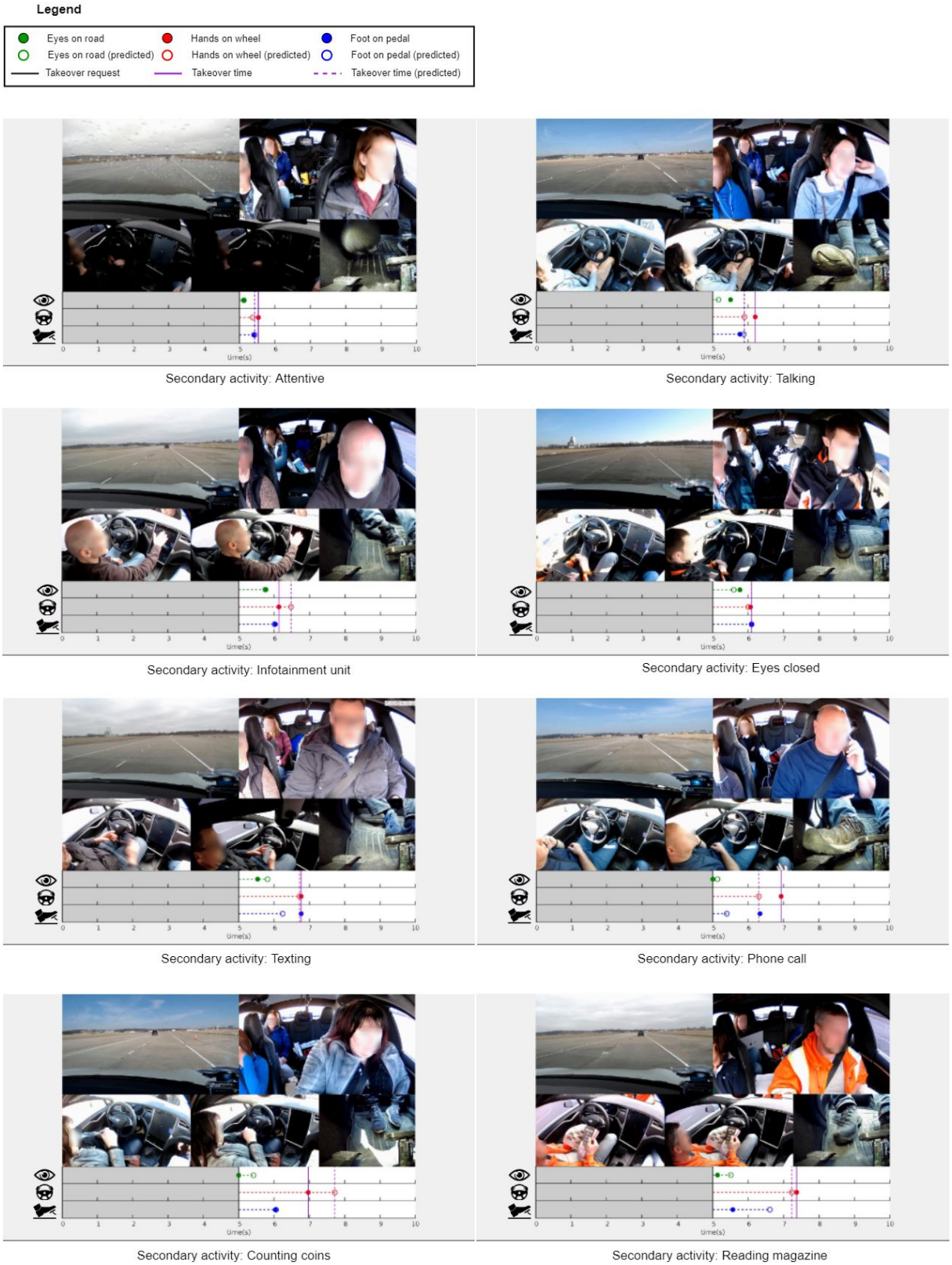


Figure 7.10. Qualitative examples: Predicted TOT for each of the 8 secondary activities in the CDS dataset.

7.5.4 Quantitative results on test set

In this section, we present quantitative error metrics on the held out test set, separate from the validation set, for all proposed models in Table 7.5. As before, we see that ID LSTMs is the best performing model. We also notice that hands-on-wheel MAEs are usually the largest due to large variance in hand behaviors, and large absolute values associated with hands-on-wheel time. We also show results for ID LSTMs when trained on 75% and 90% of available training data. This helps us gauge the expected improvement in performance as more training data is added. Based on Table 7.5, we can expect meager improvements as more data is added. This indicates diminishing returns.

