

Driving Behavior Analysis and Prediction for Safe Autonomous Vehicles

by

Mozhgan Nasr Azadani

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements
for the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Mozhgan Nasr Azadani, Ottawa, Canada, 2024

Abstract

Driving Behavior Analysis (DBA) plays a pivotal role in designing intelligent transportation systems, enhancing road safety, and advancing Autonomous Vehicles (AVs). Driver identification, as a key aspect of DBA, has the potential to provide unprecedented opportunities for enhanced security and driver profiling. However, the current solutions for driver identification suffer from demanding extensive data collection, limited scalability, and inadequate generalization. Furthermore, DBA is also essential for training AVs, addressing the main challenges they face: accurately perceiving their surroundings to make informed decisions and to navigate safely, and effectively handling unforeseen scenarios.

In the first part of this thesis, we concentrate on behavior analysis for driver identification and verification and design two novel schemes aiming to reduce data dependency and enhance the generalization ability of existing approaches. First, we propose a novel driver identification model, called DriverRep, which reduces data dependency by presenting a fully unsupervised triplet loss training. DriverRep is the first model that extracts the latent representations associated with each driver, called driver embeddings, in an unsupervised manner. In addition, we develop a novel model to tackle driver verification and impostor detection tasks based on DBA and extracted driver embeddings.

In the second part, we focus on behavior prediction for AVs and their surrounding agents. First, we tackle behavior prediction in dynamic and complex scenarios by introducing three novel prediction models for forecasting drivers intentions and behaviors at unsignalized intersections. We then address social reasoning by proposing a novel prediction model that analyzes agent interactions using graph neural networks, making the scene understanding process more informative for AVs. Our proposed prediction model, called STAG, explicitly activates social modeling with a directed graph representation while considering spatial and temporal inter-agent correlations. We further design a novel prediction system, namely CAPHA, which conditions the future behavior of agents on grid-based plans modeled as a Markov decision process and solves the prediction task via inverse reinforcement learning to produce scene compliant behaviors. Moreover, we introduce a novel goal-based prediction model, called GMP, which encodes interactions between agents and dynamic and static context information to estimate the distribution of target goals, efficiently considering the inherent uncertainty in agents behavior.

Extensive quantitative and qualitative comparisons have been conducted between the developed solutions and related benchmark schemes using various scenarios and environments. The obtained results demonstrate the potential of these solutions for the understudy tasks of DBA and real-world applications.

List of Publications

Referred Journal Papers

- M. Nasr Azadani, A. Boukerche, "CAPHA: context-aware path prediction of heterogeneous agents," *IEEE Transactions on Vehicular Technology*, 2023. Accepted
- M. Nasr Azadani, A. Boukerche, "STAG: a novel interaction-aware path prediction method based on spatio-temporal attention graphs for connected automated vehicles," *Elsevier Adhoc Networks*, 138, p. 103021, 2023.
- M. Nasr Azadani, A. Boukerche, "DriverRep: driver identification through driving behavior embeddings," *Elsevier Journal of Parallel and Distributed Computing*, 162, p. 105-117, 2022.
- M. Nasr Azadani, A. Boukerche, "Siamese temporal convolutional networks for driver identification using driver steering behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, 23 (10), p. 18076-18087, 2022.
- M. Nasr Azadani, A. Boukerche, "A novel multimodal vehicle path prediction method based on temporal convolutional networks," *IEEE Transactions on Intelligent Transportation Systems*, 23 (12), p. 25384-25395, 2022.
- M. Nasr Azadani, A. Boukerche, "Driving behavior analysis guidelines for intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, 23 (7), p. 6027-6045, 2021.

Referred Conference Papers

- M. Nasr Azadani, A. Boukerche, "GMP: goal-based multimodal motion prediction for automated vehicles," *2023 IEEE Global Communications Conference (GLOBECOM)*, 2023. Accepted.
- M. Nasr Azadani, A. Boukerche, "A context-Aware path forecasting method for connected autonomous vehicles," *2023 IEEE International Conference on Communications (ICC)*, p. 247-252, 2023.
- M. Nasr Azadani, A. Boukerche, "An interaction-Aware vehicle behavior prediction for connected automated vehicles," *2022 IEEE International Conference on Communications (ICC)*, p. 279-284, 2022.
- M. Nasr Azadani, A. Boukerche, "Convolutional and recurrent neural networks for driver identification: an empirical study," *2022 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, p. 1-6, 2022.
- M. Nasr Azadani, A. Boukerche, "An internet-of-vehicles powered defensive driving warning approach for traffic safety," *2021 IEEE Global Communications Conference (GLOBECOM)*, p. 1-6, 2021.
- M. Nasr Azadani, A. Boukerche, "Toward driver intention prediction for intelligent vehicles: a deep learning approach," *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, p. 233-240, 2021.
- M. Nasr Azadani, A. Boukerche, "Driver identification using vehicular sensing data: a deep learning approach," *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, p. 1-6, 2021.
- M. Nasr Azadani, A. Boukerche, "Performance evaluation of driving behavior identification models through CAN-BUS data," *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, p. 1-6, 2020.

Acknowledgements

First, I would like to start by expressing my profound gratitude to my exceptional supervisor, Prof. Azzedine Boukerche, for your unstinting support, patient guidance, and invaluable insights throughout my Ph.D. journey. Your mentorship has not only enriched my research skills but has also instilled in me a deep passion for inquiry and discovery. Your patience, encouragement, financial support, and constructive feedback have continuously pushed me to strive for excellence. I am truly grateful for the opportunity to work under your guidance and the trust you have placed in me.

To my beloved husband, Isar, words cannot adequately convey the depth of my appreciation for your unconditional love, endless sacrifices, and unyielding belief in me. This accomplishment would not be possible without your endless encouragement. Thank you for being my constant inspiration and for standing by my side every step of the way!

To my parents and family, from the very beginning, you have been my pillars of strength, providing constant support, encouragement, and understanding. Your love and the values you have taught me have shaped the person I am today. I am forever indebted to you. Thank you very much!

To my dear friends, both near and distant, your unshakable friendship, shared laughter, and unconditional support have made this journey unforgettable. I would like to send my special thanks to Negar. Your understanding during moments of stress and celebration during moments of triumph has made all the difference. Thank you for the countless long-distance calls that provided me with motivation and encouragement! Additionally, I express my deep appreciation to Diana, Zohreh, Alireza, Shavbo, and Atefeh for their invaluable support.

I would also like to thank to my colleagues at the PARADISE research lab. Your friendship, shared knowledge, and intellectual discussions have made my university experience truly memorable. Your continuous support, both academically and emotionally, has been instrumental in overcoming challenges and reaching milestones. Thank you all!

Last but certainly not least, I want to thank my dogs, Leo and Nancy, for being a constant source of joy and solace! Though you cannot read this, your wagging tails and gentle companionship provided much-needed respite during long hours of study and moments of stress!

Dedication

To my dear husband, Isar,
for your unconditional support, patience and faith in me!

Contents

1	Introduction	1
1.1	Challenges and Motivation	2
1.1.1	Behavior Analysis for ADAS	2
1.1.2	Behavior Prediction for AVs	3
1.2	Objectives	5
1.3	Contributions	6
1.4	Outline	8
2	Driving Behavior Analysis for Intelligent Transportation Systems	9
2.1	Introduction	10
2.2	Driving Behavior Data	11
2.2.1	Data Sources	11
2.2.2	Datasets	15
2.3	Driving Behavior Models	20
2.3.1	Threshold-based and Fuzzy Logic-based Methods	20
2.3.2	Physics-based Methods	21
2.3.3	Classical Machine Learning-based Methods	23
2.3.4	Deep Learning-based methods	25
2.3.5	Reinforcement Learning-based Methods	27
2.4	Driving Behavior Tasks	28
2.4.1	Driver Identification	28
2.4.2	Driver Inattention Detection	40
2.4.3	Driving Style Detection	43
2.4.4	Impaired Driving Detection	45
2.4.5	Driving Behavior Detection and Prediction	47
2.5	Concluding Remarks	64

I	Driver Identification and Verification for ADAS	65
3	Driving Behavior Identification through CAN-BUS Data	66
3.1	Introduction	67
3.2	Methodology and Background	69
3.2.1	An Overview of the Dataset	69
3.2.2	Driver Identification Classification Models	72
3.3	Performance Evaluation	75
3.4	Discussion	76
3.5	Concluding Remarks	78
4	DriverRep: Driver Identification through Driving Behavior Embed- dings	79
4.1	The Proposed Model	80
4.1.1	Driver Embeddings	81
4.1.2	DriverRep Architecture	84
4.2	Performance Evaluation	90
4.2.1	Datasets	90
4.2.2	Multi-way Driver Identification Results	91
4.2.3	Comparison Results	94
4.2.4	Ablation Analysis	100
4.3	Concluding Remarks	105
5	Siamese TCNs for Driver Identification using Driver Steering Behavior Analysis	107
5.1	System Overview	108
5.2	Proposed Methodology	110
5.2.1	Driver Identification	110
5.2.2	Driver Verification	114
5.2.3	Impostor Detection	116
5.3	Evaluation Results and Discussion	116
5.3.1	Evaluation Results for Identification	117
5.3.2	Verification and Impostor Detection Results	124
5.4	Concluding Remarks	128

II Agents Behavior Prediction for Autonomous Vehicles 130

6 Driver Intention and Vehicle Behavior Prediction at Unsignalized Intersections	131
6.1 Driver Intention Prediction	133
6.1.1 Problem Formulation	133
6.1.2 Methodology	136
6.1.3 Driver Intention Evaluation Results	139
6.2 Interaction-aware Driver Behavior Prediction	144
6.2.1 Problem Statement	145
6.2.2 Methods	146
6.2.3 Evaluation Results	150
6.3 Multimodal Vehicle Behavior Prediction	154
6.3.1 Problem Statement	155
6.3.2 Methodology	158
6.3.3 Driver Behavior Prediction Experimental Evaluation	164
6.4 Concluding Remarks	176
7 STAG: Spatio-Temporal Attention Graphs for Interaction-Aware Agent Behavior Prediction	177
7.1 Background	179
7.1.1 GNNs	179
7.1.2 Attention Mechanism	180
7.2 Problem Definition	181
7.3 Methodology	182
7.3.1 STAG Overview	182
7.3.2 Feature Extraction	182
7.3.3 Encoder Module	184
7.3.4 Decoder Module	188
7.4 Experimental Results	189
7.4.1 Evaluation Datasets	189
7.4.2 Evaluation Metrics	191
7.4.3 Experimental Settings	191
7.4.4 Baseline Models	192
7.4.5 Quantitative Evaluation	193
7.4.6 Qualitative Evaluation	199

7.5	Concluding Remarks	200
8	CAPHA: Context-Aware Behavior Prediction of Heterogeneous Agents	201
8.1	Background	203
8.1.1	Markov Decision Process	203
8.1.2	Inverse Reinforcement Learning	203
8.2	Problem Formulation	204
8.3	CAPHA for Behavior Prediction	205
8.3.1	Context Encoder	207
8.3.2	Policy Network	208
8.3.3	Behavior Decoder	210
8.4	Evaluation Results	212
8.4.1	Setting	212
8.4.2	Results	216
8.5	Concluding Remarks	225
9	GMP: Goal-Based Multimodal Motion Prediction for AVs	226
9.1	Proposed Model	227
9.1.1	Formulation	227
9.1.2	Goal-based Motion Prediction	227
9.1.3	Behavior History Encoding	228
9.1.4	Context Encoding	229
9.1.5	Aggregation	230
9.1.6	Goal Prediction	231
9.1.7	Behavior Estimation	231
9.2	Experiments	232
9.2.1	Setting	232
9.2.2	Results	234
9.3	Concluding Remarks	236
10	Conclusion and Future Directions	237
10.1	Contribution Summary	237
10.2	Future Research Directions	239

List of Tables

2.1	Commonly used data sources for DBA	12
2.2	A summary of available datasets for driving behavior analysis	16
2.3	Available datasets for driving behavior detection and prediction	19
2.4	Classification of the main tasks/purposes of DBA	29
2.5	Overview of DBA-based driver identification methods	32
2.6	Driver identification features comparison (ND: Number of drivers, S: scalability, G: generalization, DD: data dependency, H: High, M: Medium, L: Low)	34
2.7	An overview of social reasoning methodologies for DB prediction	49
2.8	A summary of interaction-aware behavior prediction methods (Mu: Multimodal, CA: Context-aware)	50
2.9	Methods of context modeling for driving behavior prediction	55
2.10	An overview of Context-aware behavior forecasting methods (MU: Multimodal, I: Interaction-aware)	57
2.11	A summary of multimodal behavior anticipation approaches (Ca: Context-aware, I: Interaction-aware)	59
3.1	Evaluation results (window size = 60).	76
4.1	Average driver identification multi-way accuracy of the DriverRep model.	93
4.2	List of optimized hyper-parameters for the deep learning approaches where b represents the number of blocks.	99
4.3	Prediction accuracy (ten-way) of different time series classification models over the HCI-lab dataset.	99
4.4	Driver identification accuracy scores for different encoder architectures with respect to three different representation size on HCI-lab.	103
5.1	The main characteristics of the used subset of UYANIK dataset.	110

5.2	Hyper-parameter settings for evaluations.	118
5.3	Average accuracy results from 1000 combinations per group size for driver identification.	122
5.4	Evaluation results of an open-set setting impostor detection for different number of drivers.	128
6.1	Extracted maneuvers according to approaching direction-destination pair. For instance, the term East-South refers to east as origin and south as destination	135
6.2	Parameters settings for the introduced TCN-based model used for experiments	140
6.3	Comparison results for different approaches.	152
6.4	Number of employed maneuvers according to the relative destination of the vehicle motion.	157
6.5	Optimized hyper-parameters settings	165
6.6	Comparison results of all models over left maneuvers.	169
6.7	Evaluation results for all compared models over straight maneuvers. . . .	170
6.8	Comparison results of all models over right maneuver trajectories.	171
6.9	Evaluation results for all models over all existing maneuvers in the trajectories.	172
7.1	Evaluation results of structured highways scenario over NGSIM dataset for various state-of-the-art approaches. RMSE values in meters are reported for 3s observation. STAG outperforms all baselines.	194
7.2	Comparison results (ADE / FDE) over inD dataset for obs=8 and H=12 steps.	195
7.3	ADE / FDE results for different types of agents involved within a scene over inD dataset.	195
7.4	Experimental results of validation on unseen scenario i.e., roundabouts (R1-R4) from INTERACTION dataset. ADE / FDE in meters are reported for obs=8 steps.	197
7.5	Ablation analysis of mask threshold ε for prediction steps of 10 and 12. .	197
8.1	Comparison results (ADE / FDE) over inD dataset for observation=8 and horizon=12.	218

8.2	ADE / FDE results over inD dataset categorized by agents types and interactions.	218
8.3	ADE / FDE over inD dataset per agent type.	219
8.4	Long-term behavior prediction over inD dataset (pedestrians agents and the split used in Y-net.)	220
8.5	ADE / FDE results over SDD dataset (all).	221
8.6	Results on Trajnet split of the SDD dataset.	221
8.7	Ablation analysis over SDD dataset.	222
8.8	Effect of feature map size over SDD dataset.	222
8.9	Inference and training time comparison results. Training time is in s/epoch for all modules of the methods.	223
9.1	Evaluation results on the test set of Argoverse 1. The best results are shown in bold.	233

List of Figures

2.1	Taxonomy of driving behavior analysis studies	11
2.2	Taxonomy of driving behavior detection and prediction	48
3.1	Non-intrusive driver identification framework: an illustrative example. . .	68
3.2	A sample feature visualization for two groups of drivers.	70
3.3	Sliding window with the window size of 100 and the time step of 50. . .	71
3.4	Exploring feature correlations: Pearson correlation coefficient analysis for selected features.	72
3.5	Schematic diagram of the CNN model containing two convolutional and maxpooling layers.	73
3.6	The training accuracy and loss (window size = 60).	77
3.7	The training accuracy and loss (window size = 45).	77
4.1	An overview of the proposed driver identification system, called Driver- Rep, and its model architecture.	81
4.2	The triplet loss training illustrative example.	82
4.3	Driving instances before training and the obtained driver embeddings after training.	85
4.4	(left) Architecture of the i^{th} layer of the DriverRep model. (right) illus- tration of stacked dilated convolutions.	89
4.5	A sample of HCRL dataset from driver A.	91
4.6	An example of speed signal from HCI-lab dataset for three drivers. . . .	92
4.7	Extracted driver embeddings by the DriverRep for two datasets. (a) three- way identification (b) five-way identification and (c) ten-way identification.	94
4.8	Confusion matrix of ten-way identification over the HCI-lab dataset. . . .	95

4.9	Comparison between the instances of four driver pairs. Each alphabetical letter represents the results for the same two drivers. Parts 4.9a1- 4.9d1 demonstrate the original driving instances, i.e., the input. Parts 4.9a2- 4.9d2 illustrate their extracted latent representation.	96
4.10	Comparison of different driver identification schemes in terms of accuracy.	97
4.11	Prediction results of sparse labeling.	100
4.12	The effect of different embedding sizes on the performance (three-way) of the DriverRep with $neg = 10$	101
4.13	The model ablation study results (three-way) for the encoder architecture of the DriverRep ($neg = 10$).	104
4.14	Comparing extracted driver embeddings by DriverRep and RNN for four drivers.	105
5.1	Steering wheel angles (in degrees) of three different drivers at the same turn.	108
5.2	General flow of the high-level driver identification system architecture. . .	109
5.3	Total driving duration for each driver in the used dataset.	111
5.4	Histogram of steering wheel angles (SWA), yaw rate (YR) and steering wheel relative speed (SWRS) for two groups of drivers.	111
5.5	Architecture of the deep neural network for driver identification.	113
5.6	Illustration of the window sampling technique for a sample signal.	114
5.7	System architecture for the driver verification task.	115
5.8	System architecture for impostor detection task.	117
5.9	Comparison results on driver identification ($n = 3$).	119
5.10	Normalized confusion matrix for driver identification ($n = 60$).	123
5.11	Normalized CMC curve for identification ($n = 30$).	124
5.12	Effect of window length on accuracy ($n = 3$).	124
5.13	Evaluation results for 57 drivers in terms of four metrics.	125
5.14	ROC curve for driver verification.	126
5.15	Two-dimensional projection of the encodings for drivers that the model had (left) and had not (right) difficulty verifying.	127
6.1	A sample of vehicles trajectories within the unsignalized T-junction. . . .	134

6.2	Extracted maneuvers from each trajectory according to the relative position from the center of the T-junction (i.e., (0.0) refers to the center of the intersection) denoted by origin-destination pair. The first sample for each maneuver is shown in bold color. The relative positional data are rotated for visualization purposes.	136
6.3	Relative position, speed, and heading of the data for all maneuvers originated from the south (i.e., south-west and south-east). The first sample of each category is shown in bold. Smoothed speed values are calculated for visualization.	137
6.4	Difference between the architecture of causal CNNs with and without dilation.	138
6.5	Intent prediction results for the scenarios where vehicles are moving toward the T-junction from south.	141
6.6	Prediction cross-validation results for the scenarios where vehicles are reaching the T-junction from east.	142
6.7	The scenarios where vehicles are approaching the T-junction from west. Top: the prediction accuracy result. Bottom: Speed of the vehicles for both maneuvers. One sample of each category is shown in bold. Smoothed speed values are calculated for visualization.	143
6.8	The interaction-aware prediction problem for CAVs i.e., the occlusion challenge.	145
6.9	A broad view of the architecture of our vehicle behavior prediction system.	147
6.10	Graph-based modeling of TV and SVs interactions (left), and graph-based convolution process (right).	149
6.11	Example cases from the unsignalized intersections. The positions of the TVs are relative to the world space.	151
6.12	Ablation analysis results for the proposed method.	153
6.13	Qualitative results over the MA intersection.	153
6.14	Multimodal output of the vehicle behavior prediction with the horizon of 4.8 seconds. Different modes and probabilities are shown in blue and heatmaps, respectively.	155
6.15	Different maneuvers occurred at one of the roundabouts in the dataset. The samples are taken from different origin-destination pairs for each maneuver. The used dataset is collected in Sydney, where the vehicles move clockwise.	158

6.16	Average speed and heading over all used trajectories.	159
6.17	High-level architecture of the proposed vehicle behavior prediction model.	160
6.18	Structure of the residual blocks.	162
6.19	Qualitative analysis of behavior prediction results for our proposed model. Trajectories are shown from twelve different vehicles when carrying out various maneuvers: straight (a1-a4), right (b1-b4), and left (c1-c4). Dif- ferent mode and probability outputs are shown in blue and heatmaps, respectively. Unimodal baseline results are demonstrated for comparison.	174
6.20	Predicted trajectories for two sample vehicles while carrying out a right- turn maneuver. Each row represents a complete sequence of predictions at various distances (i.e., -5, 0, 5, 10, and 20 meters) from the entrance of the roundabout. Negative distance indicates the target vehicle has not entered the roundabout yet.	175
7.1	Graph-based interaction-aware behavior prediction while considering sym- metric and asymmetric social interactions between the target (TA) and surrounding agents (SA)s.	178
7.2	An overview of the architecture of STAG including three main steps: (1) The feature extraction that is responsible for extracting the agents states and relations from the agents motion history using MLP; (2) The en- coder module that consists of graphs construction, learning asymmetric masks through a mask block (MB), spatial and temporal attention layers to model spatio-temporal interactions, and graph convolutions for obtain- ing trajectory representations; (3) The decoder module including blocks of TCNs to generate future behavior distribution.	183
7.3	An illustration of the mask block of STAG with attention mechanism to compute the asymmetric masks.	186
7.4	An illustration of spatial attention module to model the spatial interac- tions for the target vehicle v_1	187
7.5	Qualitative analysis of predicted trajectories for the proposed STAG model over different scenarios: inD/Bendplatz (a1-a3), inD/Frankenburg (a4- a5), inD/Heckstrasse (a6), and INTERACTION/Roundabout 1 (a7-a8). The red and blue/star lines indicate the observation and true behaviors, respectively. The predicted distribution and the best sample are shown with green masks and yellow/circle lines, respectively.	198

7.6	Qualitative analysis of the effect of vehicles mobility over INTERACTION dataset (Roundabout 2).	199
8.1	Multiple modes of the future behavior. The predicted behaviors are scene-compliant.	202
8.2	An overview of CAPHA with three primary modules: (1) The context encoder to embed the motion and context features and learn the IRL rewards; (2) The policy network to compute the policy maps through the value iteration network; (3) The behavior decoder module, including stacked transformers and MLP to forecast the distribution of SAs' future motions.	206
8.3	An illustration of the plan transformer. The local context, SA features at each state $s_i^k \in s^k$, and location coordinates (x, y) are encoded using the plan transformer.	211
8.4	Qualitative analysis of the forecast behaviors for the introduced CAPHA model over several trajectory scenes: the agent-centered BEV images (a1-e1), the goals (a2-e2), the behavior SVFs (a3-e3), the predicted distribution (a4-e4), the multimodal outputs with $k=20$ (a5-e5), and the multimodal outputs with $k=5$ (a6-e6). Observations are shown with red lines. The predicted and real behaviors are represented as blue/star and green/circle lines, respectively.	224
9.1	Illustration of GMP. I-I denotes the interactions where A, L, and R represent agents, lanes, and ROIs, respectively.	228
9.2	Qualitative analysis on Argoverse 1 for various scenarios. The behavior history and $K = 6$ anticipated behaviors are shown in red and blue, respectively. The TAs' true behaviors are shown in green. Stars represent the estimated final endpoint.	235
9.3	Probability scores of $K = 6$ estimated modes.	235

Glossary of Terms

ADAS Advanced Driver-Assistance Systems

ADE Average Displacement Error

AE Autoencoder

AV Autonomous Vehicle

BEV Bird’s Eye View

CA Constant Acceleration

CAN Controller Area Network

CAV Connected Automated vehicle

CV Constant Velocity

CNN Convolutional Neural Network

CVAE Conditional Variational Autoencoder

DB Driving Behavior

DBA Driving Behavior Analysis

DNN Deep Neural Network

DT Decision Tree

FCN Fully Convolutional Network

FDE Final Displacement Error

GB Gradient Boosting

GAN Generative Adversarial Network

GAT Graph Attention Network

GC Graph Convolutions

GCN Graph Convolutional Network

GMM Gaussian Mixture Model

GNN Graph Neural Network

GPR Gaussian Process Regression

GRU Gated Recurrent Unit

HD High-Definition

HMM Hidden Markov Model

IMU Inertial Measurement Unit

IRL Inverse Reinforcement Learning

KF Kalman Filtering

KNN K-Nearest Neighbors

LSTM Long Short-Term Memory

LVQ Learning Vector Quantization

ME-IRL Maximum Entropy Inverse Reinforcement Learning

MDN Mixture Density Network

MHD Modified Hausdorff Distance

MHA Multi-Head Attention

MDP Markov Decision Process

MLP Multi-Layer Perceptron

NB Naive Bayes

NN Neural Network

OBD On Board Diagnostics

PRU Prototypical Recurrent Unit

SA Surrounding Agent
SV Surrounding Vehicle
SVF State Visitation Frequencies
SVM Support Vector Machine
SVDD Support Vector Domain Description
RF Random Forest
RL Reinforcement Learning
RMSE Root Mean Square Error
RNN Recurrent Neural Network
TA Target Agent
TV Target Vehicle
TCN Temporal Convolutional Network
VAE Variational Autoencoder
VIN Value Iteration Network

Chapter 1

Introduction

The evolution of vehicle automation from Advanced Driver Assistance Systems (ADASs) to full Autonomous Vehicles (AVs) has the potential to completely revolutionize transportation systems by enhancing safety and accessibility, optimizing performance, and transforming mobility management. Currently, vehicles heavily rely on human drivers for operation and control, making them susceptible to human error, which is a significant factor in compromising traffic safety. This stems from the fact that human errors, such as distraction, fatigue, inattentiveness, or impaired driving contribute to a large proportion of accidents on the roads. According to the latest report by WHO [1], car crashes are responsible for the death of over 1.3 million people every year. In addition, as reported by Transport Canada [2], vehicle collisions caused more than 1700 fatalities in 2020 in Canada. Gradually moving toward higher levels of autonomy and deployment of AVs have the potential to mitigate the impact of human error through advanced sensors and artificial intelligence and enhance overall traffic safety.

In this context, the significance of Driving Behavior Analysis (DBA) and predictions extends across various levels of autonomy, ranging from the current landscape with advanced driving assistant systems to future scenarios where autonomy expands from specific driving conditions to fully automated vehicles operating without the intervention of human drivers. Driving behavior analysis emerges as a pivotal tool, capable of considerably transforming the development of ADASs, traffic safety, intelligent transportation systems, and advancements in autonomous driving. By modeling and characterizing driving behavior, DBA offers invaluable insights, addressing crucial tasks within ADAS such as driving style detection, identification of driver distraction or drowsiness, impaired driving detection, driver behavior modeling, driver identification and verification,

and driving behavior prediction. Furthermore, as the automotive landscape progresses towards fully automated vehicles, DBA takes on a critical role in training AVs. Analyzing real-world driving behaviors can cast light on human driving patterns and decision-making processes. Crucially, driving behavior analysis extends beyond the vehicle itself to encompass predicting the behavior of other road users, a vital component for the safe deployment of AVs. These roles of driving behavior analysis, both in understanding and predicting behaviors, establishes it as a foundation for improving road safety.

1.1 Challenges and Motivation

1.1.1 Behavior Analysis for ADAS

Advanced driver assistance systems require an enhanced understanding of driving behavior and human-machine interactions. This is not only necessary to prevent car crashes and ensure adequate driving performance and safety but also helps to better achieve personalization. This personalization is not limited to the infotainment perspective, such as adjusting the air conditioning, music, mirrors, and seat positions, and can also be beneficial for smoothing maneuvers according to the individual driving habits [3]. Therefore, for designing and integrating of these ADAS systems, it is essential to recognize which member of a family or a group is behind the wheel. Furthermore, a highly accurate driver verification and impostor detection system can be designed to warn business owners, such as insurance companies and ride-sharing companies as well as personal vehicle owners of any potential misuse or car theft risks. Early studies in driver identification [4–8] have made efforts to identify the individual behind the wheel using their physiological characteristics, such as sitting postures, finger-vein patterns, handgrip patterns on the steering wheel, or voice signals. However, these approaches suffer from several shortcomings, including: (1) their activation is dependent on the occurrence of particular events, such as when the driver starts to speak, (2) they are highly costly or intrusive, and (3) they do not consider continuous real-time identification.

Alternatively, in recent years, analysis of individuals driving behavior has drawn considerable attention for the development of advanced driver identification and verification systems. Analyzing and characterizing the driving behavior of the drivers has proved its effectiveness [9, 10] in a wide variety of research fields, such as intelligent vehicles, road safety, or fuel management, to mention a few. One way to design and deploy such a driver identification system is to map driving data to individual-specific characteristics through

data-driven models. Nonetheless, there are still significant challenges associated with behavior analysis for driver identification and verification, including data dependency and generalization issues. Each of these challenges is explained in the following.

- **Data dependency** Driver identification and verification systems heavily rely on data for their effectiveness and accuracy. The data dependency of these systems poses challenges from two key aspects. Firstly, it requires a significant amount of data to ensure accurate and reliable identification. This involves collecting and processing extensive datasets that capture a wide range of driving behaviors and patterns. Secondly, driver identification also requires a substantial amount of labeled data. The process of labeling data, which involves associating specific drivers with their respective behaviors or characteristics, can be time-consuming and resource-intensive. Obtaining a sufficient quantity of accurately labeled data is essential for training effective identification algorithms. Therefore, more cost-effective, accurate, and reliable models are needed to make practical and almost real-time applications possible.
- **Generalization** Another primary challenge of driver identification and verification systems is how to train a model to infer the identity of the drivers with high accuracy without the dependency on a large amount of driving data in a way that such a model generalizes over various scenarios and individuals. Achieving such generalization is particularly challenging for impostor detection task, where training data specific to the target impostor is not available. By addressing the generalization challenge, driver identification systems can enhance their accuracy and reliability, even when faced with limited or unavailable data for certain individuals or impostors.

1.1.2 Behavior Prediction for AVs

Autonomous vehicles offer the potential to enhance road safety, optimize traffic flow, and provide increased accessibility and convenience for passengers. For full adaptation to future use on public roads, AVs are required to guarantee the safety of all road users, including their passengers, while operating efficiently without disrupting traffic flow. In fact, deploying AVs is anticipated to improve traffic safety and decrease greenhouse gas emissions and the high number of car accidents [11]. Evidently, controlling a vehicle is a complex task that involves sophisticated scene understanding and making prompt

and crucial decisions accordingly. Notably, a human driver under normal conditions (i.e., being fully attentive and experienced) may have no difficulty developing situational awareness or predicting other drivers intentions, which appear to be of utmost significance for motion planning. Engaging in these tasks, however, poses a non-trivial challenge for an AV, demanding sophisticated capabilities in perception, navigation, decision-making, and real-time adaptation to dynamic environmental conditions.

Behavior prediction is of utmost importance in the automated driving field where AVs that share the road with other road users are expected to forecast the behavior of all Surrounding Agents (SAs) within a dynamic environment, thereby developing situational awareness. Such forecasting is essential not only for AVs' motion planning and policy generation but also for enhancing the effectiveness of transportation systems and traffic management as well as the safety and security of all road users, including passengers, drivers, and pedestrians [11]. Nevertheless, behavior prediction for AVs is highly challenging due to the necessity of incorporating factors, including social reasoning, scene conformability, and multimodality.

- **Social Reasoning.** There exists a correlation between the behavior of a Target Agent (TA) and that of all of its SAs, where they can affect each other's behavior. Drivers, cyclists, and pedestrians engage in cooperative behavior as they navigate through traffic, influenced by the presence and actions of other agents around them. As an example, a vehicle's velocity on freeways is constrained by the velocity of its leading vehicles or the feasibility of a lane-change maneuver is determined by the positions and velocities of vehicles in adjacent lanes. Social reasoning in this context refers to the capacity of the models to understand the social interactions among various agents within a driving environment. Therefore, more effective models are required for accurate social reasoning of behavior prediction models.
- **Scene conformability.** Stationary elements within the surrounding environment, such as lanes, crosswalks, sidewalks, structures, and parked vehicles, provide valuable information for understanding the goals and path preferences of SAs. Vehicles typically follow the centerlines of lanes and adhere to traffic rules. Pedestrians commonly walk along designated areas. Consequently, it is crucial for AVs to generate behavior forecasts that align with the surrounding environment and adhere to realistic driving behavior to achieve accurate trajectory predictions over long prediction horizons. As a result, there is a growing need for prediction models that can manage unfamiliar scene scenarios and adjust to dynamic situations.

- **Multimodality.** There is an underlying uncertainty in the future behavior of SAs. The pedestrians and the drivers of neighboring vehicles have a wide variety of potential future behaviors to select from. Thus, the distribution of future behavior of SAs is multimodal, representing different targets, paths, and behavior profiles along those paths. Multimodal behavior prediction is crucial for AVs to make safe and informed decisions since relying on a unimodal distribution for prediction can result in a loss of distinct modes, leading to inaccurate predictions. Generating one mode may result in an invalid motion since the model tries to find the average of various possible paths, which may not be a possible behavior for the target agent.

1.2 Objectives

The main objective of this thesis is to design novel models for behavior analysis and prediction specifically tailored for different levels of vehicle autonomy to ultimately improve driving experience and road safety. One primary focus will be on developing robust and accurate methods to identify individual drivers based on their unique driving behaviors, mainly enabling AVs to differentiate between authorized drivers and potential impostors. We aim to address the challenges associated with driver identification, such as data dependency and generalization, by developing innovative identification methods that mitigate these issues. The focus will be on designing algorithms that can effectively infer driver identities with high accuracy, even with limited driving data, and that demonstrate strong generalization capabilities across diverse scenarios and individuals.

Furthermore, this thesis aims to create advanced prediction models that accurately anticipate the future behavior and intentions of all surrounding agents of an AV, further enhancing the capabilities of AVs in terms of safety, performance, and proactive decision-making. We intend to design models that precisely predict future short-term behaviors of SAs by considering the dynamic interactions between vehicles, cyclists, pedestrians, and the surrounding environment. We intend to incorporate scene elements, such as lane graphs and location of obstacles, and extract local topological features to enhance the conformity of behavior predictions with real-world driving scenarios, enabling more realistic and context-aware results for both short-term and long-term forecasts. Additionally, we aim to design algorithms that can effectively capture and predict the diverse set of feasible behaviors exhibited by SAs in various driving situations. By incorporating multimodality, we provide more accurate and realistic behavior predictions.

1.3 Contributions

The main contributions of this thesis are listed below.

- **A comprehensive classification of DBA tasks and models.** This contribution provides a comprehensive categorization of state-of-the-art driving behavior works based on design and model architecture, the eventual aims of the analysis, and the types of data used for the analysis. We provide the fundamental characteristics of DBA, its existing challenges to form the basis for potential progress in this direction.
- **A performance evaluation of driver identification techniques.** We present a critical analysis of the principles considered in designing the driver identification models and assess their performance over real-world datasets. We shed light on each model’s capabilities and shortcomings.
- **DriverRep: driver identification through driving behavior embeddings.** This contribution addresses the necessity of a large amount of labeled data for driver identification models to guarantee their performance. We reduce the amount of data needed to accurately distinguish between drivers by presenting a fully unsupervised triplet loss training approach. DriverRep is the first driver identification model that selects the triplet samples and calculates drivers’ latent representations, called driver embeddings, from driving data in an innovative unsupervised manner.
- **A novel driver identification, verification, and imposter detection scheme.** This contribution addresses data dependency and the generalization ability of the driver identification systems. We reduce unnecessary and extensive data collection as well as the data dependency of the identification models by modeling the steering behavior of drivers. We also design a novel model to address driver verification as a comparison problem while having access to a short interval of driving data. Further, we eliminate the need for re-training the model for different types of impostors by proposing a novel imposter detection scheme.
- **Three novel approaches for driver intention and behavior prediction at unsignalized intersections.** Predicting drivers intention and vehicle behavior is substantially challenging at unsignalized intersections where there are no traffic lights to control the flow, or there exists no turning lane. This contribution consists of three novel approaches for this challenging scenario in which we address

anticipating driver intention as early as possible while the vehicle is approaching the T-junction, the interaction modeling among the agents, and the multimodality of the potential future paths of the vehicles at unsignalized intersections.

- **STAG: a novel interaction-aware behavior prediction method based on Spatio-Temporal Attention Graphs.** Full modeling of the social interactions between the agents within a driving scenario is a challenging task, especially for long-term anticipations due to complex interactions between the agents. Previous approaches consider interactions between all neighboring agents to be symmetric and model them as a dense undirected graph where it is assumed that the strength of agents' influence on one another is equal. This contribution addresses this issue by introducing a novel approach, namely STAG, where we avoid superfluous and fixed interactions, making the social reasoning process more informative for AVs.
- **CAPHA: a novel context-aware behavior prediction system of heterogeneous agents.** Evidently, the predicted behavior of the target agent and all of its surrounding agents must conform to the environmental circumstances and layouts, such as road structures and traffic rules. In other words, the anticipated behavior should be feasible to be performed in a real-world situation, which is a highly challenging task. In this contribution, we design CAPHA, a novel planning-based behavior prediction framework for heterogeneous agents with context awareness. CAPHA not only produces behaviors that conform to the environmental circumstances and are plausible for practical deployment but also addresses the multimodality and interpretability of the prediction problem.
- **GMP: a novel goal-based multimodal motion prediction.** In this contribution, we focus on both scene conformability and multimodality challenges of the behavior prediction problem. We design a novel goal-based motion prediction framework that anticipates the distribution of final TA goals on which their potential future behaviors are conditioned. We argue that estimating the endpoint distributions while taking account of lane and environmental interdependencies and considering them as behavior conditions can benefit the forecasting model to generate more diverse predictions that also conform to the scene and map layouts.

1.4 Outline

The remainder of this thesis is organized as follows. Chapter 2 provides a comprehensive overview of DBA models, data sources, primary tasks, and their fundamental definitions and challenges. Chapter 3 proposes an analytical framework for driver identification to investigate the capabilities of several machine learning and deep learning approaches. Chapter 4 introduces DriverRep, our novel driver identification model, which uses driver embeddings. Chapter 5 proposes a novel approach that models the steering behavior of the drivers for driver verification and impostor detection. Chapter 6 addresses behavior prediction at unsignalized intersections and presents one novel driver intention prediction and two innovative behavior prediction models. Chapter 7 designs an interaction-aware behavior prediction model called STAG, which avoids modeling superfluous social interactions among agents. Chapter 8 introduces a novel context-aware prediction approach, namely CAPHA, which is based on inverse reinforcement learning. Chapter 9 proposes a novel multimodal behavior prediction, called GMP, that predicts the distribution of agents final goals as a primary step before conducting behavior prediction. Chapter 10 summarizes the contributions of this thesis and discusses several potential research directions.

Chapter 2

Driving Behavior Analysis for Intelligent Transportation Systems

The advent of in-vehicle networking systems as well as state-of-the-art sensors and communication technologies have facilitated the collection of large volume and almost real-time data on vehicles and drivers, thus opening up future possibilities. Processing and analyzing this data provides unprecedented opportunities to offer remarkable insights and solutions for Driving Behavior Analysis (DBA). Characterizing driving behavior plays a key role in a variety of research areas such as traffic safety, the development of automated vehicles, energy and fuel management, risk assessment, and driver identification and profiling. Advances in DBA-based driver inattention or alcohol-impaired driver detection can help reduce fatal car crashes, and understanding the driving style (e.g. eco-friendly or aggressive) of drivers can contribute to fuel management and risk assessment of the drivers. These facts have led to a growing interest in addressing DBA challenges.

This chapter aims to present the state-of-the-art methodologies for DBA and provide a clear roadmap about the main current trends in DBA. To this end, we propose categorizing the current research on driving behavior based on the types of data employed for the analysis, the techniques based on which the driving data are modeled, and the ultimate goals of the analysis. We also provide an overview of different data resources and available datasets for various tasks of DBA. Moreover, we discuss the application of DBA along with the key research challenges in this field.

2.1 Introduction

Driving behavior is a complex concept that provides a description of the humans' decision while driving and how drivers control the vehicle in various conditions. DBA can be considered as characterizing the human driving behavior to fulfill a specific purpose such as identifying unsafe driving events, eco-friendly driving style recognition, authentication of drivers, risk assessments, etc. The possibility of characterizing and understanding driving behavior has been demonstrated since the early studies such as [12–14] in the field. In recent years, DBA has been receiving plenty of attention from both academia and industry due to its potential capabilities for cutting-edge automated driving innovations, intelligent transportation systems, ADASs, vehicle energy management, helpful insights for automotive manufacturers, environmental concerns, and traffic safety.

Many factors have motivated progress in the DBA research area. As reported by the World Health Organization [1], every year, more than one million people die as a result of fatal car accidents, and an average of 35 million more people suffer the injuries caused by these crashes. Human factors such as reading/sending text messages or emails while driving, talking on a handheld cellphone, drowsiness, and drunk-driving are among the major causes of road traffic crashes. For instance, inattentive driving claimed more than 2500 lives in 2018 in the United States alone [15]. DBA studies, particularly DBA for inattentive driving, alcohol-impaired driver detection, or the development of automated vehicles (gradually removing the human from the loop) are of utmost importance to tackle this issue. Furthermore, environmental concerns have recently gained a lot of interest. According to the International Energy Agency [16], road vehicles such as cars and buses are the cause of approximately three-quarters of transport CO₂ emissions. Understanding and promoting an eco-friendly driving style can definitely alleviate this problem. The above-mentioned factors are among the principal reasons why further steps must be taken by industrial and scientific communities toward DBA.

In this chapter, we provide a comprehensive survey of the fundamental characteristics of DBA and its existing challenges to form the basis for potential progress in this direction. To the best of the authors' knowledge, this work is the first one devoted to categorizing, and presenting a review of every aspect of DBA. More specifically, we propose classifying the current DBA studies based on three criteria including the most common data resources, the used models and algorithms in the literature, and the primary purpose of analyzing driving behavior. Figure 2.1 presents the introduced taxonomy. Utilizing these criteria, we believe that the introduced classification offers a thorough

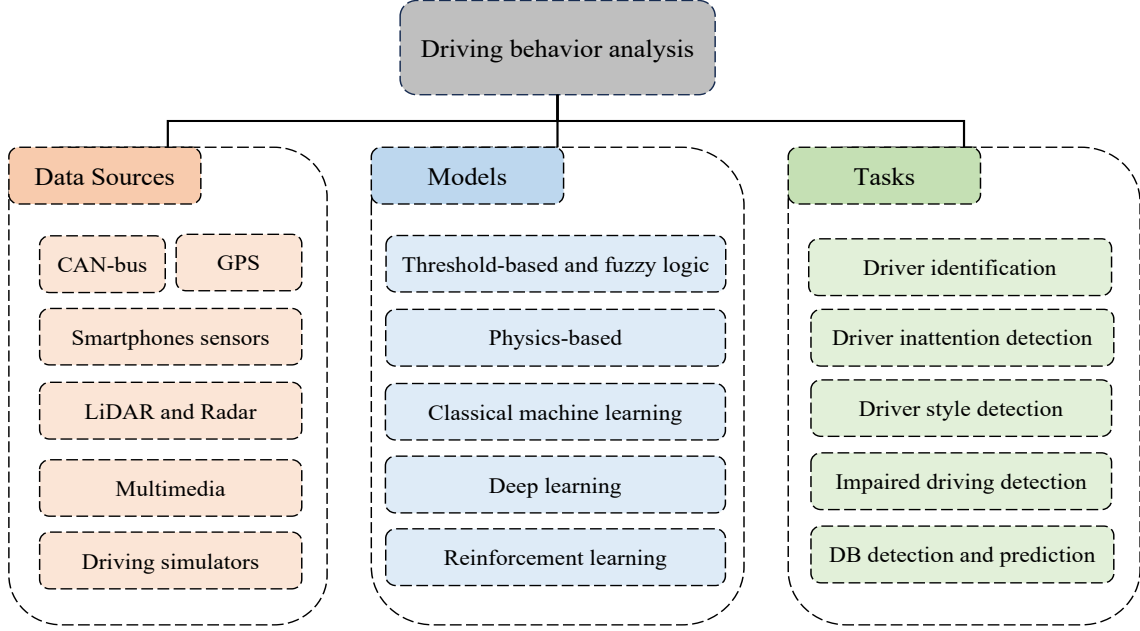


Figure 2.1: Taxonomy of driving behavior analysis studies

representation of methodologies designed for DBA. As illustrated, we classify DBA tasks into driver identification, driver inattention detection, driver style detection, impaired driving detection, and driving behavior detection and prediction.