7.5.5 Qualitative examples

We also provide qualitative examples of predictions made by the ID-LSTMs model, with the drivers performing each of the 8 secondary activities (Figure 7.10). Each example shows the 5 camera views at the instant where the TOR is issued. The true values of the 3 takeover times (eyes on road, hands on wheel, foot on pedal) are shown in the plot as solid circular markers, while the corresponding predicted values are shown as hollow circular markers of the same color. Finally, we show the ground truth takeover time as a solid purple line and the predicted takeover time as a dashed purple line. We note that the model accurately predicts short takeover times when the driver is attentive, talking or operating the infotainment unit, and longer takeover times when the driver is texting, making a phone call, counting coins or reading.

7.6 Conclusions

This chapter presented one of the largest real-world studies on takeover time prediction and control transitions in general. We introduced a dataset of take-over events captured via controlled driving studies in a commercially available partially autonomous

vehicle, with a large pool of test subjects performing a variety of secondary activities prior to the control transition. We proposed a machine learning model for take-over time prediction based on driver gaze, hand and foot activity prior to the issue of take-over requests. We also proposed a data augmentation and transfer learning scheme for best utilizing the limited number of take-over events in our dataset. Our experiments show that our model can reliably predict takeover times for various secondary activities being performed by the drivers. In particular, we showed the usefulness of analyzing driver hand, foot and gaze activity prior to issuing the take-over request. We also showed the utility of our transfer learning and data augmentation schemes for best utilizing limited training data with control transitions. We believe that this study outlines the sensors, datasets, methods and models that can benefit the intermediate stages of automation by accurately assessing driver behavior, and predicting takeover times - both of which can be used to smoothly transfer control between human and automation.

Acknowledgements

Chapter 7, in part, is based on "Take-over Time Prediction for Autonomous Driving in the Real-World: Robust Models, Data Augmentation, and Evaluation," Akshay Rangesh, Nachiket Deo, Ross Greer, Pujitha Gunaratne, Mohan M. Trivedi, currently submitted to IEEE Transactions on Human Machine Systems. The dissertation author was one of the primary investigators and authors of this paper.

Chapter 8

Conclusions

The goal of this dissertation was to equip autonomous vehicles with foresight – to allow them to interact and cooperate with humans in and around them as well as human driven vehicles that share the roads with them. We developed models for predicting the behavior of surrounding pedestrians and vehicles to aid safe path planning. We also developed models for predicting driver behavior during takeovers to ensure safe and smooth control transitions. Concretely, we made the following contributions.

In chapters 2 and 3, we addressed vehicle motion prediction on multi-lane highways. In chapter 2, we proposed a unified framework for maneuver recognition and trajectory prediction. We showed that vehicle motion prediction on highways can be simplified by breaking it down into two tasks, predicting discrete maneuvers and predicting continuous trajectories conditioned on these maneuvers. We showed that maneuver conditioned trajectory prediction better models rare but safety critical behaviors such as overtakes and cut-ins. We also showed that jointly modeling the behavior of all vehicles in the scene leads to better prediction that modeling each agent's behavior independently. In chapter 3, we improved upon the framework proposed in chapter 2, by proposing differentiable and end-to-end trainable modules for encoding agent motion, modeling agent-agent interaction, and predicting vehicle maneuvers and maneuver-conditioned trajectories. Our key contributions were *convolutional social pooling* to robustly model agent-agent

interaction and maneuver based LSTM decoders for predicting multimodal trajectory distributions over maneuver classes.

In chapters 4 and 5, we addressed the more complex task of predicting agent trajectories in urban environments, where static scene elements play a major role in regulating agent motion. In chapter 4, we leveraged a grid-based discrete policy to infer goal and path preferences of agents in unknown scenes and proposed a trajectory generator the outputs continuous trajectories conditioned on roll-outs of the policy. We showed that conditioning predictions on our grid based policy led to a diverse set of scene-compliant trajectories over long prediction horizons. In chapter 5, we improved upon our grid-based policy by encoding the scene as a graph, defined over the lane network. We proposed a model consisting of a discrete policy that explores the lane graph and a trajectory decoder that predicts future trajectories conditioned on graph traversals sampled from the policy. We showed that this led to more accurate predictions with a much lighter and faster model than the one in chapter 4.

Finally, **in chapters 6 and 7**, we proposed models for predicting driver behavior during control transitions. In chapter 6, we proposed a metric for driver takeover readiness based purely on observable cues, using subjective ratings assigned by multiple human observers. We also proposed an LSTM model for predicting the driver's takeover readiness based on gaze, hand and foot activity. Our analysis showed a high degree of consistency in the ratings assigned by multiple human raters. Our analysis also showed the utility of driver gaze, hand, pose and foot activity for predicting the driver's takeover readiness. In chapter 7, we went a step further and proposed a model to predict the reaction times of the driver during takeovers based on their gaze, hand and foot activity using a real world dataset of control transitions in an autonomous vehicle. We proposed a data augmentation scheme and a transfer learning approach to address the limited training data with real takeovers and showed the utility of both through our experiments.

Bibliography

[1] Automated driving at a new level: the audi ai traffic jam pilot. https://www.audi-mediacenter.com/en/press-releases/automated-driving-at-a-new-level-the-audi-ai-traffic-jam-pilot-9300. Accessed: 2018-10-31.