2.2 Driving Behavior Data

In this section, we start by introducing the commonly used sources for collecting data in DBA, as elaborated in Section 2.2.1. Later, we proceed to discuss available datasets and offer a qualitative comparison in Section 2.2.2.

2.2.1 Data Sources

Studying driving behavior is not feasible without sufficient and reliable data from different data sources that can be used to obtain driving-related information from cars, drivers, and surrounding environments. In this part, we introduce the most commonly used data sources to collect driving-related information encountered in the DBA literature. Table 2.1 presents a summary of the introduced data resources.

Table 2.1: Commonly used data sources for DBA

Source	Data	Advantages	Disadvantages
CAN-bus data	Fuel related (e.g. fuel consumption, fuel trim), engine related (e.g. torque or temperature), transmission related, speed, steering wheel angle, pedals related, etc.	Low cost, efficient, easily captured and stored as structured data, real-time vehicle parameters	Need additional device (i.e., data logger) for data collection step, high-dimensional, not accurate information about the surrounding environment or other vehicles
GPS locations	Longitude, latitude, time, speed, and headings	Easily accessed for a large number of vehicles/drivers, real-time data collection, cost-effective	Accurate measurement of GPS (e.g. the environmental challenges such as urban canyon effect or underpasses) , may not contain enough information for some tasks of DBA
Smartphone sensors	Accelerometer, GPS, gyroscope, magnetometer	No additional equipment is needed, cost-effective, ubiquitous presence/accessibility, efficient	The orientation of the device, different kinds of smartphones, battery power
Multimedia data	Images and videos of a camera, audios from a microphone	Overcoming the limitations of GPS for some tasks such as lane detection , accessibility, providing information about surrounding environment	Mounting the smartphone on the dashboard/windshield of the car (for camera usage), noisy speech data, high illumination sensitivity, finding the correct spot of the smartphone placement (to effectively capture both audio and video/image)
LiDAR / Radar	Point clouds, 3D images / Radar scans	Accurate and precise distance, location, information about surrounding environment / Almost impervious to poor weather conditions and darkness, compact, cost-efficient, providing precise velocity measurements	Harder to incorporate, limitations in adverse weather conditions, not cost-efficient, interference / Less accurate than LiDAR in the size and shape of the obstacles, interference
Driving simulator	Can be a desirable data type/signal such as CAN-bus, images, etc	Flexible data collection for different types of scenarios with different traffic or weather conditions	The obtained results may defer when deploying in real-world situations because of the controlled environment

CAN-bus data. A commonly used source for performing DBA is CAN-bus [10, 17–21], which was developed and then officially released by Robert Bosch GmbH in the mid 1980s [22] to reduce the weight and complexity of the communication of the electronic control units in vehicles. Fugiglando *et al.* [9] investigated the potential benefits of the CAN-bus data for DBA in uncontrolled environments. More than 2000 CAN-bus signals were recorded among which 8 including brake and gas pedal, RPM, speed, steering wheel angle and momentum, and frontal and lateral acceleration were exploited in the feature extraction step of their exploratory analysis. Employing CAN-bus data, they delineated the common characteristics between the drivers through k-means clustering algorithm.

GPS locations. GPS-based technologies and applications are growing, and GPS sensors data as they are and as an indirect measurement of vehicle speed and acceleration can be beneficial for DBA. The fact that GPS sensors are embedded in smartphones and other systems, makes them suitable for DBA as they are easy to access and applicable to a large number of cars/drivers. Based on the literature review, there exist some DBA schemes with data dependency on GPS data [23–25]. In contrast to these studies, in a majority of works, GPS data have been used as a piece of complementary information to other types of data [26–28].

Smartphone sensors. Thanks to the ever-increasing availability of smartphones, both industry and academia have found a new source of data collection, which has turned out to be beneficial for numerous research fields among which DBA is no exception. Focusing on DBA research, several studies investigated the potential applications of smartphone data for DBA and, in general, for intelligent transportation systems. The ubiquitous presence of smartphones along with their broad range of embedded sensors such as accelerometer, gyroscope, proximity, and GPS resulted in the development in different aspect of DBA such as alcohol-impaired driving detection [29], driving event detection [30, 31], and driving style recognition [32]. For more detailed and comprehensive research on smartphone-based sensing for intelligent transportation systems, one may refer to the survey by Engelbrecht *et al.* [33].

Multimedia data. Not all types of sensors existing in a smartphone are beneficial for various applications of DBA. For instance, considering lane change detection, GPS-based positioning would not be perfectly adequate since an error of a few meters can change the desirable outcome of the lane positioning system [34, 35]. Consequently, a better alternative solution can be the utilization of the videos and images of the roads captured by cameras or drones. Precise detection of lane position is beneficial for several main purposes of DBA such as alcohol-impaired driving [36] or maneuver detection [35, 37].

Moreover, it should be noted that, in this review, multimedia data are mentioned as a source of DBA for their considerable potential in detecting driving behavior and not for their direct capabilities such as driver identification based on face recognition or drowsiness detection based on eye pupil as these methods are not dependent on driving behavior.

LiDAR and Radar. LiDAR and Radar appeared to be two main state-of-the-art technologies in the development of automated vehicles [38]. Concerning the working principles, both technologies measure the location and the distance of surrounding objects by sending signals and receiving the bounced-back ones from the objects with the only difference being that LiDAR sends the laser signals and Radar sends the radio waves. Considering DBA applications, the data obtained from these two sensors can be beneficial in tasks such as driver intention understanding and prediction [39] and turning behavior classification [40].

Driving simulator. Even though, nowadays, there are many types of sources from which researchers can acquire data for DBA, there exist some works in the literature that have used driving simulators for data collection [41–44]. The motivation for utilizing simulators can be the fact that in some situations, there are no available datasets to be used, or collecting the real driving data is quite time-consuming. Furthermore, for some tasks such as drowsiness [45,46] or alcohol-impaired driving recognition [47–49], it seems more appropriate to either use driving simulators or collect data in a very controlled environment. The main advantage of employing the simulators is their flexibility in providing a wide range of driving scenarios. However, the fact remains that the simulators' capabilities are limited in simulating all aspects of real driving experience [50].

Due to the fact that there are no sensors without inherent weaknesses and each type of sensor has its own merits and demerits [51], some studies such as [52–54] have taken advantage of sensor fusion, i.e., combining the data obtained from different types of sensors. On the one hand, sensor fusion can be an extremely challenging task due to a variety of data types within the sensors including structured, semi-structured, and unstructured data. Understanding and merging related data from these diverse types need extra intuition and effort. Moreover, integrating data from many sources results in a huge amount of data that might reach even petabytes of data [51]. Handling this high volume of data requires modern data cleaning, pre-processing, and storage techniques. On the other hand, the combination of data from many sources leads to having much more information about the vehicles, drivers, and the surrounding environment such as weather conditions, real-time traffic, and biomedical signals (using wearable devices)

from the drivers. This additional information may yield data enrichment, higher data accuracy, and consequently better results for specific tasks of DBA [55–57].

2.2.2 Datasets

Datasets play a pivotal role in advancing the field of DBA, providing the essential raw material upon which models and algorithms are built and evaluated. In this context, a variety of datasets are available for performing DBA. In this part, we summarize DBA datasets that are either publicly available or available on demand along with their main features and how they can be used for different purposes. We will discuss these datasets for behavior analysis and prediction, separately.

Datasets for Behavior Analysis

Table 2.2 summarizes the main characteristics of DBA datasets for driving behavior analysis. Based on our findings, we discuss the strengths and limitations of the reviewed datasets.

Several datasets such as [58–61] collected data from various available data sources to incorporate more information about real-time vehicle parameters, drivers, and environment, which has been demonstrated to contribute to overcoming the possible malfunctions [51,62,63] and to take advantage of the communication flexibility of others such as smartphones, cameras, and GPS. Honda Research Institute Driving Dataset (**HDD**) [59] was proposed to make research on learning driver behavior feasible. The data is 104 video hours and is collected in San Francisco Bay area. Six sensors including vehicle speed, accelerator, brake, steering angle, rotation speed, and yaw rate are provided from CAN-bus data. **Comma.ai** [61] released a part of their dataset for research purposes. The dataset contains data obtained from various resources such as cameras, GPS, and CAN-bus. **Brain4cars** dataset [60] is a publicly available dataset that contains data from cameras, GPS, and street maps. **HCIIlab** [58] released their dataset to be publicly available for the research community. Three different types of data resources including physiological sensors, smartphones (GPS and accelerometer), and two cameras were used. Ten drivers were asked to drive in 5 road types including 30/50 km/h zones, highway, freeway, and tunnel.

Ideally, the real-world data represents the actual behavior exhibited by the drivers. However, obtaining real-world data poses several challenges and dangers for some tasks of DBA such as drowsiness or alcohol-impaired detection where data collection in real

Table 2.2: A summary of available datasets for driving behavior analysis

Dataset	Resources	Volume	Length	Primary Purpose	Road Type
HDD [59]	Camera, LiDAR, GPS, CAN-bus	150 GB	104 hours	Event detection	Urban, suburban, highway
Comma.ai [61]	Camera, GPS	80 GB	7.5 hours	Driving simulation	Highway
HCIIlab [58]	GPS, multimedia, wearables	450 MB	300 minutes	Driving workload analysis	Highway, freeway, tunnel
Multimodal [64]	Driving simulator, multimedia, wearables	57.5 GB	10.9 km	Distraction detection	Highway
SHRP2 [65]	Camera, GPS, CAN-bus, radar, smartphones	+700 TB	+45,000 hours	Driver profiling, accident prevention, risk assessment	Highway, freeway, parking lot
100-car [66]	Multimedia, GPS, CAN-bus, radar, smartphones	6.4 TB	+43,000 hours	Accident prevention	Urban, rural
SPMD [67]	CAN-bus, multimedia, radar, RSE, V2V devices	N/A	1.5 year	Road safety	Urban
400-car [68]	Multimedia, GPS, CAN-bus, radar	60 TB	6 month	Road safety	Urban, rural
UYANIK [69]	Multimedia, CAN-bus, GPS	9 TB	+25 km	Style recognition	City way, highway
UAH-Driveset [70]	Smartphone	N/A	500 minutes	Style recognition	Motorway, secondary road
HCRL [71]	CAN-bus	17.7 MB	23 hours	Driver identification	City way, motorway, parking

scenarios is not possible or can cause devastating consequences. The **multimodal** dataset [64] is a freely accessible dataset that is collected employing a driving simulator. It includes data from more than 60 drivers while driving under four different conditions based on the amount of distraction including none, cognitive, emotional, and sensorimotor distractions. The dataset contains various features such as speed, acceleration, steering, brake, heart rate, facial expressions, eye-tracking data, etc.

In contrast to the datasets that are limited in terms of the number of drivers/vehicles under study [58, 60, 70, 71], there exist some large naturalistic driving studies around the world resulting in big datasets [65–68]. The second Strategic Highway Research Program (**SHRP**) [65] created a naturalistic driving dataset which is the largest one to date. The raw data in the dataset contain more than 29000 driving events. The data are also annotated for driving events including crashes, near crashes, and a baseline that can be employed for accident prevention/prediction and risk analysis scenarios. The **100-Car** Naturalistic Driving Study [66] was the first naturalistic driving study to collect large-scale data from equipped vehicles. Similar to SHRP, this dataset is also annotated for driving events such as crashes, near crashes, maneuvers, and pre-crashes events. Safety pilot model deployment data (**SPMD**) [67] is a collection of datasets that contains data related to driving, network, basic safety messages, roadside units (RSE), and weather. Various types of data are collected from passenger vehicles, trucks, and means of public transportation all of which are equipped with V2V communication devices. A limitation is that this dataset does not include traffic conditions of the roads, which may affect DBA [72]. The Australian **400-car** Naturalistic Driving Study [68] collected driving data from 400 drivers while driving their vehicles in normal, impaired, and safety-critical situations. Different data sources such as CAN-bus, cameras, microphones, GPS, and radar have been used to obtain data from the vehicles.

Among the datasets in which the drivers' gender and age information is available [58, 64, 66, 68–70], most of them are limited with respect to the gender or age distribution of the drivers [58, 66, 69, 70] which may lead to biased conclusions since it has been shown [73] these two factors have effects on the driving style of the individuals. In a large dataset of driving behavior collected using **UYANIK** [69], the participants include 38 males ranging from 22 to 61 years old and 15 females ranging from 21 to 48 years old. **UAH-DriveSet** [70] dataset contains information about 6 different drivers 5 of whom are males ranging from 20 to 50 and one of whom is a female between 30 and 40 years old. The participants' gender in [66] consists of approximately 60% males and 40% females. HCIIlab dataset [58] includes 10 drivers including 7 males and 3 females with an average age of 35 years old. On the contrary, the multimodel dataset [64] is balanced in terms of gender (i.e., 33 females and 35 males) and age (34 young and 34 old participants) of the drivers. Also in [68], the participants are equally recruited from males and females.

Furthermore, many datasets such as [59, 69, 71] have taken account of the road type in their data collection. It has been shown in several studies such as [74, 75] that the road type has an impact on the driving behavior of the drivers, thus making it an

important factor to consider. Nevertheless, the road type is not the only environmental factor influencing driver behavior and other factors such as weather could be critical for DBA [76], which raises the need for more comprehensive datasets.

The primary purposes of the reviewed datasets are presented in Table 2.2. However, it should be noted that their purposes are not limited to what has been mentioned in Table 2.2. For instance, although among the reviewed datasets, only **HCRL** [71] is initially introduced for driver identification task, other datasets such as [58, 69, 70] can also be employed for such a purpose. This is due to the fact that, for driver identification, it is simply required to know which samples, e.g. sequences in the time series, belong to the same driver identified by a number, a file, or a letter.

Datasets for Behavior Prediction

We provide a summary of available datasets specifically for driving behavior prediction in Table 2.3. Different datasets are available to researchers for behavior prediction, each offering distinct advantages and limitations. **Argoverse** datasets [77, 80], for instance, provides high-resolution data collected from Argoverse AVs, offering a diverse set of scenarios and interactions between vehicles. Their advantage lies in their real-world nature, allowing the study of complex urban driving situations for both short-term and long-term predictions. Next Generation Simulation (**NGSIM**) dataset [88, 89], on the other hand, comprises real-world traffic data collected from highway surveillance cameras, offering insights into highway driving behaviors. It is particularly useful for studying highway scenarios and driver behaviors at high speeds. However, it may lack the diversity of urban traffic scenarios which are later covered by datasets such as **KITTI** [87] or **HighD** [85] which provides real-world data from German highways, allowing researchers to study high-density traffic situations. This dataset focuses on dense traffic scenarios, which are common in urban areas during peak hours. However, it may not capture the complexities of urban interactions as seen in datasets such as Argoverse, **interaction** [82], or **inD** datasets. Intersection Drone (**inD**) dataset [78] records drone-captured trajectories over four intersections in Germany, providing valuable insights into heterogeneous agents' behaviors at intersections. This dataset concentrates on intersection scenarios, which are critical for predicting agent behaviors in complex urban driving situations. Furthermore, trajectory prediction at unsignalized intersections is also a critical aspect of traffic management and autonomous driving systems. Unsignalized intersections lack traffic signals or control mechanisms, leading to complex and dynamic interactions among vehicles as they navigate through these junctions. **ACFR** [81] dataset provides real-world trajec-

Table 2.3: Available datasets for driving behavior detection and prediction

Dataset	Year	Resources	Length	Road type	Data type	Agent type
Argoverse 2 [77]	2021	LiDAR, camera	250,000 scenarios	Urban	trajectories, HD maps	Vehicles
InD [78]	2020	Drones	13599 trajectories	Intersections	Trajectories, scene images	Vehicles, pedestrians, cyclists
Waymo open dataset [79]	2020	LiDAR, camera	103354 segments	Urban	trajectories, HD maps	Vehicles, pedestrians, cyclists
Argoverse 1 [80]	2019	LiDAR, camera	324,557 scenarios	Urban	trajectories, HD maps	Vehicles
ACFR [81]	2019	LiDAR, camera	23000 trajectories	Unsignalized intersections	trajectories	Vehicles
Interaction [82]	2019	Drones	40054 vehicles	Intersections, highway	trajectories, HD maps	Vehicles, pedestrians
ApolloScape [83]	2019	LiDAR, camera, GPS	143,906 video frames	Urban	Trajectories	Vehicles, pedestrians, cyclists
NuScenes [84]	2019	LiDAR, camera	1000 scenarios	Urban	trajectories, HD maps	Vehicles, pedestrians, cyclists
HighD [85]	2018	Drones	110500 vehicles	Highway	trajectories, lanes	Vehicles
SDD [86]	2016	Drones	19000 agents	Stanford campus	Trajectories, scene images	Pedestrians, vehicles, cyclists
Brain4cars [60]	2015	Camera, GPS, maps	1180 miles	Highway, city way	trajectories, head motions	Vehicles
KITTI [87]	2012	LiDAR, GPS camera	11520 trajectories	Rural, high-way	Trajectories, point clouds, images	Pedestrians, cyclists
NGSIM [88]	2006	Camera	90 minutes	Highway	Trajectories, lanes	Vehicles

tories for more than 23000 vehicles collected at five different unsignalized intersections. Moreover, most reviewed datasets [78, 79, 82, 83, 86, 87] contain information on different types of agents including vehicles, pedestrians, and cyclist which cast light on behavior predictions for heterogeneous agents. While each dataset contributes valuable information to the development and evaluation of trajectory prediction models, a combination of datasets may be necessary to achieve comprehensive insights and robust predictions in diverse driving scenarios.

Furthermore, most recent datasets [77, 79, 80, 84] provide High-Definition Maps (HD maps) in addition to agents' trajectories since these maps play a critical role in the advancement of autonomous driving technology. Unlike traditional navigation maps, HD maps provide highly detailed, precise, and up-to-date information about road geometries, lane markings, traffic signs, and other static features of the environment. These maps are essential for AVs to navigate safely and efficiently, as they enable the AVs to understand their surroundings accurately and plan their trajectories accordingly. HD maps serve as a crucial reference for AVs' perception and localization systems, allowing them to align sensor data with the map to determine their precise position on the road. Moreover, these maps facilitate advanced decision-making by providing additional contextual information, such as traffic rules, speed limits, and complex intersections. However, creating and maintaining HD maps present significant challenges, as they require continuous updates to account for road changes, construction work, and temporary traffic conditions. As a result, some datasets [78, 86] use Bird's Eye View (BEV) images instead.

2.3 Driving Behavior Models

DBA approaches in the literature can be classified based on not only their used source of data but also the algorithms and models based on which driving behavior is characterized. In this section, we review some of the main and implemented algorithms appropriate for different tasks of DBA.

2.3.1 Threshold-based and Fuzzy Logic-based Methods

Threshold-based and fuzzy logic algorithms perform DBA tasks based on presenting several forms of knowledge about a specific problem along with modeling the correlation existing between different parameters/indicators using fixed thresholds. Rules can be particularized to specific driving behavior or events in the form of IF-THEN statements. Although rule-based algorithms are easy to implement, their simplicity and interpretability are limited by the number of parameters. Fuzzy logic is also an effective tool to capture the potential uncertainty within driving data employing fuzzy sets that make it possible to imitate human reasoning.

The algorithms in this category are mostly used for driving style detection [90–94]. Bejani and Ghatee [90] designed a rule-based fuzzy system to aggregate the results of the initial driving style classification with the data from the driver's profile. This system

consists of two modules. The first one is responsible for assessing the current driving maneuver and outputs an initial driving style, while the second one makes a decision based on the output of the first module and the ratio of the number of risky maneuvers to the number of safe maneuvers in driver's profile. Magaña & Organero [92] proposed a fuzzy logic system to evaluate whether or not drivers are following an energy-efficient manner with respect to fuel consumption. Different types of driving data including speed, acceleration, and engine load are used as an input for this system. The output is a score ranging from 0 to 10. The higher the score is, the more eco-friendly the driver behaves. The proposed system is used to not only rate the drivers and provide efficient recommendations but also recognize the areas in which drivers are driving inefficiently. Dörr *et al.* [93] benefited from fuzzy logic to calculate a driving style coefficient based on which drivers are categorized into 3 groups for exhibiting normal, comfortable, and sporty styles. Several features extracted from CAN-bus including acceleration, deceleration, and speed along with road types are fed to a fuzzy system for classification.

Fuzzy systems have been used for other tasks of DBA as well. Aksjonov *et al.* [95] used fuzzy logic to evaluate the level of driver distraction. IF-THEN statements are employed to make a decision based on two parameters including lane offset of the vehicle and speed limit maintenance of the drivers. For instance, the results indicated that IF the lane offset of the vehicle is "out" of the road bounds and the difference between the vehicle's speed and the road speed limit is "bad", THEN driver distraction is "high". Nilsson *et al.* [96] introduced a rule-based system to predict driving maneuvers, i.e., left/right lane changing. Both lateral and longitudinal parameters such as lateral velocity and lane offset as well as time-to-collision and time-gap are used to generate the rules. Although threshold-based and fuzzy logic algorithms are simple to implement and may produce acceptable results, their performance is dependent to a great extent on the choice of the thresholds. In other words, the approaches that use these algorithms are limited to pre-defined thresholds for some driving behavior indicators. These approaches are also limited in terms of the number of parameters. The higher the number of parameters, the more accurate the approaches. However, this comes with inefficiency as an increase in the number of parameters will lead to a substantial increase in the number of rules.

2.3.2 Physics-based Methods

Physics-based approaches are mostly used for behavior prediction task. They consist of dynamics and kinematics models. While dynamics models can become extremely

complex and contain numerous parameters, they often are less efficient and impose additional computational cost. As a result, simpler dynamics models are generally employed for behavior forecasting. In this task, vehicles are commonly represented using a bicycle model, which simulates their motion driven by the front wheels. On the other hand, kinematics models, known for their simple structure, are more frequently employed. Kinematics models mostly include the following: (1) Constant-Velocity (CV), (2) Constant-Acceleration (CA), (3) Constant Turn Rate and Velocity (CTRV), (4) Constant Turn Rate and Acceleration (CTRA), (5) Constant Steering Angle and Velocity (CSAV), (5) Constant Steering Angle and Acceleration (CSAA). The methods using these models can be categorized into three group. **Single behavior methods** involves directly calculating the future states of the agents by applying kinematic models on agents' current states [97, 98]. These methods offer low computational cost, making them suitable for less constrained applications. However, a limitation is their inability to consider scene-related factors and multimodality.

While single behavior methods assume that the states of agents do not encounter any noise or uncertainty, the **Kalman Filtering (KF) methods** are designed to deal with such noises by modeling it in the agents' current state employing a Gaussian distribution [99]. These methods incorporate prediction and update steps in a loop, allowing the estimation of the mean value and covariance matrix of the agent state for each future time step. One advantage of these methods is their consideration of the uncertainty in the forecasted behavior. However, using a unimodal Gaussian distribution may not be sufficient to represent various behaviors. To address this limitation, Kaempchen *et al.* [99] propose the Interacting Multiple Model IMM approach for multimodal behavior prediction. In another approach, the switched Kalman filter [100], the authors employ a set of KFs to represent different physical modes of the agents and switch between them. **Monte Carlo methods** are often employed to approximate the state distribution. These methods involves randomly sampling the input and adapting the kinematic models to produce potential future behaviors [101, 102]. To ensure the plausibility of the predicted behavior, the produced behaviors are filtered based on desired factors, such as maximum lateral acceleration allowed.

Physics-based methods in general are behavior prediction approaches that leverage dynamic and kinematic models to achieve accurate results while keeping computational resources relatively low. While these methods may have lower accuracy compared to other techniques, many newer models incorporate the principles of physics-based models to enhance their predictive capabilities. Physics-based methods tend to perform better

when the movement of agents can be effectively captured by kinematics or dynamics models. However, since the physical characteristics of traffic participants constantly change, most of these methods are more appropriate for short-term prediction.

2.3.3 Classical Machine Learning-based Methods

Machine learning algorithms cover a major part of driving behavior models. Recent algorithms employed for DBA can be categorized based on their functionality. As a powerful group of techniques, **ensemble algorithms** combine the predictions of multiple base models to create a more accurate and robust prediction. They often outperform individual models by reducing overfitting and leveraging the strengths of different algorithms. Random Forest (RF) as a well-established ensemble method, builds multiple decision trees during training and averages their predictions to improve accuracy and control overfitting. Each tree is trained on a random subset of the data with replacement, and feature selection is performed at each node to enhance diversity. RF has been commonly used to carry out main tasks of DBA including detecting driving events [34], alcohol-impaired drivers [50], driving styles [103], drivers inattention [46,104], and drivers themselves [25,63,105,106]. Although a majority of studies in the literature took advantage of Random Forest, there exist several works based on other ensemble algorithms such as Gradient Boosting (GB) [34,54], Adaboost [50], bagging [30], and an ensemble of several classical machine learning algorithms [45,107].

Bayesian algorithms work based on Bayes theorem for probability computation. Bayesian networks, a type of probabilistic graphical model, represent conditional dependence between random variables by edges in a graph and are suitable for predicting the probability of different classes/events. They are used to make probabilistic predictions and decisions. Several Bayesian algorithms used for different tasks of DBA can be summarized as: (1) Naive Bayes (NB) which is a simple probabilistic classifier based on Bayes' theorem and the assumption of conditional independence between features, (2) Bayesian Networks which model probabilistic relationships among a set of variables using a directed graph. Conditional probability distributions are assigned to nodes, allowing for efficient probabilistic reasoning and inference. (3) Hidden Markov Model (HMM) which is a statistical model used to describe sequences of observable events that are governed by an underlying hidden process, (4) Gaussian Mixture Model (GMM) that is a probabilistic model that represents data as a mixture of several Gaussian distributions. GMM is commonly used for density estimation tasks. Bayesian algorithms have

been actively used for DBA purposes, including, but not limited to, drivers stress detection [108], maneuver detection [37], recognition of driving styles [41, 57, 109], distraction detection [110], lane-change detection [111]. Bayesian algorithms are particularly useful when dealing with uncertainty, limited data, or cases where prior knowledge can provide valuable insights into predictions.

Instance-based algorithms, also called memory-based algorithms, make predictions based on the similarity between new data instances and the training instances. These algorithms store the training instances and use them to make predictions for new data points. In this category, K-Nearest Neighbors (KNN) is commonly used in the DBA research field [17, 106, 111]. KNN predicts the class of a new data point by considering the majority class among its k nearest neighbors in the training data. It measures similarity using different distance metrics like Euclidean distance. Also, Learning Vector Quantization (LVQ) is a prototype-based algorithm that adjusts prototype vectors in response to the training data. It classifies new instances based on their similarity to these prototypes. LVQ has also been used for DBA tasks, such as driver identification [105]. Furthermore, while Support Vector Machine (SVM) itself is not strictly an instance-based algorithm like KNN, it can be considered somewhat similar in spirit because it identifies support vectors, which are a subset of training instances that significantly influence the construction of the decision boundary. SVM has been extensively implemented for various tasks of DBA such as recognizing different driving events [34], alcohol-impaired drivers [36, 48, 50], driving styles, [43], drivers distraction [18], and the person behind the wheel [63, 105, 112]. Central to the SVM's efficacy is its adeptness at identifying support vectors that align with the classification criteria. The pivotal task is then to determine the optimal hyperplane that maximizes the margin between classified data points. SVM is also a robust tool capable of acquiring and discerning a driver intention within complex environments [113–115]. In this context, SVM categorizes driving maneuvers into distinct classes like turning left, turning right, and maintaining a straight path. Leveraging a kernel function, the input data is transformed into a high-dimensional space, facilitating linear classification that identifies driving maneuvers and, subsequently, aids in vehicle behavior prediction.

Regression algorithms are concerned with predictive modeling of the correlation between variables. Regression algorithms investigate the causal relationship between the targets and predictors, which makes them suitable for time series classification. These algorithms are used for detecting alcohol-impaired drivers [47], classifying driving styles [109], or measuring the drowsiness level of drivers [116]. **Decision tree algo-**

Decision Trees (DT) construct a tree based on finding the best split points among the features using impurity measures. They have been used for DBA as well [107,117] since they are easy to implement. However, they are limited in terms of the number of features used.

Classical machine learning techniques have been widely employed for DBA, providing valuable insights into driver intentions, and patterns. These techniques often involve extracting relevant features from input data and using various algorithms to model relationships and make predictions. These methods have shown effectiveness in scenarios like driver intention prediction, driving style detection, distraction detection, and anomaly detection. However, they often rely on handcrafted features and struggle to capture complex patterns and dependencies present in large-scale, high-dimensional data.

2.3.4 Deep Learning-based methods

As discussed in the previous section, classical machine learning methods often struggle with the high-dimensional and non-linear nature of driving data, making it challenging to capture subtle patterns in the driver or vehicle behavior. Therefore, Deep Neural Networks (DNN), as a subset of machine learning models, have gained popularity in driving behavior analysis due to their ability to automatically learn complex patterns and features from raw data. Neural networks, the foundation of deep learning, consist of layers of interconnected nodes that process data hierarchically, enabling them to extract meaningful representations automatically. Similar to other machine learning models, there are several types of deep learning-based approaches used for driving behavior analysis, each tailored to address specific challenges and tasks within the field.

Convolutional Neural Networks (CNNs) are a major deep learning tool for processing spatial information and learn a representation of the feature space hidden in data. They excel at tasks such as face recognition, lane detection, and scene understanding. CNNs learn hierarchical features by applying convolutional filters to input images, enabling them to capture spatial patterns effectively. Although CNNs are particularly used for computer vision purposes [118,119], they have shown efficient results for time series classification, especially for driving style detection [120] and driver identification [20,21,121,122]. Bejani and Ghatee [120] proposed to employ adaptive regularization and dropout to address the overfitting problem of CNNs for driving style detection. Jeong *et al.* [122] used CNN to investigate the effect of extracting special segments of driving signals including acceleration and deceleration sections on the accuracy of the driver identification systems. The authors suggested detecting these particular sections through the

pressure on both pedals and the speed of the car. The results demonstrated that this method could moderately improve the accuracy from almost 88% to almost 90%. This is due to the fact that the extracted sections can reflect unique driving characteristics of the individuals. Moreover, Fully Convolutional Networks (FCN) are deep learning architectures designed for tasks that involve structured data. FCNs employ convolutional layers, which are well-suited for retaining spatial or temporal information while effectively learning hierarchical features. In DBA, FCNs have been employed [62, 123] to process sequential driving data, such as trajectories or sensor measurements. Furthermore, Temporal Convolutional Networks (TCNs) are designed to capture temporal dependencies in sequential data efficiently. They can be used for DBA task like behavior prediction, where the model learns to predict future vehicle positions based on historical trajectory data.

Recurrent Neural Networks (RNNs), as another important type of neural networks, are well-suited for sequential data analysis, making them ideal for tasks involving time-series data. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) variants of RNNs help capture long-term dependencies and temporal dynamics in driving behavior. They have been exploited in various tasks of DBA including maneuver prediction [35, 39], driver identification [21, 62], driving style detection [124] or driver distraction detection [125]. For instance, Würtz and Göhner [124] used LSTM to automatically learn features from GPS data for driving style detection.

Multilayer Perceptron (MLP) is a foundational deep learning architecture commonly used in DBA tasks. It consists of multiple layers of interconnected nodes, each performing weighted summations followed by activation functions. MLPs excel at learning complex patterns and relationships within driving data, making them valuable tools for tasks such as behavior prediction [126, 127] and driver identification [107, 128]. Their ability to model nonlinear dependencies in data allows them to capture driving behavior patterns, enhancing the capabilities of AVs.

Graph Neural Networks (GNNs) are a class of deep learning models designed to process data structured as graphs, making them particularly suitable for tasks involving complex interactions and relationships [129]. In other words, GNNs are designed to operate on graph-structured data, where nodes represent entities (e.g., vehicles, pedestrians) and edges represent relationships between these entities. GNNs iteratively update node representations by aggregating information from neighboring nodes, capturing both local and global context. In the context of DBA, different types of GNNs, such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), offer powerful

approaches for understanding the interactions between vehicles and their surroundings in various scenarios. As a result, in recent years, there has been a growing interest for using GNNs within DBA tasks such as behavior prediction [130–132]. **Attention mechanisms** are key components of modern deep learning models that can enhance their ability to focus on specific parts of input data, enabling more effective processing and improved performance in various tasks. In the context of DBA, attention mechanisms, such as Multi-Head Attention (MHA), play a can improve performance by capturing relevant information from complex scenes and sequences of data [133].

Generative Adversarial Networks (GANs) are a class of deep learning that consist of two neural networks, the generator and the discriminator, which are trained in a competitive manner to produce high-quality synthetic data [118]. In the context of DBA, GANs can be used for generating realistic driving scenarios or augmenting datasets. By capturing the underlying distribution of real-world driving scenarios [134], GANs contribute to more robust, comprehensive, and adaptable models for autonomous vehicles and driver assistance systems. **Variational Autoencoders (VAEs)** are another class of generative models that combine elements of both autoencoders and probabilistic graphical models. The key innovation of VAEs lies in introducing a probabilistic interpretation to the latent space [135]. Instead of a deterministic latent vector, VAEs learn a distribution from which latent variables are sampled. This enables VAEs to capture the inherent uncertainty in the data. This characteristic makes VAEs particularly suitable for various DBA tasks such as unusual events detection [136], alcohol-impaired driver detection [137], and behavior prediction [138, 139], where inherent uncertainty in human behavior needs to be effectively captured.

2.3.5 Reinforcement Learning-based Methods

Reinforcement Learning (RL) is a machine learning paradigm that involves an agent learning to make sequential decisions by interacting with an environment to maximize a cumulative reward [140]. RL-based methods have been applied to various aspects of AVs, particularly in decision-making scenarios. These algorithms learn optimal policies through trial and error, where the agent takes actions in the environment and receives feedback in the form of rewards. In the context of driving, RL methods, which mostly take advantage of Markov Decision Process (MDP) [141], can learn to navigate through complex scenarios, make lane changes, follow traffic rules, and perform other maneuvers. RL algorithms typically require a predefined reward function, which can be challeng-

ing to specify accurately for complex tasks like driving [142], where human behavior is complicated and influenced by various factors. In such cases, **Inverse Reinforcement Learning (IRL)** has gained attention as a promising solution. IRL addresses the issue of reward function specification by aiming to learn the reward function from observed behavior rather than assuming it beforehand. By analyzing human-driven behaviors and actions, IRL attempts to reverse engineer the underlying motivations and preferences that guide human behavior. This approach is particularly valuable in DBA, as it enables automated systems to understand and imitate the decision-making processes of human drivers. the inherent nature of Inverse Reinforcement Learning (IRL) makes it particularly well-suited for behavior prediction tasks [143, 144], where the goal is to anticipate how agents, such as drivers, will act in various situations.

2.4 Driving Behavior Tasks

DBA has the potential to mitigate harmful effects of car accidents, CO₂ emissions or dangerous individual driving habits [19, 76], and consequently, to enhance the performance of ADASs, the development of automated vehicles, traffic safety, and energy management [145, 146]. Unlocking this potential has been among the main motivations for DBA research, and there exist various types of DBA tasks each of which tries to serve a different purpose. In this section, we categorize the reviewed studies according to their main task/purpose. Table 2.4 summarizes the main tasks of DBA, their challenges as well as their potential applications. A more comprehensive description of each task is provided in the subsequent subsections.

2.4.1 Driver Identification

Driving behavior analysis can be beneficial for *driver identification*, also called *driver fingerprinting* [17], which is the task of identifying the individual behind the wheel. In this category, the possibility of uniquely recognizing the driver by analyzing their driving behavior has been investigated. Identifying the driver among a set of drivers can be performed based on some information collected about something she/he has, is, or has the knowledge of [63]. Different methods can be employed to identify the drivers. The early studies on driver identification benefited from sitting postures [4], handgrip on steering wheel [5], or finger-vein patterns analysis [7]. Others used fingerprints, iris technologies, RFID cards, camera recorders, and voice/face recognition [6, 8]. These

Table 2.4: Classification of the main tasks/purposes of DBA

Task/Purpose	Main Idea	Challenges	Potential Applications
Driver Identification	Identifying the person behind the wheel using characterizing the unique driving behavior	Costly data collection and data dependency, accuracy and scalability trade-off for practical applications, generalization, privacy of the drivers	Vehicle safety, fleet management, personalized in-vehicle infotainment, driver authentication, imposter detection, insurance telematics
Driver Inattention Detection	Analyzing the way drivers perform the driving task to detect signs of distraction/drowsiness such as slower reaction time, sudden steering wheel movements or changes in lane positions	Being adaptive to various driving styles of the same/different drivers, detecting cognitive distractions, the necessity for quick detection and warning	Collision warning, road and driving safety improvement, car accident prevention
Driving Style Detection	Leveraging DBA to point out determinant of for instance aggressive or eco-friendly driving such as harsh acceleration/deceleration or sudden maneuvers	Having large amount of driving data, labeling data manually, driver behavior uncertainty	Energy management, fuel management, curve safety speed modeling, insurance telematics
Alcohol-impaired Driving Detection	Employing DBA to spot the signs for alcohol-impaired driving behavior including delays in response time, high number of variations in acceleration or speed	Distinguishing different levels of BAC, different traffic and road conditions as well as time of the day, collecting naturalistic dataset	Road safety, preventing fatal traffic crashes
Driving Behavior Detection and prediction	Leveraging the behavior history of target agents and their surrounding environment to predict their intentions/maneuvers (i.e., turns, lane changes, sudden movements, etc.) or motions	Detecting human intentions such as performing maneuvers in a limited time before the actual occurrence of the maneuver, social reasoning, detecting corner cases and in-between maneuvers, scene conformability, multimodality	Developing automated vehicles and robust ADAS, driving style detection, driver profiling, trajectory prediction, traffic and road safety,

methods are considered highly intrusive as they require highly sensitive information from the drivers and their activation is dependent on the occurrence of particular events, such as when the driver starts to speak. In this section, however, we focus on non-intrusive studies that have made efforts to identify the drivers based on their driving behavior rather than their physical or physiological features which are beyond the scope of this chapter. In the following, we survey DBA-based driver identification and verification methods. Table 2.5 provides a summary of discussed methods. Furthermore, we provide a qualitative feature comparison between the discussed driver identification and verification methods in Table 2.6.

Non-intrusive Driver Identification

Considerable efforts have been made with the aim of characterizing and identifying the drivers based on DBA. Earlier studies have used driving simulators as their primary data source to collect data from vehicles and drivers. Wakita *et al.* [147] proposed identifying drivers based on their car following behavior using the data collected from driving simulators. Several driving and vehicle-related signals such as those from gas and brake pedals, vehicle speed, and the distance from the next vehicle, as well as GMM were used to characterize driving behavior. In another work, Miyajima *et al.* [148] focused on distinguishing the drivers based on their car following patterns using the headway distance from the lead vehicle, accelerator and brake positions, and vehicle speed. The car following behavior was modeled based on GMM. Qian *et al.* [149] verified the authorized drivers by modeling their dynamic driving behavior. To this aim, they collected signals from steering and gas and brake pedals using a simulator. The authors employed Fast Fourier Transform and Principal Component Analysis in the preprocessing step and SVM model for classification. In a more recent work, Espino-Salinas *et al.* [150] took advantage of vehicles' acceleration, brake, and steering wheel angles collected from driving simulators. The authors adopted a genetic algorithm to perform recognition task. Although the use of driving simulators has the advantage of flexibility in providing various driving, road, traffic, and weather conditions, there is one shortcoming involved, that is the capability of simulators is limited in demonstrating and capturing the exact real-world driving experience and conditions. For instance, it has been shown [50] that steering wheel angles and lateral position signals obtained by simulators can be different from those collected in real-world scenarios.

With the advent of intelligent vehicular and in-vehicle communication systems [151–153], nowadays, obtaining real-world data from drivers and vehicles is not exceedingly difficult. Driving and driver-related data in naturalistic driving settings is collected from different sources in the literature including CAN-bus signals, smartphones, GPS, IMUs, or wearables. Several methodologies of driver identification have been proposed based on the data collected from CAN-bus using the On-board Diagnosis system (OBD-II). Kwak *et al.* [75] used 26 features related to fuel, transmission, wheel, and engine, which are collected from in-vehicle networking, along with 11 more extracted features using statistical analysis and discrete wavelet transform for driver identification. This method achieves about 87% accuracy for five drivers employing an SVM classifier. Bernardi *et al.* [128] addressed driver identification in the context of infotainment in which the

ADAS in the vehicle can be customized to specific members of a family depending on who is driving the vehicle. CAN-bus data and an MLP network are utilized to perform the evaluation. Some other studies in this field [63, 74] benefited from the ubiquitous presence of mobile devices such as smartphones and smartwatches.

Recently, Putri *et al.* [154] proposed an identification system based on the drivers' maneuvers including right and left turns. This system works based on acceleration and orientation data collected from smartwatches and is examined in both real and simulated environments. An error rate of almost 18% is reported when using GMM-based approaches. Some studies took advantage of steering signals in combination with several other signals collected from different sources to model the driving behavior of drivers. In this direction, Wahab *et al.* [155] implemented an MLP network consisting of 12 neurons for driver identification and verification. GMM and wavelet transform techniques were used to extract features from raw signals. Their evaluation results indicated an accuracy of 73.13% when brake pedal pressure was used and 80% when both the accelerator and brake pedal pressures data were employed for five drivers. Öztürk and Erzin [156] leveraged cepstral features extracted from vehicle velocity, gas and brake pedals, and car following behavior of the drivers. Using GMM and a prediction window of 10 minutes, the authors obtained an accuracy of about 57% for a subset of 23 drivers from the dataset. Jafarnejad *et al.* [157] extracted 40 different features from gas pedal and steering wheel angle signals. They employed the GMM model to characterize the driving behavior of the drivers. Using a window of 15 minutes for training and a prediction window of one minute, they achieved an average accuracy of approximately 94%, and 88% for groups of 5 and 15 drivers (total of 67 drivers), respectively. Yang *et al.* [158] took advantage of channel state information extracted from WiFi signals while the drivers were performing different driving maneuvers to distinguish between the drivers. The authors used a deep neural network over preprocessed signals to recognize the drivers.