[2] Introducing navigate on autopilot. https://www.tesla.com/blog/introducing-navigate-autopilot. Accessed: 2017-10-31.

[3] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[4] Samer Ammoun and Fawzi Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *International Conference on Intelligent Computer Communication and Processing*, pages 417–422. IEEE, 2009.

[5] Georges S Aoude, Brandon D Luders, Kenneth KH Lee, Daniel S Levine, and Jonathan P How. Threat assessment design for driver assistance system at intersections. In *13th international ieee conference on intelligent transportation systems*, pages 1855–1862. IEEE, 2010.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[7] Mohammad Bahram, Constantin Hubmann, Andreas Lawitzky, Michael Aeberhard, and Dirk Wollherr. A combined model-and learning-based framework for interaction-aware maneuver prediction. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1538–1550, 2016.

[8] Alexander Barth and Uwe Franke. Where will the oncoming vehicle be the next second? In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1068–1073, 2008.

[9] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[10] Ardhendu Behera, Alexander Keidel, and Bappaditya Debnath. Context-driven multi-stream lstm (m-lstm) for recognizing fine-grained activity of drivers. In *Pattern Recognition*, pages 298–314, 2019.

[11] Luis Miguel Bergasa, Jesús Nuevo, Miguel A Sotelo, Rafael Barea, and María Elena Lopez. Real-time system for monitoring driver vigilance. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):63–77, 2006.

[12] Frauke Berghöfer, Christian Purucker, Frederik Naujoks, Katharina Wiedemann, and Claus Marberger. Prediction of take-over time demand in highly automated driving. results of a naturalistic driving study prediction of take-over time demand in conditionally automated driving-results of a real world driving study. *Proceedings of the Human Factors and Ergonomics Society Europe*, 2019.

[13] Holger Berndt, Jorg Emmert, and Klaus Dietmayer. Continuous driver intention recognition with hidden markov models. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1189–1194, 2008.

[14] Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional flow variational autoencoders for structured sequence prediction. *arXiv preprint arXiv:1908.09008*, 2019.

[15] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8485–8493, 2018.

[16] Apratim Bhattacharyya, Christoph-Nikolas Straehle, Mario Fritz, and Bernt Schiele. Haar wavelet based block autoregressive flows for trajectories. In *DAGM German Conference on Pattern Recognition*, pages 275–288. Springer, 2020.

[17] Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer.

[18] Guido Borghi, Elia Frigieri, Roberto Vezzani, and Rita Cucchiara. Hands on the wheel: a dataset for driver hand detection and tracking. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pages 564–570. IEEE, 2018.

[19] Christian Braunagel, Enkelejda Kasneci, Wolfgang Stolzmann, and Wolfgang Rosenstiel. Driver-activity recognition in the context of conditionally autonomous driving. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 1652–1657. IEEE, 2015.

[20] Christian Braunagel, Wolfgang Rosenstiel, and Enkelejda Kasneci. Ready for take-over? a new driver assistance system for an automated classification of driver take-over readiness. *IEEE Intelligent Transportation Systems Magazine*, 9(4):10–22, 2017.

[21] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[22] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[23] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[24] Sandra Carrasco, D Fernández Llorca, and MA Sotelo. Scout: Socially-consistent and understandable graph attention network for trajectory prediction of vehicles and vrus. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1501–1508, 2021.

[25] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning (CoRL)*, pages 947–956, 2018.

[26] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning (CoRL)*, pages 86–99. PMLR, 2020.

[27] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[29] Hallie Clark and Jing Feng. Age differences in the takeover of vehicle control and engagement in non-driving-related activities in simulated driving with conditional automation. *Accident Analysis & Prevention*, 106:468–479, 2017.

[30] James Colyar and John Halkias. Us highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[31] James Colyar and John Halkias. Us highway i-80 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[32] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2090–2096, 2019.

[33] Nikhil Das, Eshed Ohn-Bar, and Mohan M Trivedi. On performance evaluation of driver hand detection algorithms: Challenges, dataset, and metrics. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 2953–2958. IEEE, 2015.

[34] Nachiket Deo, Akshay Rangesh, and Mohan Trivedi. In-vehicle hand gesture recognition using hidden markov models. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 2179–2184. IEEE, 2016.

[35] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140, 2018.

[36] Nachiket Deo and Mohan M Trivedi. Learning and predicting on-road pedestrian behavior around vehicles. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[37] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018.

[38] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184, 2018.

[39] Nachiket Deo and Mohan M Trivedi. Looking at the driver/rider in autonomous vehicles to predict take-over readiness. *IEEE Transactions on Intelligent Vehicles*, 5(1):41–52, 2019.

[40] Nachiket Deo and Mohan M Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020.

[41] David F Dinges and Richard Grace. Perclos: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance. *US Department of Transportation, Federal Highway Administration, Publication Number FHWA-MCRT-98-006*, 1998.