Martínez *et al.* [159] leveraged 14 signals collected from CAN-bus, IMUs, and laser scanners in their proposed driver identification systems. The authors used headway distance, XYZ accelerometers, roll rate, pitch rate, steering wheel, vehicle speed, speed of the wheels, engine RPM, and gas and brake pedal pressures. Their experimental results demonstrated an accuracy of almost 76% using a single-layer feedforward neural network for 11 drivers. Hallac *et al.* [160] investigated the task of driver identification with respect to turning events. The authors used 12 signals from CAN-bus data such as gas and brake positions, accelerations, torque, steering wheel angle, vehicle speed, engine RPM, or throttle position. An accuracy of 50.1% was achieved for a group of

Table 2.5: Overview of DBA-based driver identification methods

Ref / Year	Contribution	Strength	Constraints
Xu <i>et al.</i> [123] 2023	Analyzing drivers' eye movement in addition to their driving behavior	Different road types, high generalization	Not real-world data, higher data dependency
Yang <i>et al.</i> [158] 2023	Extracting features from channel state information of Wi-Fi signals	Testing on real-world and simulation data, affordable, different maneuvers	Extensive preprocessing, scalability, not cost-effective
Espino-Salinas <i>et al.</i> [150] 2023	Using genetic algorithm to optimize feature selection	Considering intelligent feature selection	Not real-world data, limited number of drivers hence not scalable, not cost-effective
Kristianto and Lin [161] 2022	Using unsupervised autoencoders for imposter detection	Cost-effective, different road conditions, generalization	Less scalability, high data dependency
Ding <i>et al.</i> [162] 2022	Analyzing the car-following behavior of the individuals	Generalization	Not scalable, extensive preprocessing, data dependency
Lim <i>et al.</i> [163] 2022	Adapting the analysis window size for different levels of trip difficulty	Accuracy and latency trade-off, various road types	Not cost-effective, not scalable
Sánchez <i>et al.</i> [164] 2022	Transforming accelerometer signals to two-dimensional images	Less data dependency	Relatively low accuracy for identification, extensive preprocessing step
Zeng <i>et al.</i> [165] 2022	Applying transfer learning for driver identification task	Different weather conditions, different vehicle and road types, generalization	Not cost-effective
Ravi <i>et al.</i> [166] 2022	Optimizing LSTM for driver identification	Simple preprocessing, high accuracy	Not scalable, not cost-effective
Kwak <i>et al.</i> [75] 2021	Preprocess data by Wavelet transform and extract entropy and statistical features	Cost-effective	Data dependency, generalization
Abdenmour <i>et al.</i> [167] 2021	Focusing on latency and training time of identification	Several road types	Not scalable, data dependency. computational cost
Rahim <i>et al.</i> [63] 2020	Extracting the zero-to-stable objects from the stream of driving instances	Cost-effective, fairly scalable, least number of extracted features from GPS	Extensive preprocessing or feature extraction step
Rahim <i>et al.</i> [112] 2020	Detecting the zero-to-stable events from GPS data	Less data dependency, unobtrusive, costly-effective	Canyon problem of the GPS, extensive preprocessing
Xun <i>et al.</i> [20] 2020	Unsupervised anomaly detection method for imposter detection	Cost-effective, two types of vehicles, relatively high accuracy	Scalability
Continued on next page			

Ref / Year	Contribution	Strength	Constraints
Putri <i>et al.</i> [154] 2020	Using smartwatch sensors, extracting log-likelihood of drivers model	Practical, less data dependency, real and simulation environment	Extensive preprocessing, scalability
Dang <i>et al.</i> [168] 2019	Mapping drivers' maneuver execution into a lower-dimensional space for identification	Generalization	Data dependency, cost-effective
Mekki <i>et al.</i> [62] 2019	Investigating effect of data anomaly on the identification task	Relatively robust to data anomalies, having stable accuracy for low numbers of drivers	Cost-ineffective, data dependency, not scalable (relatively low accuracy scores for high number of drivers)
Zhang <i>et al.</i> [21] 2019	End-to-end training attention-based model, sliding window segmentation	Simple pre-processing steps, less pre-processing time, high accuracy	High computational cost and not scalable
Chen <i>et al.</i> [169] 2019	Automatically optimizing the window size for driving time series analysis	Considering window-size optimization	High computational cost, not scalable, potential of high number of parameters
Chowdhury <i>et al.</i> [25] 2018	Using GPS data from drivers smartphones	Less data Dependency, driver trips in various timings	Canyon problem of GPS, driver action required, low accuracy
Ezzini <i>et al.</i> [107] 2018	Extracting driving patterns for various time intervals	Considering both driving and driver-related patterns	High data dependency, limited number of drivers hence not scalable
Jafarnejad <i>et al.</i> [157] 2018	Reducing data dependency by using gas pedal position and steering wheel angle signals	Less data dependency, high number of drivers	Scalability, Extensive feature extraction step
Tanprasert <i>et al.</i> [170] 2017	Employing anomaly detection to manage computational resources	Cost-effective and less data dependency	Not scalable, relatively low accuracy
Moreira-Matias <i>et al.</i> [105] 2017	Analyzing drivers' mobility seasonality (the timing and places drivers visit during a period of time)	Affordable, cost-effective, reasonable accuracy	Not scalable, not efficient when used alone
Hallac <i>et al.</i> [160] 2016	Using short time interval events such as a single turn	Cost-effective, using only a few seconds of data, considering various turn types	Less accurate for higher number of drivers (e.g. 5 drivers)
Martinez <i>et al.</i> [171] 2016	Adaptive single-layer neural network	Using non-intrusive data	Scalability and big data
Enev <i>et al.</i> [17] 2016	Extracting statistical, descriptive, and frequency features from signals to capture the characteristics of in-vehicle sensors data	Simple to implement, efficient, various road types	Not scalable, extensive pre-processing step

Table 2.6: Driver identification features comparison (ND: Number of drivers, S: scalability, G: generalization, DD: data dependency, H: High, M: Medium, L: Low)

Ref / year	Model	Source	Signals	ND	S	G	DD
Xu <i>et al.</i> [123] 2023	Siamese + FCN	Simulator, eye tracker	Steering wheel angle, gas and brake pedals, 13 gaze-relate features	24	M	H	M
Yang <i>et al.</i> [158] 2023	DNN	Wireless transmitter and receiver	Statistical features from channel state information	4	L	L	M
Espino-Salinas <i>et al.</i> [150] 2023	RF	Simulator	Brake, gas, steering wheel angle	4	L	L	H
Kristianto and Lin [161] 2022	Deep Autoencoder	CAN-bus	Vehicle dynamics	10, 14	L	H	L
Ding <i>et al.</i> [162] 2022	GMM	CAN-bus, LiDAR	Speed, acceleration, car-following distance, reaction time	8	L	M	H
Lim <i>et al.</i> [163] 2022	CNN-LSTM	CAN-bus	Speed, brake, engine, steering wheel angle	55	L	L	M
Sánchez <i>et al.</i> [164] 2022	Siamese + ResNet	Smartphone	Accelerometer	25	L	H	L
Zeng <i>et al.</i> [165] 2022	GRU-CNN	CAN-bus	Vehicle dynamics	10, 20	M	H	M
Ravi <i>et al.</i> [166] 2022	LSTM	CAN-bus	40 signals	10	L	L	H
Kwak <i>et al.</i> [75] 2021	Ensemble (SVM, RF, XGBoost)	CAN-bus	26 signals + wavelet transform	10	L	L	H
Abdenmour <i>et al.</i> [167] 2021	CNN	CAN-bus	51 signals	10	L	M	H
Rahim <i>et al.</i> [63] 2020	RF, KNN, SVM	GPS data	Speed, orientation, acceleration	25	M	L	L
Rahim <i>et al.</i> [112] 2020	SVM	GPS data	Speed, orientation, acceleration	25	M	L	L
Xun <i>et al.</i> [20] 2020	CNN, LSTM, SVDD	CAN-bus	23-27 signals on average	45	L	M	H
Putri <i>et al.</i> [154] 2020	GMM, SVM	Smartwatch	Acceleration, orientation	90	M	M	M
Dang <i>et al.</i> [168] 2019	Siamese + LSTM	GPS, vehicle dynamics, map	Speed, acceleration, yaw rate, steering wheel speed	32	L	H	M

Continued on next page

Ref / year	Model	Source	Signals	ND	S	G	DD
Mekki <i>et al.</i> [62] 2019	LSTM-FCN, SVM	CAN-bus, GPS, wearables, simulator, smartphone	Vehicle dynamics, ECG	10, 6, 64	L	M	H
Zhang <i>et al.</i> [21] 2019	Attention-based DNN	CAN-bus	51 signals	10	L	M	H
Chen <i>et al.</i> [169] 2019	Autoencoder	CAN-bus	10 signals	10	L	L	H
Chowdhury <i>et al.</i> [25] 2018	RF	Smartphone	GPS, acceleration	38	L	L	H
Ezzini <i>et al.</i> [107] 2018	SVM, RF, Adaboost, DT, MLP, GB	CAN-bus, GPS, wearables, smartphone	51 signals, ECG	10	L	L	H
Jafarnejad <i>et al.</i> [157] 2018	GMM	CAN-bus	Steering wheel and Gas pedal	67	L	L	L
Tanprasert <i>et al.</i> [170] 2017	Feedforward NN	Smartphone	GPS, accelerometer	10	L	L	L
Moreira-Matias <i>et al.</i> [105] 2017	Ensemble (SVM, RF, DT, LVQ)	In-vehicle data recorder	11 features	242	L	M	H
Hallac <i>et al.</i> [160] 2016	SVM, RF	CAN-bus, GPS, IMU	Steering, gas, brake, throttle, acceleration, engine	64	M	L	L
Martinez <i>et al.</i> [171] 2016	Feedforward NN	CAN-bus, IMU, accelerometers	Vehicle dynamic, headway distance	23	L	M	H
Enev <i>et al.</i> [17] 2016	SVM, RF, NB, KNN	CAN-bus	15 signals brake, engine, transmission	15	L	L	H

5 drivers using the RF classifier. Some studies investigate driver identification aiming at reducing data dependency of driver identification methods. Chowdhury *et al.* [25] demonstrated that it was feasible to distinguish the drivers by collecting data from the smartphones GPS. Moreover, Rahim *et al.* [63, 112] demonstrated experimentally that applying classification methods on data extracted from GPS could have comparable performance to other types of data sources.

Due to the fact that sensors that are used to collect data for driver identification are not completely robust to malfunctioning or failure [62], it is normal to have data anomalies within the sensor data. Consequently, having a proper strategy to detect and correct anomalous data in order to have stable results can be beneficial. Mekki *et al.* [62] investigated the need for data anomalies detection and correction for driver identification

schemes. The performance of their method was tested with the presence/absence of data anomalies detection techniques. The obtained results showed that applying anomaly correction, in most cases with different anomaly rates, could yield better results. Moreover, it is worth mentioning the study of Tanprasert *et al.* [170], which considered data anomaly detection as a technique to manage the computational resources in their framework. The authors proposed to use anomaly detection one stage before the driver identification to recognize any deviation from normal driving patterns.

There have been some efforts toward real-time applications of driver identification. Hallac *et al.* [160] investigated the possibility of identifying drivers in a short amount of time. The results showed that even a small part of a road trip such as a single turn could be used to identify the unique patterns in driving behavior. Furthermore, Ezzini *et al.* [107] investigated potential ways to optimize and improve driver identification with respect to timing and accuracy. They leveraged not only the driving behavioral features but also driver-related patterns. The obtained results demonstrated that these types of features such as heart rate could further improve the performance of the models. The authors also suggested that adapting the size of the processing window was effective for having a reliable yet relatively fast model. Also, Jafarnejad *et al.* [172] examined the effect of adding data records incrementally on the performance of the driver identification models. To do so, the authors defined a metric, namely *time to detection*, to assess the performance. The results indicated that such approach could improve the prediction time such that for a scenario of 5 drivers, more than 75% of the true predictions occurred in less than 65 seconds.

Most recently, deep learning approaches have demonstrated promising performance in a variety of fields, including intelligent transportation systems [11], in comparison with classical approaches. Some methods are devoted to extracting hidden features in driving data with the purpose of driver recognition [21, 62, 75, 162, 166, 167, 169]. Chen *et al.* [169] proposed to use a deep non-negativity constrained autoencoder to extract features from the data. In this model, after optimizing the window size for data segmentation, hidden features are extracted by a greedy method layer by layer. The extracted features will then be fed to a softmax classifier to obtain the probabilities for each driver. Campo *et al.* [173] managed to achieve an accuracy of 76% for a group of four drivers benefiting from 20 different features and an MLP with two hidden layers. Echanobe *et al.* [174] used a single-layer feedforward network and obtained an accuracy of about 90% for only one group of 11 drivers. To better compare with other works, it should be noted that they used a prediction window of 2 minutes and extracted 42 different features from 14 signals

such as vehicle speed, roll and pitch rate, XYZ accelerometers, gas and brake pedals, steering wheel angle, and engine RPM. Zhang *et al.* [21] proposed attention-based Deep Convolutional and recurrent network GRU and LSTM. The models are fed with raw data without feature engineering process. The layers of the neural network are responsible to extract the hidden temporal dependencies within the data. Zeng *et al.* [165] proposed using transfer learning for deep learning-based identification approaches to avoid from-scratch model training when new drivers are added to the system. Lim *et al.* [163] adopted four different features extracted from CAN-bus signals, including steering wheel signals, velocity, and brake pedal to identify the drivers. They used a combination of convolutional neural networks and recurrent units followed by a softmax layer.

Driver Verification

There are a few studies focusing on driver verification or impostor detection, also known as driver authentication, whose aim is to detect when an individual is not from a group of authorized or registered drivers. In other words, the purpose is to identify the person behind the wheel from an open-set of drivers. It is noteworthy that this task is different from driver identification in which we try to identify a driver among a closed-set of known drivers. Rahim *et al.* [112] defined and created a *reject* class in their experiments for this task. An SVM classifier was then trained on GPS data from not only a group of registered drivers but also the drivers in the generated *reject* class. However, this method is limited due to the fact that the accuracy of the impostor detection is highly dependent on whether or not the impostor demonstrates the same driving behavior as the defined behavior in the *reject* class. Similarly, Martínez *et al.* [171] trained a single-layer neural network over the data collected from CAN-bus, IMUs, and the laser scanner. The network was trained over the data from several authorized drivers as well as the rest of the drivers defined as *others* class. The results indicated an accuracy of 50% for 5 drivers. Virojboonkiate *et al.* [106] investigated the possibility of impostor detection using histograms of acceleration data. The authors used a single-layer neural network and groups of 4 drivers in their experiment. An impostor F1 score of 56% was achieved. Nevertheless, the employed dataset in their study is limited to the campus shuttle buses. Kristianto and Lin [161] adopted a deep autoencoder network to detect the anomalies within the driving behavior of drivers which can be further employed to identify an authorized driver. Their results indicated an average accuracy of 97.90% for 10 drivers using 51 different features extracted from CAN-bus signals. Furthermore, Xun *et al.* [20] proposed using an unsupervised anomaly detection method, namely Sup-

port Vector Domain Description, to detect imposters. The authors suggested using the output of a CNN model as an input for anomaly detection. To detect imposters, their data samples were mapped to the feature space and the decision was made based on whether or not the mapped point is in the optimal hypersphere. Their results revealed an accuracy of above 95% for two different types of vehicles.

Siamese neural networks [175] are introduced for the scenarios where only a limited amount of data is available. The basic idea of these networks is carried within the term *Siamese* used in the name, which means *twin*. These networks refer to any desirable twin networks (i.e., identical architecture with shared weights) that are given two different samples to learn and measure the similarity. The purpose is to use the same network to learn latent representations each of which belongs to an input. These representations can then be compared against each other. The output of such architecture can be used to check whether two samples (e.g. faces or signatures) are the same. These networks are particularly beneficial in scenarios where there exists a large number of classes with a limited amount of data of each class. They have shown promising results for image verification tasks such as face verification, handwritten character recognition, or signature authentication [176, 177]. Moreover, triplet loss and representation learning have been widely used in other research areas such as natural language processing or the image classification domain, with a lot of focus on face recognition [178]. However, a few related works in the literature have made efforts to extract embeddings for other purposes rather than word or face embeddings. Liu *et al.* [179] leveraged a predictive model in combination with cosine similarity to characterize traffic interactions based on the information obtained from road embeddings, namely Road2Vec. Each road segment was mapped to a vector to fully capture the spatial heterogeneity of interactions in their methodology.

Dang *et al.* [168] leveraged Siamese concept based on two-way LSTM networks with attention mechanism as the base model. They analyzed left-turn maneuvers of 32 drivers using signals such as vehicle velocity, accelerations, and steering wheel speed. Their results demonstrated an AUC of 70% for different groups of drivers (2 drivers per group) while executing one left-turn maneuver. Jafarnejad [180] employed deep residual learning to extract the driver embeddings. In the context of driver verification, an equal error rate of 0.107 was achieved for 15 drivers. Sánchez *et al.* [164] took advantage of these networks as well as transform learning for the task of driver verification. The authors adopted an already trained neural network architecture in image classification, namely ResNet-50, for their proposed Siamese architecture. Their results indicated an error rate of almost

29%. One limitation of such methodology is that a major preprocessing step is needed, i.e., to transform one-dimensional accelerator data into two-dimensional images. Yang *et al.* [181] proposed a deep learning approach to efficiently extract the embeddings from the vehicular sensor data. The authors took advantage of the temporal convolutional neural networks as well as the wavelet transform to map the driving intervals to the embeddings vector. Their architecture was tested on a dataset extracted from a driving simulator. The results indicated an accuracy of approximately 83% for the pairwise (2-way) comparison using the Gradient Boosting classifier over the embeddings. In a more recent work, Xu *et al.* [123] benefited from the Siamese network to verify the person behind the wheel. They used blocks of fully convolutional networks followed by batch normalization, ReLU activation, and a squeeze and excitation block. Their proposed architecture was tested on a dataset extracted from a driving simulator. The average results for 20 drivers indicated a precision of 96.91%, and a recall of 95.80%.

To sum up, literature review for driver identification revealed that DBA plays a pivotal role in classifying the drivers. However, most of the solutions suggested for DBA-based driver identification such as [17, 107, 160, 171] relied on extensive data collection ensuring the algorithms have enough samples to work on [105, 154, 182]. Such a requirement imposes important limitations on the practical application of driver identification systems. Furthermore, having a scalable driver identification method to be used for a large number of drivers is desirable. The performance of solutions discussed are mostly (e.g. [21, 107, 160, 169, 171, 183]) tested over a limited number of drivers. Hence, the generalization ability of the models (i.e., their ability to be effective for new and unseen data) for a large number of drivers is still open to question. Also, it is of utmost importance to investigate the ability of the methods to handle the need for extra computational power and other resources when the number of drivers increases considerably. In addition, the computational power and cost of the algorithms used within the driver identification systems have always been a major priority for real-time applications of driver identification including real-time authentication or car-theft detection [63, 112, 164, 184]. It is evident that the systems with higher performance and lower complexity are preferred. Moreover, although sustained effort has been devoted to making the best of the large amount of data generated from the connected vehicles, privacy implication and potential compromises [17] have received less attention. More and more data is being shared between the vehicles and third parties, vehicle manufacturers, or road infrastructure. This data streaming and driver identification systems should not risk the privacy of the drivers.

2.4.2 Driver Inattention Detection

According to the report on road safety by WHO in 2018, road accidents are among the top 10 causes of death for all age groups, having an unacceptably high number of 1.3 million fatalities each year [1]. Human factors are among the main causes of critical crashes or near-crashes situations. According to the National Highway Traffic Safety Administration (NHTSA) [15], in 2018, 8% (i.e., 2628) of all fatal crashes in the United States were caused by distracted drivers resulting in more than 2500 deaths. Also, an in-depth analysis of 100-car Naturalistic Driving Study revealed that drowsy driving increases the possibility of crashes and near-crashes by 4 to 6 times, contributing to 22% to 24% of all crashes or near-crashes, whereas secondary-task distracted driving increases the risk by three times, becoming a major factor in more than 22% of crashes and near-crashes [185]. These statistics indicate how inattentive driving poses a danger to the people in the car, pedestrians or cyclists, and how it remains a critical contributing factor in fatal crashes. Consecutively, there has been a growing interest in detecting the driver inattention early and raise a warning signal at the right time.

Driver inattention is defined as having not sufficient or not at all attention to the activities vital for driving safely [186]. This diminished or lack of attention is the result of either concentrating on a secondary task such as another activity, object, or thought while driving (distraction) or feeling drowsy and fatigued (drowsiness). Distracted driving occurs when the driver is engaged in any activities that divert their attention from the principal task of controlling the vehicle. These activities refer to anything that makes the driver look away from the road (visual), be lost in thoughts (cognitive), or take their hands off the wheel to do something else (manual) [187]. Driving is a highly demanding task, and it has been shown [188] that no matter what causes driver inattention, substantial changes will be evident in the driving performance of the driver or their biological and physical reactions. Researchers have used these changes to measure driver inattention levels. These measures can be divided into four categories including driving behavior, biological, driver physical, and subjective measures [145, 189]. In this section, we focus on driving behavior measures and how using DBA can lead to better and more accurate detection of drivers drowsiness or distraction, and traffic safety. For a comprehensive overview of driver inattention methods and their application in safe driving and automated vehicles, see the surveys by Kaplan *et al.* [189] and El Khatib *et al.* [145].

Reviewing the literature revealed that lateral and longitudinal acceleration, steering angle, and speed are among the commonly used [45, 46, 190] driving behavior features

for detecting distracted and drowsy states as they have noticeable negative impacts on driving behavior such as sudden steering wheel turns or increase in vehicle speed. Wang *et al.* [46] showed that employing both lateral and longitudinal acceleration as well as steering wheel angles resulted in the least average error rate of 0.23. Dehzangi and Masilamani [45] used three vehicle-based measures including acceleration, braking, and steering wheel angle to detect drowsiness through a controlled experimental environment. Segmenting the data by the sliding window technique, the authors were able to reach the accuracy of almost 99 % for one driver. Bingham *et al.* [110] came to the conclusion that drivers' driving behavior varies considerably when they start texting while driving. A limited number of studies [18,191] used the time headway metric which is the amount of time it takes for a vehicle to reach the position of the next vehicle. Zhang *et al.* [191] investigated the correlation between time headway and the level of drowsiness. The result indicated that the higher the fatigue level of the driver, the shorter the time headway.