[42] Ebru Dogan, Mohamed-Cherif Rahal, Renaud Deborne, Patricia Delhomme, Andras Kemeny, and Jérôme Perrin. Transition of control in a partially automated vehicle: Effects of anticipation and non-driving-related task involvement. *Transportation research part F: traffic psychology and behaviour*, 46:205–215, 2017.

[43] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[44] Anup Doshi and Mohan Trivedi. A comparative exploration of eye gaze and head motion cues for lane change intent prediction. In *2008 IEEE Intelligent Vehicles Symposium*, pages 49–54. IEEE, 2008.

[45] Anup Doshi and Mohan M Trivedi. Tactical driver behavior prediction and intent inference: A review. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1892–1897, 2011.

[46] Anup Doshi and Mohan M Trivedi. Head and eye gaze dynamics during visual attention shifts in complex environments. *Journal of vision*, 12(2):9–9, 2012.

[47] Katherine Driggs-Campbell, Victor Shia, and Ruzena Bajcsy. Improved driver modeling for human-in-the-loop vehicular control. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1654–1661. IEEE, 2015.

[48] Na Du, Feng Zhou, Elizabeth M Pulver, Dawn M Tilbury, Lionel P Robert, Anuj K Pradhan, and X Jessie Yang. Predicting driver takeover performance in conditionally automated driving. *Accident Analysis & Prevention*, 148:105748, 2020.

[49] Jacob Velling Dueholm, Miklas Strøm Kristoffersen, Ravi Kumar Satzoda, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi. Trajectories and maneuvers of surrounding vehicles with panoramic camera arrays. *IEEE Transactions on Intelligent Vehicles*, 1(2):203–214, 2016.

[50] Alexander Eriksson and Neville A Stanton. Takeover time in highly automated vehicles: noncritical transitions to and from manual control. *Human factors*, 59(4):689–705, 2017.

[51] Lex Fridman, Philipp Langhans, Joonbum Lee, and Bryan Reimer. Driver gaze region estimation without use of eye movement. *IEEE Intelligent Systems*, 31(3):49–56, 2016.

[52] Lex Fridman, Joonbum Lee, Bryan Reimer, and Trent Victor. 'owl'and 'lizard': patterns of head pose and eye pose in driver gaze classification. *IET Computer Vision*, 10(4):308–314, 2016.

[53] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[54] Christian Gold, Daniel Damböck, Lutz Lorenz, and Klaus Bengler. "take over!" how long does it take to get the driver back into the loop? In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 1938–1942. SAGE Publications Sage CA: Los Angeles, CA, 2013.

[55] Christian Gold, Moritz Körber, David Lechner, and Klaus Bengler. Taking over control from highly automated vehicles in complex traffic situations: the role of traffic density. *Human factors*, 58(4):642–652, 2016.

[56] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[57] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

[58] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[59] Ross Greer, Nachiket Deo, and Mohan Trivedi. Trajectory prediction in autonomous driving with a lane heading auxiliary loss. *Robotics and Automation Letters (RA-L)*, 2021.

[60] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 2255–2264, 2018.

[61] Michael R Hafner, Drew Cunningham, Lorenzo Caminiti, and Domitilla Del Vecchio. Cooperative collision avoidance at intersections: Algorithms and experiments. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1162–1175, 2013.

[62] John HL Hansen, Kazuya Takeda, Sanjeev M Naik, Mohan M Trivedi, Gerhard U Schmidt, Yingying Jennifer Chen, and Wade Trappe. Signal processing for smart vehicle technologies: Part 2 [from the guest editors]. *IEEE SIgnal ProcESSIng MagazInE*, 34(2):18–21, 2017.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[64] Christoph Hermes, Christian Wohler, Konrad Schenk, and Franz Kummert. Long-term vehicle motion prediction. In *IEEE intelligent vehicles symposium (IV)*, pages 652–657, 2009.

[65] Jrg Hillenbrand, Andreas M Spieker, and Kristian Kroschel. A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs. *IEEE Transactions on intelligent transportation systems*, 7(4):528–540, 2006.

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[67] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019.

[68] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle trajectory prediction based on motion model and maneuver recognition. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4363–4369, 2013.

[69] Dongliang Huang and Henry Leung. Em-imm based land-vehicle navigation with gps/ins. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 624–629, 2004.

[70] Gaojian Huang, Clayton Steele, Xinrui Zhang, and Brandon J Pitts. Multimodal cue combinations: a possible approach to designing in-vehicle takeover requests for semi-autonomous driving. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 63, pages 1739–1743. SAGE Publications Sage CA: Los Angeles, CA, 2019.

[71] Steven Hwang, Ashis G Banerjee, and Linda Ng Boyle. Predicting driver's transition time to a secondary task given an in-vehicle alert. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[72] Ashesh Jain, Hema S Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3182–3190, 2015.