A limited number of studies investigated cognitive distraction since, in most cases, there is no apparent and observable exterior sign, making it challenging to detect through analyzing driving behavior. Liao *et al.* [192] observed that analyzing driving behavior of the drivers in addition to their eye movement could substantially improve the performance of cognitive distraction detection at the stop-controlled interactions scenarios. The authors defined distracted driving as driving while performing clock task. An accuracy of almost 95% was achieved when both driving behavior and eye movement measures were used. Moreover, Xie *et al.* [104] have investigated the possibility of detecting cognitive distraction based on data collected from smartphones sensors including IMU and GPS. Both temporal and spectral features were extracted from the raw data after removing the noise. It was shown that lateral maneuvers were an important factor to detect cognitive distraction. The results showed an accuracy of 87% with the window-size of 80 seconds.

A limitation of some of the studies for driver inattention detection, such as works [110, 192], is that they do not consider different driving styles of the individuals as a factor in their system design. It has been demonstrated [10] that driving styles such as normal, mild, or aggressive styles have impacts on driver inattention detection. Hu *et al.* [10] proposed an abnormality index that is capable of distinguishing various driving states such as normal, alcohol-impaired, distracted, or reckless. This abnormality index is the scaled value of σ^*_{TP} which is the variance of throttle position multiplied by the standard deviation of the vehicle speed error. Taking three different driving styles into account (mild, moderate, and aggressive), the average range of the index for phone usage while driving is around 0.20 to 0.25.

As mentioned earlier, driving behavior measures are not the only ones to help detect when inattentive driving is happening. Combining various types of measures may lead to better performance and increase driver inattention detection accuracy. In this direction, Gwak *et al.* [54] used data fusion by using different features from driving behavior, physiological signals such as ECG, and visual behaviors such as blink frequency or PERCLOS. The result showed a high accuracy of almost 95% for moderate and almost 82% for slight drowsiness. Moreover, it was shown that considerable changes in the drivers drowsiness level had more effects on biological and physical measures rather than driving behavior ones. In another work, Wang and Xu [116] investigated the effect of using driving behavior characteristics as well as eye features such as PERCLOS, blink frequency. Nineteen different driving features related to the steering wheel, lateral positions, and vehicle speed were collected in addition to four eye features using an advanced driving simulator. Their results indicated that higher accuracy was achieved when both feature types had been used.

Driver inattention detection as well as driver monitoring systems show the potential to mitigate several adverse effects of distracted or drowsy driving and to improve road safety. Literature review indicated that it is of utmost importance to recognize the inherent limitation of data acquisition for driver inattention and crashes/near-crashes situations as this may cause severe injuries and fatalities, which is why most studies exploit conducting experiments in an extremely controlled environment such as [10,18,110] or collecting data via driving simulators such as [45,46,192]. Consequently, it should be noted that the obtained results may not necessarily hold when applying to naturalistic driving data. In addition, having highly accurate results should not, on its own, be sufficient to judge the performance of a driver attention monitoring system in real life. Other factors such as the number of drivers participating in the research, fault tolerance of the system, or considered levels of inattention are of fundamental importance. For instance, most studies in the literature such as [10,193] have considered only two levels of drowsiness including fully alert and drowsy. However, this type of classification does not seem to alert drivers properly [54,116], not warning the driver before serious crash risks or having a high number of false/not necessary alarms. Compared with other types of measures for driver inattention detection, especially biological and physical measures [189], driving behavior measures may have a lower accuracy level as driving behavior can be radically different for individuals. This suggests the need for creating an efficient and reliable driver profile as it seems a challenging task to make the driver attention models fit for every driver.

2.4.3 Driving Style Detection

Driving style can be defined as the way a driver controls the vehicle considering external factors such as road and traffic conditions or weather. The potential of understanding the driving style of individuals (e.g. normal, risky, eco-friendly, or aggressive) through analyzing the driving behavior has been widely investigated since one of the early works in the field [194]. As defined by the AAA Foundation for Traffic Safety, aggressive driving is any intentional unsafe driving behavior [195] such as speeding, tailgating, running red lights, etc. Engaging in aggressive behavior has impacts not only on the drivers themselves but also on other drivers as well as pedestrians and cyclists. Many crashes can be prevented and many lives can be saved each year if such behavior is detected properly. Furthermore, according to the 2020 tracking transport reports by The International Energy Agency [16], the transport sector generates 24% of CO₂ emissions in the world. In fact, road vehicles are responsible for almost 3/4 of this sector's CO₂ emissions. Although research toward hybrid electric or battery electric vehicles are beneficial for energy management [76, 196], DBA methods like adapting driving style to reach the optimum fuel consumption are needed to maximize the effect. In the following, we survey recent DBA-based approaches for driving style detection.

Recognizing and analyzing driving style can assist drivers to manage their energy consumption such as fuel consumption or charging sustainability specific for hybrid electric vehicles. For example, Yang *et al.* [197] investigated methods to minimize energy consumption for hybrid electric vehicles for drivers taking account of different driving styles. The authors were able to reduce energy losses by more than 7% and 5% for aggressive and moderate drivers, respectively. Magaña and Muñoz-Organero [92] introduced Artemisa, a driving assistant which offers recommendations for fuel saving based on customized drivers' driving style. As an example, to suggest optimal average speed for drivers, their system analyzes road conditions, weather data, and whether the drivers are driving fuel-wise efficiently (eco-friendly) or not (normally or aggressively). The results demonstrated that leveraging such system could reduce fuel consumption by almost 11%. Similarly, detecting driving styles can be beneficial for curve speed modeling. As the study by Chu *et al.* [56] suggests, aggressive drivers tend to drive faster while entering a curve. These results are of great value to adjust a speeding threshold for various styles to prevent side-slips or rollovers at curves.

Indulging in dangerous driving maneuvers and events is among the signs of aggressive and dangerous driving styles. Several studies such as [90, 103] in the literature have

taken advantage of maneuver detection to classify driving styles in terms of duration or frequency of the maneuvers. Bejani and Ghatee [90] introduced a context-aware system whose architecture consists of three main modules, namely maneuver, traffic, and vehicle classifications. Three different types of maneuvers including U-turn, turns and lane changing are detected and classified into dangerous and safe groups. The results from these three modules are then integrated through a rule-based fuzzy system to identify the driving style. An accuracy of 94% was achieved. Li *et al.* [103] investigated driving style detection based on transition probabilities between various types of maneuvers. The results indicated that high-risk drivers are more likely to exhibit near following or limited lane changes behavior. Ma *et al.* [198] proposed a novel approach in which a driver is labeled by experts to have a specific driving style, and then their corresponding driving maneuvers (overtaking, lane changing, uniform rectilinear motion, and car following) are labeled the same style. Drivers are then classified based on exhibiting particular maneuvers. They achieved an accuracy of 95% for 20 drivers. Maneuver-based classifications, however, can be limited with respect to the assessment of exact timing/duration of the maneuvers or types of the maneuvers [120]. Bejani and Ghatee [120] proposed a driving style classification system namely Ex-CADSE which can analyze driving behavior in a long time interval and classify the drivers effectively. This is helpful since, in a short time interval, it is possible that dangerous drivers behave as normal drivers, making it even harder to classify them.

Throttle position of the vehicle is a contributing feature for distinguishing driving styles, especially aggressiveness as aggressive drivers exhibit sharper changes in throttle position values. Shi *et al.* [19] took advantage of this feature to firstly normalize the driving behavior and secondly to propose an aggressiveness index based on statistical features such as variance or energy spectral density of throttle position. The higher the index value, the more aggressive the drivers. Wang *et al.* [43] made efforts toward reducing the labeling task's endeavor for driving data by proposing a semi-supervised method. Employing throttle position and the speed of the vehicle, the accuracy was improved by about 87%.

Several studies proposed to benefit from longitudinal behavior of the drivers to classify driving styles. Wang *et al.* [57] characterized a driver's driving style based on their car-following behavior. Driving patterns were extracted from the vehicle's acceleration, relative range, and relative range rate between the target vehicle and the lead vehicle. The results indicated that most drivers tend to take harsh acceleration/brake to get near/fall behind the lead vehicle while driving at a close distance from it. Han *et al.* [41] addressed

the problem of driving style detection for curvy roads. The authors took advantage of path following behavior of the drivers to divide them into 7 groups from normal to aggressive. Following patterns were extracted from the speed and throttle position of the vehicle. Also, time headway can be a contributing factor in the recognition of risky drivers. Risky drivers are exhibiting small and decreasing time headway [55].

Based on the literature review, it can be seen that less attention has been devoted to the labeling issue for the driving style detection task. On the one hand, recent developments in vehicular and in-vehicle technologies [199–203] have helped researchers access a tremendous amount of data, referred to as *big data*. On the other hand, Most DBA approaches such as [57, 92, 103, 109, 120, 198] have used classical machine learning, deep learning, and fuzzy-based approaches for learning the driving style of drivers in a supervised way in which the true labels/classes of the data are a must. In a real-world setting, the generated data is neither pre-processed nor labeled. First, it is time-consuming to manually label all of the collected data, and second, it is open to the subjective assessment of the individuals. More novel approaches [43] are needed to model driving styles with unsupervised models such as clustering to divide the drivers into groups or semi-supervised which can be applied even with a few labeled samples. In addition, from the literature review, it can be inferred that some of the datasets collected and used for this task such as those in [43, 55, 56] are imbalanced with respect to the age and gender of drivers, having the potential to causing several biased conclusions. A person can exhibit various driving styles not only under various traffic, weather, or mood conditions but also as they grow up [73]. Therefore, taking account of such factors in a balanced way in datasets is of key importance. The result of a study by Yan *et al.* [109] revealed that the drivers risk level changed dramatically with different conditions. The authors investigated the effect of various conditions on drivers driving risk state based on data collected from the rural and urban roads, three types of weather, and traffic volume.

2.4.4 Impaired Driving Detection

The term *drunk driver*, also called *alcohol-impaired driver*, refers to a driver driving a vehicle while their ability to control the vehicle is compromised by using alcohol (a blood alcohol concentration (BAC) of .08 grams per deciliter (g/dL) or higher [204]), influencing reasoning, brain function, and reaction time, all of which are critical for safety. According to the traffic safety fact reports by NHTSA [204], in the USA, 29% of total fatalities in 2017 and 2018 were in alcohol-impaired driving crashes, claiming

more than 10,000 lives each year. Therefore, research on early detection of alcohol-impaired driving and ultimately eliminating it has been gaining attention from academia and industry for its prominent role in traffic and road safety. The methods for alcohol-impaired driving detection can be divided into two broad categories: (1) direct and (2) indirect. The former includes accurate and reliable measuring of the BAC in a driver's blood. Breath-based and touch-based sensors can be utilized to read the alcohol levels of the drivers blood. Alternatively, the latter involves detecting alcohol-impaired drivers employing DBA or drivers physiological measures such as ECG, heart rate, etc.

Driving under the influence of alcohol can have dramatic effects on driving behavior measures of the drivers, especially those who vary tremendously based on the drivers driving states such as lateral position, speed, steering angle, etc. The analysis variance of Li *et al.* [48] demonstrated that there was a substantial difference between driving behavior of the driver under normal and alcohol-impaired conditions. In fact, it was revealed that the higher the level of the BAC, the more the slope and the range of the lateral position of the vehicle. Moreover, the range and standard variation of the steering wheel angle for a driver is higher while driving drunk. The authors then exploited the statistical features of both the steering wheel angle and lateral position of the vehicle and SVM model to classify normal and alcohol-impaired drivers. Their model reached an accuracy of 80%. Shirzai and Rad [49] took advantage of the fact that alcohol-impaired drivers have difficulty maintaining the appropriate lateral position, resulting in several issues such as drifting or weaving. Steering wheel angles and the lateral preview error was used to build a model which could effectively classify sober drivers from alcohol-impaired ones. Moreover, Li *et al.* [205] performed feature selection based on RF to reveal the most contributing features in alcohol-impaired driver classification. The result showed the crucial role of acceleration and brake depth, vehicle acceleration, and steering.

Furthermore, the experiment conducted by El Masri et al [47] has illustrated that alcohol-impaired drivers exhibit more variations and peaks in their driving data, especially the speed and lane position of the vehicle, rather than normal drivers. The authors exploited a window size of 10s to label the data. An accuracy of 82% was achieved. Yao *et al.* [117] showed that the average of speed and lane position of the vehicle were among the main alcohol-impaired driving indicators on both straight and curved roads. In fact, the obtained results confirmed that the effects of alcohol were more notable while driving in curved road types, leading to a higher speed and more variation in the lateral position. Hu *et al.* [137] observed that an alcohol-impaired driver exhibited large fluctuations in the throttle position as well as substantial delays. Chen and Chen [36] investigated the

effect of a hybrid approach on alcohol-impaired driving detection. The authors combined physiological and driving behavior measures. Although the accuracy of the model is relatively low because of the limited sample size, the authors revealed several interesting results. It was shown [36] that alcohol-impaired drivers were most likely slower in adjusting the lateral position of the vehicle. Moreover, alcohol-impaired drivers experience more lane crossing and an increase in steering angle.

Finally, all or part of the data used in DBA-based drunk driving detection studies [36, 47–50, 117] has been collected through driving simulators. It is evident that for several tasks of DBA including drunk driving detection, driving simulators exhibit the potential to provide a controlled environment [206], making it possible to conduct experiments safely and without any harms. Yet, the fact remains that driving simulators may not always make available the exact real-world conditions. For instance, it is noted that lateral position and steering wheel angles collected from simulators might be different from those obtained from real vehicles [205]. In addition, identifying different levels of BAC based on DBA is not mature enough. In fact, most possible solutions for drunk driving detection such as [47, 49, 137, 206] appear to classify the driving data into two categories of normal and drunk drivers. It is of interest to know whether or not drivers exhibit abnormal behavioral performance while driving with a BAC of less than 0.08 such as 0.05. Furthermore, it has been stated that the number of drunk driving crashes during the night in 2018 was approximately 3.5 times more than those that occurred at daylight [204]. Such numbers highlight the need to further investigate drunk driving behavior at night.

2.4.5 Driving Behavior Detection and Prediction

Autonomous vehicles play crucial roles in shaping the future of transportation, having the potential to revolutionize mobility, and making transportation safer. By removing human errors, AVs can significantly reduce accidents and fatalities on the roads. Notably, *behavior* prediction, also known as *motion* prediction or *trajectory*¹ prediction, is essential for ensuring the safe and efficient interaction of AVs with other road users. Accurately predicting the future behavior of surrounding agents², including pedestrians, cyclists, and other vehicles enables autonomous systems to make informed decisions and plan

¹Throughout this thesis, the terms "*behavior*," "*motion*," and "*trajectory*" are used interchangeably.

²Note that the term "*agent*" merely refers to a diverse set of traffic participants, including vehicles, cyclists, and pedestrians.

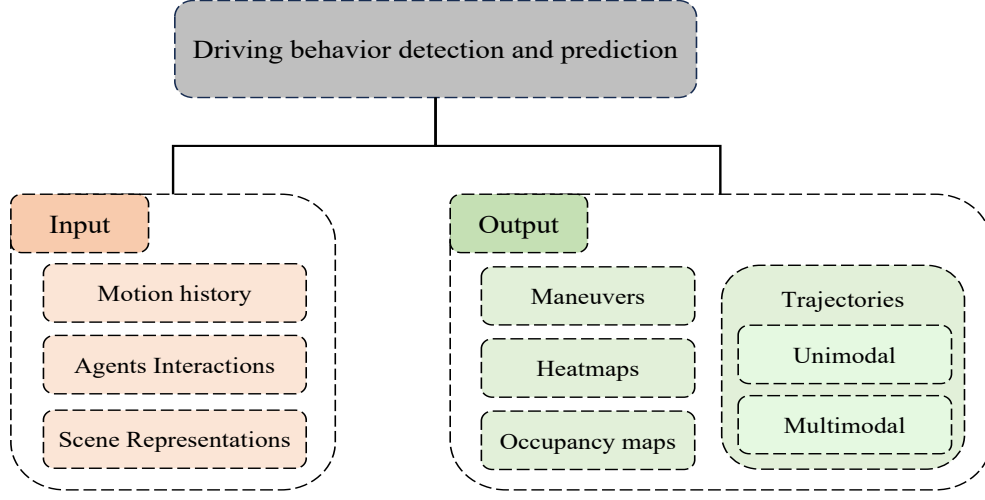


Figure 2.2: Taxonomy of driving behavior detection and prediction

their movements accordingly. This capability is vital for maintaining smooth traffic flow, preventing collisions, and enhancing overall road safety [207, 208]. Moreover, DB detection and prediction facilitates efficient route planning and helps autonomous vehicles anticipate and respond to unexpected situations effectively. Currently, the related work on agent behavior prediction can be categorized based on the input and output type of the model and the adopted approach to predict target agents' future behavior. In Fig. 2.2, we provide a categorization of the previous studies according to the first two metrics. Note that the models are discussed in Section 2.3 in general.

Behavior prediction methods that rely solely on the motion history of agents have been a significant area of research in the field of autonomous vehicles. These methods leverage past observations of agent movements to estimate their future trajectories. By analyzing the historical data, such as position, velocity, and acceleration, these methods aim to uncover patterns and behavioral tendencies that can inform predictions. These methods often take into account factors like speed or direction to improve accuracy. Several studies benefit solely from the motion history of a target agent to predict its future behavior. For instance, Zyner *et al.* [39] modeled the TA's behavior by processing its track history using an LSTM-based encoder-decoder architecture at unsignalized intersections. The authors used lateral and longitudinal positions, TA's velocity, and heading over several time steps. While these approaches may not capture complex contextual information or account for the intentions and decision-making processes of agents, they provide valuable insights into short-term trajectory patterns. By leveraging the motion history of agents, these

Table 2.7: An overview of social reasoning methodologies for DB prediction

Method	Main Idea	Strengths	Constraints
Motion history of TA and SAs	Analyzing historical movement patterns to anticipate future behaviors	Agents behavior pattern recognition, feature extraction, efficient	Less accuracy, scalability, not suitable for long term prediction or complex interactions
Social pooling	Aggregating information from multiple agents (e.g., pedestrians, vehicles) in a scene to improve predictions or understanding of their interactions	Interaction understanding, handling ambiguity, group dynamics, reduced overfitting	Increased complexity, data dependency, poor effect on SAs, scalability
Graph representation	Modeling agents and their interactions as nodes and edges in a graph	Structured modeling, flexibility, contextual understanding, cost-effective	How to build an accurate graph, sparsity, overfitting

prediction methods contribute to the development of safer and more efficient autonomous systems by enabling anticipatory behavior and proactive decision-making in dynamic environments. However, it is important to continue exploring and integrating additional sources of information, such as environmental cues and social interactions, to further enhance the robustness of behavior prediction methods.

Interaction-Aware Behavior Prediction

With the advent of perception systems, high-quality cameras and drones, and intelligent vehicular systems, obtaining data from various driving agents and the environment is not exceedingly difficult anymore. As mentioned above, several studies [39, 209, 210] employ the behavior history of a TA to predict its future motion. Nonetheless, non-interaction-aware approaches pay no attention to interactions between a TA and the agents surrounding it. Although the TA’s behavior history is informative and beneficial for short-term predictions, relying only upon it can result in erroneous conclusions, especially in long-term predictions and in complex scenarios. Evidently, taking into account inter-agent interactions and modeling inter-agent behavior dependencies contribute to a better understanding of the driving scenes and ultimately lead to higher prediction accuracy. This section focuses on interaction-aware studies on behavior prediction and presents a concise review on this matter. Table 2.7 provides an overview of methodologies employed in the literature for incorporating interactions among agents. Moreover, a summary of interaction-aware behavior prediction studies are provided in Table 2.8.

Method	Year	Interaction	Main idea	Output	Mu	Ca
Dai <i>et al.</i> [216]	2019	Motion history	Two groups of LSTMs one for considering agents interactions and another for prediction	Trajectory	×	×
Ma <i>et al.</i> [217]	2019	Graphs	Making a two-layer graph from different agents and use LSTM-based network for prediction	Trajectory	×	×
Ding & Shen [218]	2019	Motion history	Using an RNN-based optimization policy network	Trajectory	✓	✓
Messaoud <i>et al.</i> [229]	2019	Social pooling	Modeling the interactions using convolutional and MHA pooling layer, LSTM encoder-decoder	Trajectory	×	×
Ding <i>et al.</i> [220]	2019	Motion history	Pairwise interaction calculation using agents previous maneuvers, fully connected layer for prediction	Trajectory	✓	×
Li <i>et al.</i> [230]	2019	Graphs	Making an undirected graph for agents interactions, using graph convolutions followed by an LSTM-based encoder-decoder	Trajectory	×	×
Deo & Trivedi [231]	2018	Social pooling	Learning local and interaction features using convolutional and pooling layers, LSTM decoder	Heatmap,	✓	×
Vemula <i>et al.</i> [232]	2018	Graphs	Making a spatio-temporal graph using attention, RNN for prediction	Trajectory	×	×
Deo & Trivedi [221]	2018	Motion history	An LSTM for maneuver classification, LSTM encoder-decoder for prediction	Heatmap, maneuvers	✓	×
Alahi <i>et al.</i> [233]	2016	Social pooling	Social LSTM, sharing states between adjacent units	Heatmap, trajectories	✓	×
Gupta <i>et al.</i> [134]	2016	Social pooling	Social GAN, using a MLP followed by max-pooling	Heatmap, trajectories	✓	×

To heighten interaction awareness for behavior prediction approaches, many studies made efforts to model inter-agent correlations among all agents within a driving scene. Early interaction-aware literature applied traditional approaches, including classical machine learning models such as HMM, SVM, and Bayesian networks [211–214]. These conventional methodologies, however, suffer from low performance in long-term predictions, especially for complex scenarios, and compatibility with real-time analysis [215]. Recent advances in deep learning models as powerful tools and their transferability and promising performance in various research fields including vehicle behavior prediction in

realistic scenarios have caused many researchers to use these techniques such as RNNs, temporal CNNs, and GNNs. Furthermore, different techniques are used throughout the literature to model the inter-agent interactions for vehicle behavior prediction. One method to take account of the impact of surrounding agents is to explicitly employ the behavior history of the TA and that of its SAs as the input of the model [216–221]. In this direction, Dai *et al.* [216] proposed using two groups of LSTM-based RNNs to model the interactions among the TA and each of its neighbors as well as the behavior history of the TA next, based on which the future behavior of the TA is predicted. In another work, Ma *et al.* [217] introduced TrafficPredict algorithm to model the inter-agent interactions. The authors designed a system architecture with two layers each of which contains LSTM recurrent units. One layer to obtain the pattern of traffic participants' motions and another to obtain the similar behavior between the same group of traffic participants such as vehicles or bicycles. These approaches are limited since they do not consider the effect of environment and traffic rules on the behavior of the TAs.