[73] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3118–3125. IEEE, 2016.

[74] S. Jha and C. Busso. Probabilistic estimation of the gaze region of the driver using dense classification. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 697–702, Nov 2018.

[75] Joshua Joseph, Finale Doshi-Velez, Albert S Huang, and Nicholas Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383 – 400, 2011.

[76] Tzyy-Ping Jung, Scott Makeig, Magnus Stensmo, and Terrence J Sejnowski. Estimating alertness from the eeg power spectrum. *IEEE transactions on biomedical engineering*, 44(1):60–69, 1997.

[77] Nico Kaempchen, Kristian Weiss, Michael Schaefer, and Klaus CJ Dietmayer. Imm object tracking for high dynamic driving maneuvers. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 825–830, 2004.

[78] Eugen Käfer, Christoph Hermes, Christian Wöhler, Helge Ritter, and Franz Kummert. Recognition of situation classes at road intersections. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3960–3965, 2010.

[79] Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*, 2020.

[80] Aida Khosroshahi, Eshed Ohn-Bar, and Mohan Manubhai Trivedi. Surround vehicles trajectory analysis with recurrent neural networks. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2267–2272, 2016.

[81] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *IEEEInternational Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404, 2017.

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[83] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[84] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision (ECCV)*, pages 201–214. Springer, 2012.

[85] Terry K Koo and Mae Y Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, 15(2):155–163, 2016.

[86] Moritz Körber, Christian Gold, David Lechner, and Klaus Bengler. The influence of age on the take-over of vehicle control in highly automated driving. *Transportation research part F: traffic psychology and behaviour*, 39:19–32, 2016.

[87] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211, 2017.

[88] Christian Laugier, Igor E Paromtchik, Mathias Perrollaz, Mao Yong, John-David Yoder, Christopher Tay, Kamel Mekhnacha, and Amaury Nègre. Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *IEEE Intelligent Transportation Systems Magazine*, 3(4):4–19, 2011.

[89] Andreas Lawitzky, Daniel Althoff, Christoph F Passenberg, Georg Tanzmeister, Dirk Wollherr, and Martin Buss. Interactive scene prediction for automotive applications. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1028–1033, 2013.

[90] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 336–345, 2017.

[91] Sung Joo Lee, Jaeik Jo, Ho Gi Jung, Kang Ryoung Park, and Jaihie Kim. Real-time gaze estimator based on driver's head orientation for forward collision warning system. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):254–267, 2011.

[92] Marco Leo, G Medioni, M Trivedi, Takeo Kanade, and Giovanni Maria Farinella. Computer vision for assistive technologies. *Computer Vision and Image Understanding*, 154:1–15, 2017.

[93] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[94] Nanxiang Li and Carlos Busso. Predicting perceived visual and cognitive distractions of drivers with multimodal features. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):51–65, 2015.

[95] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision (ECCV)*, 2020.

[96] Yulan Liang and John D Lee. A hybrid bayesian network approach to detect driver cognitive distraction. *Transportation research part C: emerging technologies*, 38:146–155, 2014.

[97] Yulan Liang, Michelle L Reyes, and John D Lee. Real-time detection of driver cognitive distraction using support vector machines. *IEEE transactions on intelligent transportation systems*, 8(2):340–350, 2007.

[98] Chin-Teng Lin, Chun-Hsiang Chuang, Chih-Sheng Huang, Shu-Fang Tsai, Shao-Wei Lu, Yen-Hsuan Chen, and Li-Wei Ko. Wireless and wearable eeg system for evaluating driver vigilance. *IEEE Transactions on biomedical circuits and systems*, 8(2):165–176, 2014.

[99] Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. Driver distraction detection using semi-supervised machine learning. *IEEE transactions on intelligent transportation systems*, 17(4):1108–1120, 2016.

[100] Alexander Lotz and Sarah Weissenberger. Predicting take-over times of truck drivers in conditional autonomous driving. In *International Conference on Applied Human Factors and Ergonomics*, pages 329–338. Springer, 2018.

[101] Chenxu Luo, Lin Sun, Dariush Dabiri, and Alan Yuille. Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles. *arXiv preprint arXiv:2007.02574*, 2020.

[102] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, 2015.

[103] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[104] Hiren M Mandalia and Mandalia Dario D Salvucci. Using support vector machines for lane-change detection. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 49, pages 1965–1969, 2005.

[105] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15233–15242, 2021.

[106] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision (ECCV)*, pages 759–776. Springer, 2020.

[107] Sujitha Martin, Sourabh Vora, Kevan Yuen, and Mohan Manubhai Trivedi. Dynamics of driver's gaze: Explorations in behavior modeling and maneuver prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):141–150, 2018.

[108] Kenneth O McGraw and Seok P Wong. Forming inferences about some intraclass correlation coefficients. *Psychological methods*, 1(1):30–46, 1996.

[109] Kaouther Messaoud, Nachiket Deo, Mohan M Trivedi, and Fawzi Nashashibi. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 165–170, 2021.

[110] Brian Mok, Mishel Johns, Key Jung Lee, David Miller, David Sirkin, Page Ive, and Wendy Ju. Emergency, automation off: Unstructured transition timing for distracted drivers of automated vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2458–2464. IEEE, 2015.

[111] Brian Ka-Jun Mok, Mishel Johns, Key Jung Lee, Hillary Page Ive, David Miller, and Wendy Ju. Timing of unstructured transitions of control in automated driving. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1167–1172. IEEE, 2015.

[112] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[113] Brendan Tran Morris and Mohan Manubhai Trivedi. Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2287–2301, 2011.

[114] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Hyhope: Hybrid head orientation and position estimation for vision-based driver head tracking. In *2008 IEEE Intelligent Vehicles Symposium*, pages 512–517. IEEE, 2008.

[115] Rizwan Ali Naqvi, Muhammad Arsalan, Ganbayar Batchuluun, Hyo Sik Yoon, and Kang Ryoung Park. Deep learning-based gaze detection system for automobile drivers using a nir camera sensor. *Sensors*, 18(2):456, 2018.

[116] Frederik Naujoks, Christian Purucker, Katharina Wiedemann, and Claus Marberger. Noncritical state transitions during conditionally automated driving on german freeways: Effects of non–driving related tasks on takeover time and takeover quality. *Human factors*, 61(4):596–613, 2019.

[117] Matthew Niedoba, Henggang Cui, Kevin Luo, Darshan Hegde, Fang-Chieh Chou, and Nemanja Djuric. Improving movement prediction of traffic actors using off-road loss and bias mitigation. In *Machine Learning for Autonomous Driving (ML4AD) Workshop, NeurIPS*, 2019.

[118] Julia Nilsson, Jonatan Silvlin, Mattias Brannstrom, Erik Coelingh, and Jonas Fredriksson. If, when, and how to perform lane change maneuvers on highways. *IEEE Intelligent Transportation Systems Magazine*, 8(4):68–78, 2016.

[119] nuScenes Contributors. nuScenes. https://www.nuscenes.org/, 2020.

[120] Eshed Ohn-Bar, Sujitha Martin, Ashish Tawari, and Mohan M Trivedi. Head, eye, and hand patterns for driver activity recognition. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 660–665. IEEE, 2014.

[121] Eshed Ohn-Bar, Ashish Tawari, Sujitha Martin, and Mohan M Trivedi. Predicting driver maneuvers by learning holistic features. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 719–724. IEEE, 2014.

[122] Eshed Ohn-Bar and Mohan Trivedi. In-vehicle hand activity recognition using integration of regions. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1034–1039. IEEE, 2013.

[123] Eshed Ohn-Bar and Mohan M Trivedi. Beyond just keeping hands on the wheel: Towards visual interpretation of driver hand motion patterns. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1245–1250. IEEE, 2014.

[124] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014.

[125] Justin M Owens, Linda Angell, Jonathan M Hankey, James Foley, and Kazutoshi Ebe. Creation of the naturalistic engagement in secondary tasks (nest) distracted driving dataset. *Journal of safety research*, 54:33–e29, 2015.

[126] Erfan Pakdamanian, Shili Sheng, Sonia Baee, Seongkook Heo, Sarit Kraus, and Lu Feng. Deeptake: Prediction of driver takeover behavior using multimodal data. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2021.

[127] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[128] Sebastiaan Petermeijer, Pavlo Bazilinskyy, Klaus Bengler, and Joost De Winter. Take-over again: Investigating multimodal and directional tors to get the driver back into the loop. *Applied ergonomics*, 62:204–215, 2017.

[129] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020.

[130] Derek J Phillips, Tim A Wheeler, and Mykel J Kochenderfer. Generalizable intention prediction of human drivers at intersections. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1665–1670, 2017.

[131] Aris Polychronopoulos, Manolis Tsogas, Angelos J Amditis, and Luisa Andreone. Sensor fusion for predicting vehicles' path for collision avoidance systems. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):549–562, 2007.

[132] Jonas Radlmayr, Christian Gold, Lutz Lorenz, Mehdi Farid, and Klaus Bengler. How traffic situations and non-driving related tasks affect the take-over quality in highly automated driving. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 58, pages 2063–2067. Sage Publications Sage CA: Los Angeles, CA, 2014.

[133] Akshay Rangesh, Nachiket Deo, Ross Greer, Pujitha Gunaratne, and Mohan M Trivedi. Autonomous vehicles that alert humans to take-over controls: Modeling with real-world data. *arXiv preprint arXiv:2104.11489*, 2021.

[134] Akshay Rangesh, Nachiket Deo, Kevan Yuen, Kirill Pirozhenko, Pujitha Gunaratne, Heishiro Toyoda, and Mohan M Trivedi. Exploring the situational awareness of humans inside autonomous vehicles. In *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018.