An alternative approach to consider social interactions between a large number of traffic participants within a scene is to apply the social pooling mechanism [134, 227, 231, 233] which allows the LSTM units of neighboring agents to share their knowledge with one another. In this direction, Alahi *et al.* [233] introduced the S-LSTM method that allows the recurrent units corresponding to SAs to connect with each other since the authors designed one pooling layer between each existing LSTM cell to make sharing hidden states possible for the agents within an occupancy grid. To model pair-wise interactions between all SAs in a scene, Gupta *et al.* [134] proposed another pooling method called, S-GAN, based on MLP and max pooling. According to this method, a global pooling vector based on relative coordinates between the TA and all of its SAs and their hidden states is computed for each TA. Deo *et al.* [231] introduced an encoder architecture for the task of vehicle behavior prediction, namely CS-LSTM, that applies convolution and max pooling operations over a spatial grid which represents the surrounding of the TA. Nonetheless, the extracted representations of the vehicles are independent of their states, and the local computations are inefficient. To solve this, Messaoud *et al.* [133, 229] proposed employing the multi-head attention pooling method. They generated a representation vector for each vehicle based on an LSTM-based encoder. An MHA module was then applied to consider the inter-vehicle correlations between the target vehicle and its SAs within a spatial grid. It was shown that MHA reduced the number of local computations. One major shortcoming of these approaches is their inefficiency in handling complex spatio-temporal correlations among the traffic participants. Moreover, the distance used

to generate the considered occupancy grid or the number of SAs that are taken into account can affect the performance of these approaches.

More recently, graph-based interaction reasoning has gained attention in the behavior prediction research area as an alternative approach to address the shortcomings of the above-mentioned approaches. Graph-based approaches [126, 223–226, 228, 230, 232] have focused on modeling interactions between different agents as graphs where the nodes represent agents within a driving scene, and the edges are inter-agent interactions. This way, the spatial and temporal inter-agent correlations can be considered simultaneously. In this direction, Diehl *et al.* [234] represented a highway-driving scene as a directed graph. Considering a fixed number of surrounding vehicles, the authors compared the performance of GAT and GCN for the task of traffic prediction. Nonetheless, the generated homogeneous graph disregards the vehicles' dynamics and types. Li *et al.* [230] generated a homogeneous undirected graph to represent the inter-vehicle interactions. Several blocks of graph convolutions were then used to extract underlying features within the data. The future behavior of the vehicles was generated using an LSTM-based decoder. Their proposed model, GRIP, also shares the previously mentioned shortcoming. Park *et al.* [126] introduced FRM that leverages relational inference to learn the social interactions among agents via constructing latent dependency graphs. Wu *et al.* [130] proposed an encoder-decoder architecture to consider temporal interdependencies with MHA and spatial interactions with graph attentions. The outputs of these two separated modules were then aggregated to be fed into the LSTM-based decoder. Li *et al.* [127] proposed the STG-DAT system that contains three main modules, including feature extraction with MLP, representation extraction using GAT as an encoder, and behavior generation employing GRU while considering the kinematic constraints. Mo *et al.* [223] built a directed graph for different groups of agents and proposed considering individual encoders for various types of agents within the driving scene since the distinguished behavior of each agent type can affect their future behavior patterns. Sheng *et al.* [235] designed a distance-based weighted graph for the TA and all of its surrounding vehicles. The spatial graph was then analyzed using GCN. They employed GRU units to generate the future behavior of the vehicles. Gao *et al.* [131] also created heterogeneous sub-graphs for each agent and then a high-order graph to model the inter-agent interactions. Nevertheless, the generated dense graph disregarded the spatial and edge features among the agents.

The literature review indicated that interaction-aware behavior prediction approaches are primarily suitable for specific representations and scenarios. They either do not consider all spatial and temporal aspects or, if they do, create a dense undirected graph to

represent the inter-agent interactions in which it is assumed that a vehicle unexceptionally interacts with all other SAs. Their effects on each other are of the same strength.

Context-Aware Behavior Prediction

As discussed earlier, predicting the behavior of pedestrians and vehicles nearby is an crucial factor for motion planning of AVs and can enhance transportation systems and public safety. Therefore, significant attention has been given to both human and vehicle behavior prediction in the literature. While several studies concentrate on prediction methodologies based on an agent motion history, its social interactions, and the interplay between agents [130, 226, 227, 236, 237], they often ignore the scene context. As a result, their predictions might not accurately conform to the environmental circumstances and layouts such as road structures and traffic rules. In this regard, context-Aware behavior prediction refers to the process of predicting the future behaviors or motions of different agents such as pedestrians and vehicles within a given scene while taking into account the surrounding environment’s characteristics, constraints, and interactions. This approach recognizes that agent behaviors and trajectories are influenced by the scene’s geometry, the presence of obstacles, the locations of other agents, and other contextual factors. Therefore, rather than ignoring the surrounding environment, scene-compliant³ behavior prediction aims to generate trajectories that adhere to the overall structure and dynamics of the scene. In this section, We will focus on context-aware prediction methodologies.

Representing context information and extracting scene features is a decisive step in having an effective behavior prediction system. Scene images and HD maps provide the model with extremely beneficial semantic information on the road typologies and traffic rules since the future behavior of all SAs is strongly contingent upon these two factors. For instance, it is less likely for an agent to perform a right-turn maneuver when such driving ways/lanes do not exit nearby. Consequently, efficiently representing information on surrounding context and environment is crucial for behavior prediction methods so that they generate feasible trajectories. Based on the literature review, standard approaches to integrate scene information include two main methods: (1) benefiting from prior bird’s-eye-view images from the surrounding environment as rasterized data [127, 143, 223, 238–245] or (2) representing the HD maps as vectorized data [131, 132, 246]. An overview of these main two approaches for context-aware behav-

³In this context, the term "scene-compliant" indicates that the predicted behavior align with the layout, context, and rules of the scene.

Table 2.9: Methods of context modeling for driving behavior prediction

Method	Main Idea	Strengths	Constraints
Rasterized modeling	Capturing scene features as pixel-based grids over the contextual image from the driving scene	Preserving fine-grained visual details, more available, encoding spatial relationships, suitable for heterogeneous agents	Less efficient, capturing interactions and temporal dynamics
Vectorized modeling	Representing scenes through structured geometric elements like lines and polylines	Having clear and structured representation, scalability	Loss of visual features, availability, complexity of learning from structured data, how to extract the structured vectorized data

ior prediction is provided in Table 2.9. Moreover, we provide a summary of context-aware behavior prediction models in Table 2.10.

Vectorized scene modeling for behavior forecasting involves representing scenes and contextual information using structured geometric primitives, such as lines and polylines. This approach focuses on encoding spatial information and connections through mathematical representations. In this direction, Gao *et al.* [131], introduced VectorNet which directly extracted structured environment information from the vectorized representation of road elements and SAs’ trajectories. For instance, lane segments within a road were considered as several points constructing a spline, which were further represented as vectors. Once created, the vectors were then fed to a GNN to encode context attributes. Liang *et al.* [132] proposed LaneGCN that encoded semantic information by constructing lane graphs from vectorized data derived from HD maps, where the lanes were demonstrated as consecutive BEV points/nodes, and the lane connections were represented as neighbors. The authors further employed dilated graph convolutions to extract scene attributes. Zhou *et al.* [247] proposed another prediction method, called HiVT. Unlike the previous two approaches that used absolute positions of SAs, their introduced model utilized relative positions while characterizing the scene features to better ensure translation invariance. In another work, Song *et al.* [248] proposed a novel approach, called PRIME, that searched the HD map for potential behaviors before behavior prediction step. PRIME used depth-first-search algorithm to find the possible behaviors. The scene context was then encoded using temporal Convolutions followed by an LSTM. Also, Bhattacharyya *et al.* [246], in their proposed prediction model called SSL-Lanes, adopted spatial attention to aggregate environmental information from both map and surrounding vehicles’ encodings. The authors used LaneGCN for scene model-

ing. However, one shortcoming is that HD maps are mostly available for road sections, limiting their usage to homogeneous agents in particular vehicles.

In contrast, rasterized scene modeling for behavior prediction involves encoding scene information in the form of pixel-based grids or images using CNNs. This approach aims to leverage the visual information contained in images to understand and analyze the interactions and connections between environment and agents. In this direction, Mangalam *et al.* [240] introduced a novel system, called Y-net, that constructed a segmentation map from the BEV images of the scene, which were further fed to a combination of convolutional and pooling layers to extract the semantic information. Employing a similar approach, Chiara *et al.* [239] proposed Goal-SAR model using a segmentation network on BEV images while considering six predefined classes, such as pavements and trees, to extract the feature maps which are later combined with temporal features and fed to their goal generator. Lv *et al.* [238] designed SSAGCN system employing spatial and temporal attention on extracted scene features by a CNN over BEV images and agents interactions features by GNNs concurrently. Instead of directly employing context images, Li *et al.* [127] computed the occupancy map for spatial attributes and velocity map for motion attributes, both of which were then fed to a CNN to extract the scene encodings. Hence, they effectively determined drivable areas for vehicles. Zhao *et al.* [243] created a tensor that integrates motion and scene encodings extracted via a CNN. Their CNN module produced a feature encoding keeping the spatial attributes of the BEV images of the surrounding environment. However, in this way, the images captured at another time will have the same effects on the predicted behaviors. To solve this, Lu *et al.* [244] proposed a dynamic map approach in which they utilized an attention mechanism to choose only the relevant scene images w.r.t. the time. They authors leverage temporal attention to encode the temporal correlation existing in the surrounding context to extract a richer scene latent representation. These intermediate representations are then fed to a CNN to obtain context features. These approaches, however, may assign a high probability to several unlikely behaviors with infeasible outcomes that are non-realistic.

Multimodal Behavior Prediction

Multimodal behavior prediction is critical for AVs to have safe motion planning. Considering a particular driving scenario, the future behavior of the driver/vehicle is multimodal in nature, which means the vehicle can take different possible trajectories. A unimodal approach is the one that provides only one of these potential behaviors [11]. In this direction, Dai *et al.* [216] developed a spatio-temporal methodology based on LSTM re-

Table 2.10: An overview of Context-aware behavior forecasting methods (MU: Multi-modal, I: Interaction-aware)

Method	Year	Context	Main idea	Output	Mu	I
Lv <i>et al.</i> [238]	2023	Rasterized	Extracting context features using CNN and social interactions with graph attentions, TCN for prediction	Heatmap, trajectories	✓	✓
Bhattacharyya <i>et al.</i> [246]	2023	Vectorized	Extracting context and social interaction features using lane graphs, unsupervised clustering for maneuver classification	Trajectory	✓	✓
Chiara <i>et al.</i> [239]	2022	Rasterized	Processing the context information using segmentation and CNN, Fully connected layer for prediction	Heatmap, trajectories	✓	×
Guo <i>et al.</i> [143]	2022	Rasterized	CNN for context feature extraction, ConvLSTM for occupancy map decoder and IRL policy network, transformers for prediction	Heatmap, trajectories, occupancy maps	✓	✓
Song <i>et al.</i> [248]	2022	Vectorized	Depth-first-search on HD map to final plausible behaviors, temporal Convolutions followed by LSTM for scene encoding, attention for social reasoning	Trajectory	✓	✓
Zhao <i>et al.</i> [243]	2022	Rasterized	Extracting scene feature maps using CNN and LSTM-based encoder-decoder for prediction	Trajectory	✓	✓
Lu <i>et al.</i> [244]	2022	Rasterized	Using three modules for extracting features: CNN for context information, GCN for interactions, and TCN for motion history	Trajectory	✓	✓
Mangalam <i>et al.</i> [240]	2021	Rasterized	Using segmentation map and agents motion history, extracting goals, waypoints distribution, and trajectories by U-Net	Heatmap, trajectories	✓	×
Liang <i>et al.</i> [132]	2020	Vectorized	LaneGCN: creating a lane graph from HD maps in which lanes and agents are the nodes, GCN for processing the graph, MLP for prediction	Trajectory	✓	✓
Bhattacharyya <i>et al.</i> [245]	2020	Rasterized	Extracting scene features using CNN and attention, convolutional social pooling for social reasoning	Trajectory	✓	✓

Continued on next page

Method	Year	Context	Main idea	Output	Mu	I
Zhou <i>et al.</i> [247]	2020	Vectorized	Extracting scene features using LaneGCN, encoding local and global embeddings for all agents using transformers, MLP decoder	Trajectory	✓	✓
Gao <i>et al.</i> [131]	2020	Vectorized	Vectornet: creating polyline graphs for local and global interactions from the HD maps, GNN for processing, MLP for prediction	Trajectory	✓	✓
Chai <i>et al.</i> [241]	2019	Rasterized	Extracting agent-centric scene features for each agent using CNN	Trajectory	✓	✓

current units and skip-connections to anticipate the potential behavior of the vehicles. Using historical trajectory data of a vehicle and its nearby vehicles, the authors managed to achieve higher accuracy compared to the maneuver LSTM approach. Lefkopoulou *et al.* [249] leveraged Kalman filter to predict future trajectories of a vehicle. KF was used to analyze the lateral and longitudinal motions of the vehicles. Comparison results indicated an RMSE of about 4.5 for horizon prediction of 5 seconds. In another work, Cao *et al.* [225] designed a novel graphical model consisting of Fourier transform, gated convolutions, and attention mechanism to encode historical trajectory information of the target vehicles and their neighboring ones. Benefiting from the frequency domain, the authors anticipated the future behavior of a vehicle or pedestrian that is most likely to be taken. One drawback of these approaches is that they do not take the multimodal nature of drivers' behavior into consideration. Single mode behavior generation may lead to invalid future trajectories [134] as the prediction model attempts to average various possible behavior modes, which may not be a plausible behavior for an SA. In the following, we review some of the common approaches to address multimodality. Table 2.11 provides a summary of behavior prediction approach which have considered multimodality while producing their predictions.

While early studies [250, 251] have addressed multimodality via regression, recent studies have benefited from other ways to represent various possible behaviors to prevent mode collapse. One way to consider uncertainty is to directly simulate the variations in agents behavior. In this direction, Gu *et al.* [252] proposed an approach, namely MID, that produced future behaviors via moderately decreasing indeterminacy of the predictions using a markov chain and a transformer decoder. Another way to address

Table 2.11: A summary of multimodal behavior anticipation approaches (Ca: Context-aware, I: Interaction-aware)

Method	Year	Multimodality	Main idea	Output	CA	I
Westny <i>et al.</i> [255]	2023	MDN	GRU-based graph encoder, graph attention decoder	Trajectory	×	✓
Zhu <i>et al.</i> [259]	2023	Latent variable	Destination VAE latent variable, MLP-based encoder-decoder	Trajectory	×	×
Janjoš <i>et al.</i> [260]	2022	Attention	Temporal CNN for motion history, graph attentions for social interactions, GRU-based decoder	Trajectory	✓	✓
Huang <i>et al.</i> [261]	2022	Attention	Temporal CNN followed by LSTM for motion history encoding, transformers for social interactions, MLP for prediction	Trajectory	✓	✓
Gilles <i>et al.</i> [257]	2022	Heatmap, sampling	Constructing rasterized lanelets and using GNN for feature encoding	Heatmap, trajectories	✓	✓
Gu <i>et al.</i> [252]	2022	Diffusion	Progressively decreasing the uncertainty across all accessible regions using a Markov chain, transformer-based decoder	Trajectory	×	✓
He <i>et al.</i> [262]	2022	Goals, sampling	Using segmentation for scene modeling and MLP for extracting reward maps	Heatmap, trajectories	✓	✓
Messaoud <i>et al.</i> [254]	2021	MDN	LSTM-based encoder-decoder and attention, CNN for scene feature extraction	Heatmap, trajectories	✓	✓
Gilles <i>et al.</i> [258]	2021	Heatmap, sampling	Temporal convolutions for feature encoding, attention for social reasoning, CNN for scene feature extraction	Heatmap, trajectories	✓	✓
Cheng <i>et al.</i> [139]	2021	Latent variable	CVAE latent variable, attentive dynamic maps for social interaction, LSTM and fully connected layer for prediction	Trajectory	×	✓
Deo <i>et al.</i> [144]	2020	Goals, sampling	CNN for scene feature extraction, IRL for goal estimation and GRU-based decoder for prediction	Heatmap, trajectories	✓	✓
Mangalam <i>et al.</i> [138]	2020	Latent variable	CVAE latent variable, social pooling for agents interactions	Trajectory	×	✓
Salzmann <i>et al.</i> [263]	2020	Latent variable	CVAE latent variable, bi-directional LSTM encoder, GRU-based decoder	Trajectory	×	✓

Continued on next page

Method	Year	Multimodality	Main idea	Output	CA	I
Sadeghian <i>et al.</i> [264]	2019	Latent variable	LSTM-based GAN for prediction, attention and LSTM-based encoder, CNN for scene feature extraction	Heatmap, trajectories	✓	✓
Tang <i>et al.</i> [265]	2019	Latent variable	CVAE latent variable, attention social reasoning, RNN-based encoder-decoder, CNN for scene feature extraction	Trajectory	✓	✓
Makansi <i>et al.</i> [253]	2019	MDN	Breaking up the MDN optimization process into sampling and distribution fitting	Heatmap,	✓	×
Lee <i>et al.</i> [266]	2017	Latent variable	CVAE latent variable, GRU-based encoder-decoder, CNN for scene feature extraction	Trajectory	✓	✓

multimodality is to produce a distribution over potential behaviors by assigning mixture components to distinct behavior modes [39, 253–256], which is prone to overfitting. In this direction, Westney *et al.* [255] designed a novel approach, called MTP-GO, that employed a GNN-based encoder to learn the trajectory maps and graph attentions as a decoder. They addressed multimodality using mixture networks and EWTA loss to address mode collapse. Fu *et al.* [256] took this one step further by generating a navigation map based on the obtained probability distributions. They combined a recurrent-based encoder-decoder model with an attention module as well as an MDN to anticipate the possible behavior of the target vehicle. Such spatio-temporal maps can be further used for future motion planning of the ego vehicle. Other proposed models in the literature [257, 258] have concentrated on generating a probability heatmap to represent the probability distribution. Gilles *et al.* [258] used a bird’s-eye-view image of the SA surrounding context and generated a probability heatmap in their method, namely Home. The output represented the SA goals in the future. To guarantee suitable coverage of the behavior modes, their model extracted a set of final endpoints from the output. In another work [257], same authors applied a GNN to vectorized data derived from HD maps to encode context attributes as lane rasters. These rasters were further employed to construct a probability heatmap representing the SA final goals. These approaches, however, are mainly focused on the behavior prediction of homogeneous agents.

Another family of methods addresses multimodality by predicting the goals [143, 144, 240, 262]. Goal-conditioned models separate the prediction task into two parts, including goal estimation and behavior prediction. A few works [143, 144, 262] have

conditioned the SA's future behaviors on the plans sampled from inverse IRL policies. He *et al.* [262] utilized IRL to optimize policies through integrating context and motion attributes. They used semantic segmentation to encode context features from BEV images of the environment. The authors generated future behavior using a GRU-based encoder-decoder. In a similar approach, Deo *et al.* [144] employed BEV images from the surrounding environment to learn reward and policy maps via IRL upon which the future behaviors of vehicles were contingent. The sampled policies were then fed to a GRU-based encoder-decoder to forecast future behaviors. Although these approaches generate more diverse predictions, most of them are non-realistic. Therefore, Guo *et al.* [143] proposed predicting occupancy maps as an additional step in the model to maintain a balance between the variety and precision of the outputs.

Another method to address multimodality is to implicitly model various behavior modes as latent variables, where the emphasis is on gaining the inherent intents of the SAs, and generate trajectories by GANs [134, 233, 264], CVAEs [138, 139, 263, 265, 266] or Destination VAEs (DVAE) [259]. In this direction, Sadeghian *et al.* [264] introduced a model, namely SoPhie, that used a CNN and LSTMs to encode scene and motion attributes of the SAs, respectively. The obtained features were then concatenated using an attention-based approach. Their model generated future possibilities using an LSTM-based GAN. Gupta *et al.* [134] also leveraged latent variables to implicitly model several behavior modes. Their introduced model, called Social GAN, combined MLP and max pooling to represent SAs interactions. Zhu *et al.* [259] proposed employing DVAE to learn behavior distributions. The authors also benefited from stochastic variables and an MLP on top to consider multimodality in their model, namely RCPNet. Lee *et al.* [266] designed a novel approach, called Desire, that addressed multimodality by extracting diverse samples from potential future behaviors through a CVAE, where it learned the behavior distribution conditioned on the motion and scene attributes using a latent variable. Cheng *et al.* [139] introduced DCENET that incorporates a CVAE structure to address multimodality. The authors employed fully connected layers followed by LSTM to generated different predictions. Salzman *et al.* [263] overcame multimodality by utilizing a CVAE which generates the future behavior distribution through latent variables. Their model exploited discrete variables to better understand the inherent behavior. Huang *et al.* [261] introduced Multimodal Transformer that took advantage of multihead attention to model the uncertainty. It directly exploited the outputs of heads on which the final trajectories of TAs are conditioned. Janjoš *et al.* [260] introduced StarNet that used graph attentions and Multi-head Attention (MHA) for encoding the

interactions among the agents and a GRU-based decoder to estimate agents' kinematics and behavior modes. Nevertheless, these models suffer from inherent limitations such as their need for several forward passes, their real-world suitability due to stochastic sampling and randomness, their interpretability, and not guaranteeing the variety in the produced samples.