[135] Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani, and Mohan M Trivedi. Trackmpnn: A message passing graph neural architecture for multi-object tracking. *arXiv preprint arXiv:2101.04206*, 2021.

[136] Akshay Rangesh, Eshed Ohn-Bar, and Mohan M Trivedi. Hidden hands: Tracking hands with an occlusion aware tracker. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–26, 2016.

[137] Akshay Rangesh and Mohan Trivedi. Forced spatial attention for driver foot activity classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[138] Akshay Rangesh and Mohan Trivedi. Forced spatial attention for driver foot activity classification. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[139] Akshay Rangesh and Mohan M Trivedi. Handynet: A one-stop solution to detect, segment, localize & analyze driver hands. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1103–1110, 2018.

[140] Akshay Rangesh and Mohan Manubhai Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars. *IEEE Transactions on Intelligent Vehicles*, 4(4):588–599, 2019.

[141] Akshay Rangesh and Mohan Manubhai Trivedi. Ground plane polling for 6dof pose estimation of objects on the road. *IEEE Transactions on Intelligent Vehicles*, 5(3):449–460, 2020.

[142] Akshay Rangesh, Kevan Yuen, Ravi Kumar Satzoda, Rakesh Nattoji Rajaram, Pujitha Gunaratne, and Mohan M Trivedi. A multimodal, full-surround vehicular testbed for naturalistic studies and benchmarking: Design, calibration and deployment. *arXiv preprint arXiv:1709.07502*, 2017.

[143] Akshay Rangesh, Bowen Zhang, and Mohan M Trivedi. Driver gaze estimation in the real world: Overcoming the eyeglass challenge. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1054–1059. IEEE, 2020.

[144] Michael A Regan, Charlene Hallett, and Craig P Gordon. Driver distraction and driver inattention: Definition, relationship and taxonomy. *Accident Analysis & Prevention*, 43(5):1771–1781, 2011.

[145] SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, 2018.

[146] Tara Rezvani, Katherine Driggs-Campbell, Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Ruzena Bajcsy. Towards trustworthy automation: User interfaces that convey internal and external awareness. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 682–688. IEEE, 2016.

[147] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

[148] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *IEEE/CVF International Conference on Computer Vision*, pages 2821–2830, 2019.

[149] Daniela Ridel, Nachiket Deo, Denis Wolf, and Mohan Trivedi. Scene compliant trajectory forecast with agent-centric spatio-temporal grids. *IEEE Robotics and Automation Letters*, 5(2):2816–2823, 2020.

[150] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision (ECCV)*, pages 549–565. Springer, 2016.

[151] Alina Roitberg, Monica Haurilet, Simon Reiß, and Rainer Stiefelhagen. Cnn-based driver activity understanding: Shedding light on deep spatiotemporal representations. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[152] Alina Roitberg, Chaoxiang Ma, Monica Haurilet, and Rainer Stiefelhagen. Open set driver activity recognition. In *Intelligent Vehicles Symposium (IV)*. IEEE, 2020.

[153] A Sadeghian, V Kosaraju, A Gupta, S Savarese, and A Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018.

[154] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. *arXiv preprint arXiv:1806.01482*, 2018.

[155] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision (ECCV)*, pages 683–700, 2020.

[156] Julian Schlechtriemen, Andreas Wedel, Gabi Breuel, and Klaus-Dieter Kuhnert. A probabilistic long term prediction approach for highway scenarios. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 732–738, 2014.

[157] Julian Schlechtriemen, Florian Wirthmueller, Andreas Wedel, Gabi Breuel, and Klaus-Dieter Kuhnert. When will it change the lane? a probabilistic regression approach for rarely occurring events. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1373–1379, 2015.

[158] Matthias Schreier, Volker Willert, and Jürgen Adamy. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 334–341, 2014.

[159] Robin Schubert, Eric Richter, and Gerd Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *IEEE International Conference on Information Fusion*, pages 1–6, 2008.

[160] Victor A Shia, Yiqi Gao, Ramanarayan Vasudevan, Katherine Driggs Campbell, Theresa Lin, Francesco Borrelli, and Ruzena Bajcsy. Semiautonomous vehicular control using driver modeling. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2696–2709, 2014.

[161] Sayanan Sivaraman and Mohan M Trivedi. Towards cooperative, predictive driver assistance. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1719–1724, 2013.

[162] Sayanan Sivaraman and Mohan Manubhai Trivedi. Dynamic probabilistic drivability maps for lane change and merge driver assistance. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2063–2073, 2014.

[163] Han-Shue Tan and Jihua Huang. Dgps-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):415–428, 2006.

[164] Ashish Tawari, Kuo Hao Chen, and Mohan M Trivedi. Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation. In *Intelligent transportation systems (ITSC), 2014 IEEE 17th international conference on*, pages 988–994. IEEE, 2014.

[165] Ashish Tawari, Sujitha Martin, and Mohan Manubhai Trivedi. Continuous head movement estimator for driver assistance: Issues, algorithms, and on-road evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):818–830, 2014.

[166] Ashish Tawari and Mohan M Trivedi. Robust and continuous estimation of driver gaze zone by dynamic analysis of multiple face videos. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 344–349. IEEE, 2014.

[167] Rafael Toledo-Moreo and Miguel A Zamora-Izquierdo. Imm-based lane-change prediction in highways with low-cost gps/ins. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):180–185, 2009.

[168] Cuong Tran, Anup Doshi, and Mohan M Trivedi. Pedal error prediction by driver foot gesture analysis: A vision-based inquiry. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 577–582. IEEE, 2011.

[169] Cuong Tran, Anup Doshi, and Mohan Manubhai Trivedi. Modeling and prediction of driver behavior by foot gesture analysis. *Computer Vision and Image Understanding*, 116(3):435–445, 2012.

[170] Quan Tran and Jonas Firl. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 918–923, 2014.

[171] Mohan M Trivedi et al. Attention monitoring and hazard assessment with bio-sensing and vision: Empirical analysis utilizing cnns on the kitti dataset. *arXiv preprint arXiv:1905.00503*, 2019.

[172] Simon Ulbrich and Markus Maurer. Towards tactical lane change behavior planning for automated vehicles. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 989–995, 2015.

[173] Borhan Vasli, Sujitha Martin, and Mohan Manubhai Trivedi. On driver gaze estimation: Explorations and fusion of geometric and data driven approaches. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 655–660. IEEE, 2016.

[174] Dizan Vasquez and Thierry Fraichard. Motion prediction for moving objects: a statistical approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3931–3936, 2004.

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[176] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[177] Sourabh Vora, Akshay Rangesh, and Mohan M Trivedi. On generalizing driver gaze zone estimation using convolutional neural networks. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 849–854. IEEE, 2017.

[178] Sourabh Vora, Akshay Rangesh, and Mohan Manubhai Trivedi. Driver gaze zone estimation using convolutional neural networks: A general framework and ablative analysis. *IEEE Transactions on Intelligent Vehicles*, 3(3):254–265, 2018.

[179] Chuhua Wang, Yuchen Wang, Mingze Xu, and David J Crandall. Stepwise goal-driven networks for trajectory prediction. *arXiv preprint arXiv:2103.14107*, 2021.

[180] Jürgen Wiest, Matthias Höffken, Ulrich Kreßel, and Klaus Dietmayer. Probabilistic trajectory prediction with gaussian mixture models. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 141–146, 2012.

[181] Martin Wollmer, Christoph Blaschke, Thomas Schindl, Björn Schuller, Berthold Farber, Stefan Mayer, and Benjamin Trefflich. Online driver distraction detection using long short-term memory. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):574–582, 2011.

[182] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[183] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2089–2095, 2016.

[184] Kevan Yuen, Sujitha Martin, and Mohan M Trivedi. Looking at faces in a vehicle: A deep cnn based approach and evaluation. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 649–654. IEEE, 2016.

[185] Kevan Yuen and Mohan M Trivedi. Looking at hands in autonomous vehicles: A convnet approach using part affinity fields. *arXiv preprint arXiv:1804.01176*, 2018.

[186] Kevan Yuen and Mohan M Trivedi. Looking at hands in autonomous vehicles: A convnet approach using part affinity fields. *IEEE Transactions on Intelligent Vehicles*, 2019.

[187] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[188] Kathrin Zeeb, Axel Buchner, and Michael Schrauf. What determines the take-over time? an integrated model approach of driver take-over after automated driving. *Accident Analysis & Prevention*, 78:212–221, 2015.

[189] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653*, 2021.

[190] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[191] Lingyao Zhang, Po-Hsun Su, Jerrick Hoang, Galen Clark Haynes, and Micol Marchetti-Bowick. Map-adaptive goal-based trajectory prediction. In *Conference on Robot Learning (CoRL)*, pages 1371–1383. PMLR, 2021.

[192] Yanfu Zhang, Wenshan Wang, Rogerio Bonatti, Daniel Maturana, and Sebastian Scherer. Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories. In *Conference on Robot Learning (CoRL)*, pages 894–905, 2018.

[193] Zutao Zhang, Dianyuan Luo, Yagubov Rasim, Yanjun Li, Guanjun Meng, Jian Xu, and Chunbai Wang. A vehicle active safety model: vehicle speed control based on driver vigilance detection using wearable eeg and sparse representation. *Sensors*, 16(2):242, 2016.

[194] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning (CoRL)*, pages 895–904. PMLR, 2021.

[195] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12126–12134, 2019.

[196] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning (ICML)*, 2010.

[197] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438, 2008.

[198] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3931–3936. IEEE, 2009.

[199] Alex Zyner, Stewart Worrall, and Eduardo Nebot. Naturalistic driver intention and path prediction using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 2019.