Maneuver Prediction

A clear understanding of driving behavior at higher levels and improving robust ADASs and working toward the development of automated technologies and vehicles require not only an understanding of the environmental factors such as road and traffic conditions but also effective approaches and frameworks to automatically detect low-level driving events. Driving events are those related to driving maneuvers such as left/right turns, U-turns, left/right lane changes, driving straight, stopping at intersections, braking, acceleration, etc. Maneuver detection and prediction ⁴ involves anticipating the future movements, actions, or behaviors of agents (such as vehicles, pedestrians, or robots) in dynamic environments. It aims to forecast how agents will navigate, change lanes, make turns, or respond to various situations. Maneuver prediction can be broadly used for many real-world applications including those aimed at automated vehicles, traffic safety, driving monitoring, fleet management, or driver profiling for usage-based insurance. Detecting driving events and maneuvers can also serve as a preliminary step in other tasks of DBA such as driving style detection [31]. If the detected driving events are properly clustered into harsh or risky maneuvers, this information can be then applied to detect aggressive and risky driving styles among drivers, respectively. Recent DBA-based approaches for maneuver prediction are surveyed in the following.

Longitudinal and lateral accelerations are among commonly used features for driving event detection due to their being easily accessible and contributing ability to this task. Sun *et al.* [30] collected data from existing accelerometers and gyroscopes in smartphones. Employing an ensemble of bagging trees and dynamic time wrapping, the authors reached an accuracy of more than 97%. Some studies focus on distinguishing between normal driving events and harsh or sudden ones such as normal lane changes or sudden lane changes. Saiprasert *et al.* [267] developed a system to detect twelve different types of events six of which are sudden movements such as sudden turns, lane changes, braking, and acceleration. An accuracy of 100% for a relatively small dataset was achieved. Li

⁴Also known as driver intention detection and prediction.

et al. [136] designed a system to detect unusual driving events such as sudden braking and swerving. In their system, the amplitude and derivatives of acceleration and gyroscope reading, as well as the duration of the event, have been chosen to characterize sudden movements. In addition to acceleration, the steering wheel angle and the vehicle's speed are also related to the way a driver controls the vehicle's orientation. Zheng and Hansen [111] leveraged KNN and HMM models to detect right/left lane changes based on speed and steering angle signals. Their experimental results showed an accuracy of more than 80%. Xie *et al.* [268] investigated the effect of different types of feature extraction on driving event detection performance over three datasets. Extracted features using three different techniques including statistical features, PCA, and stacked sparse auto-encoders were experimentally compared. The result indicated that statistical features were more appropriate for this task.

Remarkable advances have been made toward the development of ADASs to assist drivers in traffic safety. Modeling driver behavior and understanding the intention of a driver prior to the actual occurrence of driving events can help enhance the existing ADASs in terms of vehicle control or collision avoidance. Therefore, there exist several studies in the literature that make efforts to predict the future driving events/maneuvers a few seconds before they actually happen through analyzing data from the vehicle, head position, physiological signals, gaze direction or their combination [34, 35, 37, 269–271]. Das *et al.* [34] took advantage of integrating statistical and environmental features such as roadway characteristics, speed, acceleration, yaw rate, etc. to predict right and left lane changes 5 seconds prior to the actual maneuver with the accuracy of almost 96%. The authors in [35] proposed to infer the intention of drivers by modeling the temporal dependencies in time series data by LSTM. Combining drivers head poses and gaze directions with vehicle speed and steering wheel angle, they predicted five different events 3.6 seconds before.

In brief, maneuver-based approaches attempt to recognize the intention of the nearby vehicles (e.g., turning left or changing the lane) before its actual occurrence [272–274]. These approaches assume that the vehicles' future behavior is close to the predicted maneuver class. Earlier works used classical ML approaches, including but limited to Bayesian network [275], SVM [113], and HMM [276] to analyze vehicle status-related data such as speed or steering angle to anticipate the driver intention. However, more recent methodologies benefited from deep learning models as they have proven to outperform classical ML models in many applications. In this direction, Khairdoost *et al.* [35] developed a prediction model using LSTM to anticipate drivers' future maneuvers. This

method used both gaze positions and vehicular sensor data to predict lane changes and turns less than 4 seconds before they actually occur. Moreover, as mentioned earlier, driver intent prediction at unsignalized intersections is more challenging [277] since there are no traffic lights or multiple lanes in the suburban areas. There have been some efforts in this direction. Bender *et al.* [278] benefited from the Quadratic discriminant analysis model to classify the maneuvers at a T-junction. The authors collected a naturalistic driving dataset for the evaluation. Cross-validation results achieved a belief of about 0.6 and an accuracy of approximately 60% for a vehicle moving toward the west located at 40 meters before the center of the intersection. In another work, Zyner *et al.* [279] introduced a deep learning-based driver intent prediction approach taking advantage of RNNs. The authors designed a three-layer LSTM to analyze a short interval of trajectory data collected from the target vehicle. An accuracy of about 65% was reached for vehicles approaching from the east direction located at 40 meters before the center of the T-junction. Yi *et al.* [280] proposed an approach to reduce the need for manual maneuver labeling before the training process. They used the PRM clustering model to extract different types of maneuvers. Then, a regression tree is employed to classify the maneuvers. One limitation of such a method is the need to determine the optimized number of clusters which can vary for different scenarios. One shortcoming of these approaches is that they only present the intended maneuver of the drivers, which is a high-level understanding of the behavior the target vehicle is going to take. To provide more detailed behavior, we are required to define more classes/sub-classes. Another shortcoming is that the types of maneuvers hinge upon the environment the vehicle is driving in, such as a highway or a T-junction.

2.5 Concluding Remarks

Driving behavior analysis holds the potential to improve the performance of drivers, road and traffic safety, fuel management, and designing effective intelligent transportation systems such as driver monitoring systems and ADASs. This chapter surveyed the recent research and development efforts that have been made toward DBA. We categorized the most recent studies on DBA according to the main data sources employed for the analysis, major models and algorithms used for the analysis, and the main purposes of the analysis. Also, we provided the strengths and constraints of the current approaches. The existing challenges of different tasks of DBA are also discussed.

Part I

Driver Identification and Verification for ADAS

Chapter 3

Driving Behavior Identification through CAN-BUS Data

Modern vehicles can collect several hundreds of sensor data through CAN-bus technology that provides almost real-time information about the vehicle and the driver. These data can be later processed and analyzed to offer efficient solutions and insights for driving behavior analysis and further applied in a variety of fields such as accident prevention, driver profiling, or vehicle energy consumption. Additionally, as deliberated in Chapter 2, the identification of drivers through analysis of their driving behavior has emerged as a potent and non-intrusive technique. This identification potential holds promise for developing driver profiles and reinforcing vehicle security measures, including anti-theft systems.

In this chapter, we assess the performance of data-driven end-to-end models designed for identifying driving behaviors and provide a critical analysis of the underlying design principles of these models. By doing so, we aim to provide insights that can contribute to the further refinement and design of models for driver identification. To achieve this, we implement several machine learning and deep learning models. While some of the comparison models have been introduced to serve the driver identification purpose [21, 71, 281], their performance in terms of needed data pre-processing, feature engineering, and 10-fold cross-validation (10-CV) results has not been discussed in the previous studies, which is the main objective of the present chapter.

3.1 Introduction

The continued development of Internet of Vehicles and communication technology has extensively affected the evolution of vehicles in recent years, making them smarter and more connected. Modern vehicles are now equipped with several in-vehicle interconnection systems and sensors [9, 160] that offer safety, control and infotainment in addition to guaranteeing the efficient functioning of the mechanical components. Notably, nowadays, obtaining massive and almost real-time information on the cars and the drivers is not exceedingly difficult, and data availability is not an obstacle anymore. These massive data are collected from On Board Diagnostics (OBD-II) protocol through CAN-bus which enables separate electronic control units in cars to communicate. With CAN technology being implemented in cars as well as the existing advanced GPS-based technologies, petabytes of data are collected from cars [189] that can be further processed, analyzed, and be beneficial to a wide range of problems and applications some of which are driving behavior analysis, maneuver recognition, driving style detection (aggressive or Eco-friendly), fuel consumption reduction, driver identification, automated vehicles, driving risk assessment, and fleet management.

Human driving behavior is a complicated concept of drivers' decisions and the way the driver controls the car in various conditions [76]. Analyzing the abovementioned behavior involves characterizing the driver's behavior to serve a specific purpose (e.g. driving style detection) that can play a pivotal role in achieving solutions for major problems such as designing driving models for self-driving cars, accident prevention, and risk assessment for insurance companies. Moreover, driving experience is getting more and more sophisticated and comfortable with the help of data obtained from connected vehicles. Nevertheless, this new level of connections has brought up unprecedented security concerns including hacking and car theft [189]. To alleviate this problem, it has been suggested that the drivers should be identified using the unique patterns of execution and the strategies to drive called the driver's fingerprint [17]. Consequently, driver identification, as an efficient and reliable method, has recently gained significant attention.

Driver identification is of utmost importance for car security concept in the context of connected vehicles, especially with the advent of urban mobility revolution, car sharing, and shared mobility. Focusing on this matter, while other authentication methods such as passwords, fingerprints, and face or voice recognition may not be advantageous for driving profiling, a better alternative is to identify the driver through behavioral anal-

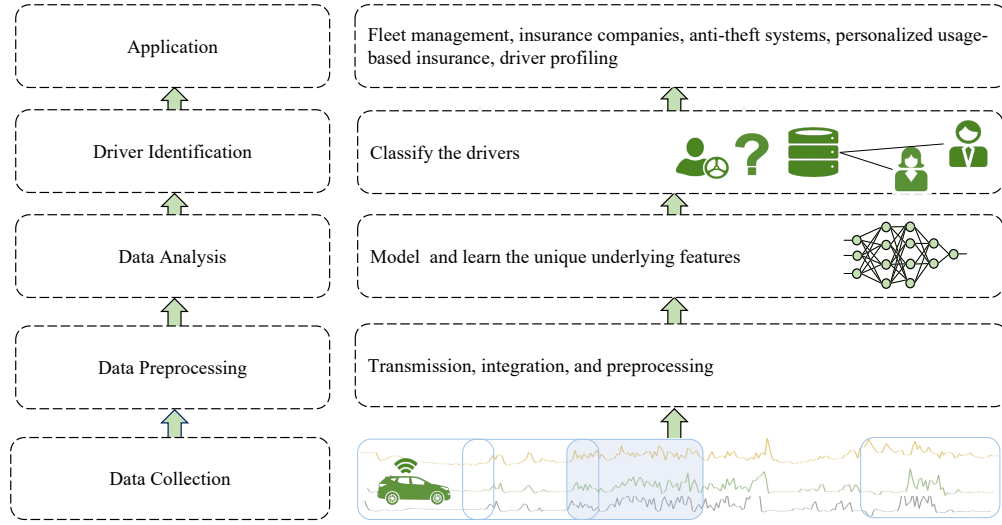


Figure 3.1: Non-intrusive driver identification framework: an illustrative example.

ysis. Taking the uniqueness of the driving behavior into account, a framework can be designed to process, study, and analyze the data obtained by CAN-bus and identify the driver. An illustrative example of driver identification framework design employing the data obtained from connected vehicles is presented in Fig. 3.1. As illustrated, various types of data are collected from the vehicle and then transmitted for further analysis. The shown driver identification framework outlines a step-by-step process for accurate driver identification. It starts with data collection, where various sensor data is gathered through technologies like the CAN-bus, offering insights into the vehicle, surroundings, and driver behavior. Next, these data undergo preprocessing, including cleaning and feature extraction, for thorough analysis. Advanced techniques are then employed to unveil distinct behavioral attributes of each driver. The core step involves driver identification, where data-driven models, such as deep learning and machine learning-based ones can be used to decipher driving patterns. Finally, the results can be beneficial for real-world applications, such as driver profiling, and vehicle security. It is noteworthy that once a driver is identified, an authenticated driver can access the personalized user interface, whereas an unauthenticated one can trigger an alarm for the real driver or the monitoring company.

The remainder of this chapter is organized as follows. Section 3.2 discusses the exploited dataset and the steps toward data processing and feature engineering. The models employed in evaluation are also described in this section. Section 3.3 represents

the performance evaluation of the models and the obtained results. Section 3.4 provides helpful insights to design new models. Eventually, Section 3.5 concludes the chapter.

3.2 Methodology and Background

In this section, we provide a description of the used dataset in experiments. We present the exploratory data analysis (EDA) results and the methods employed for preprocessing, feature selection, and normalization. We also briefly describe the classification methods that are exploited in this section.

3.2.1 An Overview of the Dataset

We have evaluated the classification models in this chapter on a public naturalistic driving behavior dataset [71]. This dataset, which is collected from KIA Motors vehicles using in-vehicle OBD-II devices with a sampling rate of 1 Hz, comprises trip records from 10 drivers labeled 'A' to 'J'. The data were collected at the same time of the day on the same paths with 46 km in total. Furthermore, it is worth mentioning that we did not have direct access to DataBase Container (DBC) files, and these files were not publicly available; only the extracted signals were accessible. There exist 51 features in the dataset each of which provides some information about the engine, fuel or transmission such as speed, fuel consumption, steering angle, accelerator, and engine speed. As an illustration, Fig. 3.2 represents a sample feature visualization of wheel velocity of drivers B and J as well as fuel consumption of drivers A and F while driving the same type of road.

Commonly, the raw data available in the datasets are not suitable to be used in subsequent steps. Therefore, preliminary processing steps such as data cleaning, normalization, and feature selection are required. Data normalization is done to unify the scales of all features each of which has a different range such that the classification models treat them equally. Let $D \in \mathbb{R}^{F \times S}$ be the defined input dataset where F is the dimensionality of the dataset and S is its size. The normalized values are calculated through Eq.3.1 where $mean(d^f)$ and $std(d^f)$ are the mean and the standard deviation of the f^{th} dimension of D , respectively.

$$D_i = \frac{d_i^f - mean(d^f)}{std(d^f)} \quad (3.1)$$

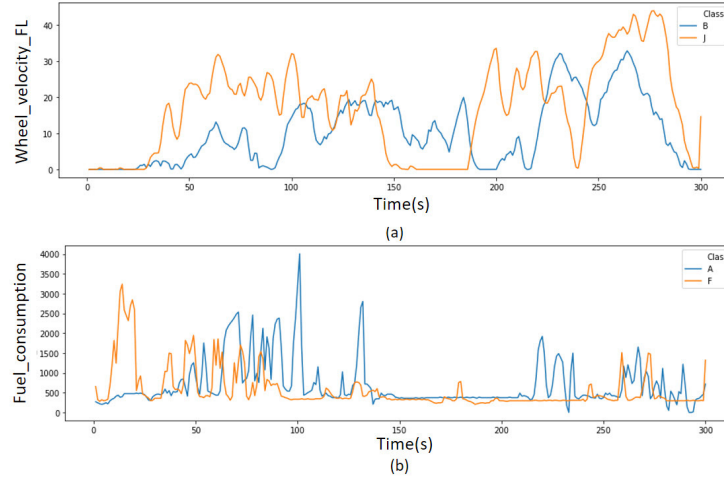


Figure 3.2: A sample feature visualization for two groups of drivers.

In the data preparation phase, we employ the sliding window technique to transform CAN-bus data into a structured format suitable for driver classification tasks. Recognizing that CAN-bus data represents a time series, where each data point holds sequential information, the sliding window method becomes essential for proper data segmentation. By applying sliding windows, we convert the continuous stream of CAN-bus data into fixed-size instances, addressing the temporal nature of the signals. This technique establishes a sequence of fixed-length segments, each serving as an instance for driver classification. The fixed window size ensures consistency in input dimensions, aiding the training of machine learning models. Furthermore, sliding windows enable the extraction of relevant features by sequentially analyzing distinct segments of the signals, cultivating a localized understanding of driving maneuvers, turns, accelerations, and decelerations. This approach acknowledges the sequential nature of driving behavior, allowing the model to learn temporal dependencies and adapt to dynamic variations. Consequently, the time series data are divided into continuous data segments employing a fixed window size and a time step (i.e., the number of samples the window is shifted over the data). As shown in Fig. 3.3, sliding window analysis employs overlapping windows to better capture the progressive nature of the time series data and to extract the underlying features. This approach enhances models effectiveness by ensuring a continuous and fine-grained capture of temporal dynamics. It improves sensitivity to subtle changes in driving behavior, reduces information loss at window edges, and cultivate a more robust representation of diverse driving styles. The overlapping windows contribute to a more adaptable and re-

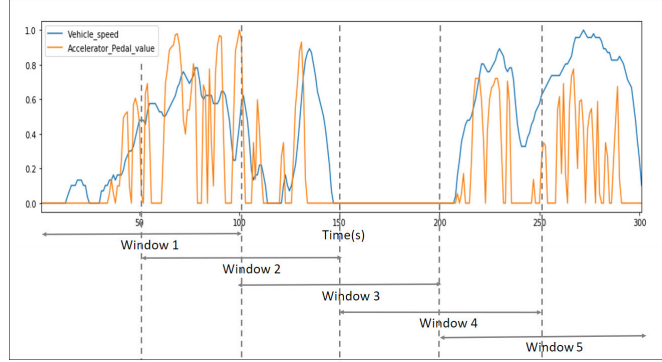


Figure 3.3: Sliding window with the window size of 100 and the time step of 50.

sponsive model, optimizing its performance in recognizing patterns that may span across multiple time frames and enhancing its overall efficacy in driver classification tasks. As an example, the window size is set to 100 seconds and the window is shifted by 50 seconds each time. Therefore, instead of having 3 segments of size 100 for 300 seconds data, the data are divided into 5 segments exploiting a 50 seconds sliding window. It is worth mentioning that since the data collection frequency rate is 1 HZ in the dataset, a window size of 100 seconds contains exactly 100 data points.

For the performance evaluation of the models, we have taken account of different feature selection methods. The first one is to use all 51 original features existing in the dataset to highlight the power of deep learning methods in extracting hidden features in comparison with the classical machine learning algorithms. Secondly, we conducted correlation analysis to pinpoint the most influential features closely linked to drivers behavior. The application of correlation analysis enabled us to identify and retain the most relevant features by examining their relationships with the target variable. This method proved valuable in eliminating redundant or highly correlated features, ensuring that the selected features contribute distinct information to the model, thus enhancing its interpretability and performance.

Figure 3.4 represents the Pearson product-moment correlation coefficient after feature selection. The values assess the linear relationship between any two continuous variables. As shown, the highly correlated features have been excluded and the 15 remaining ones are less correlated. The values less than 0 indicate the negative correlation between the variables. Moreover, we extract statistical features in order to have an effective feature characterization and reduce the effect of any existing fluctuations in the value of the

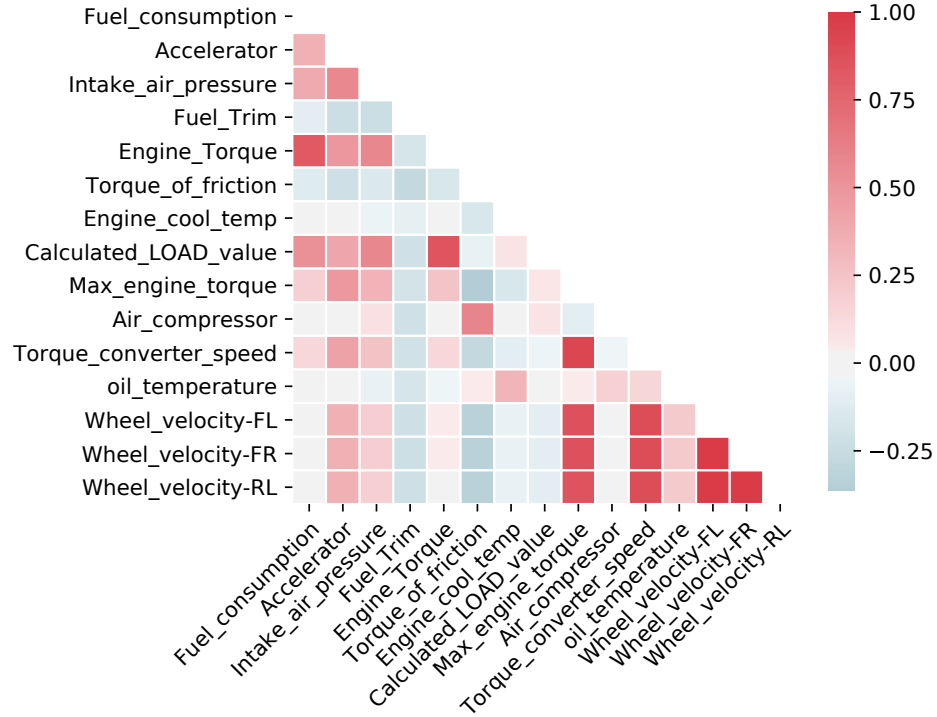


Figure 3.4: Exploring feature correlations: Pearson correlation coefficient analysis for selected features.

features. For each feature in the dataset (the 15 remaining ones after feature selection), statistical features including the mean, standard deviation, and median were extracted employing a window size of 60 and were added to the dataset. As a result, in this method, we have 15 features and 45 statistical features in total.

3.2.2 Driver Identification Classification Models

As mentioned earlier, we can consider CAN-bus driving data as time series since there are dependencies between the data with respect to the time. Time series classification problems can be categorized into two groups namely univariate and multivariate time series. The former considers one variable, whereas the latter involves more than one variable. Identifying the drivers employing extracted sensor data from the car is a multivariate time series classification problem, and since there are 10 drivers in the used dataset, it is also a multi-class classification problem. The classification models implemented and

evaluated in this chapter are briefly discussed in the following.

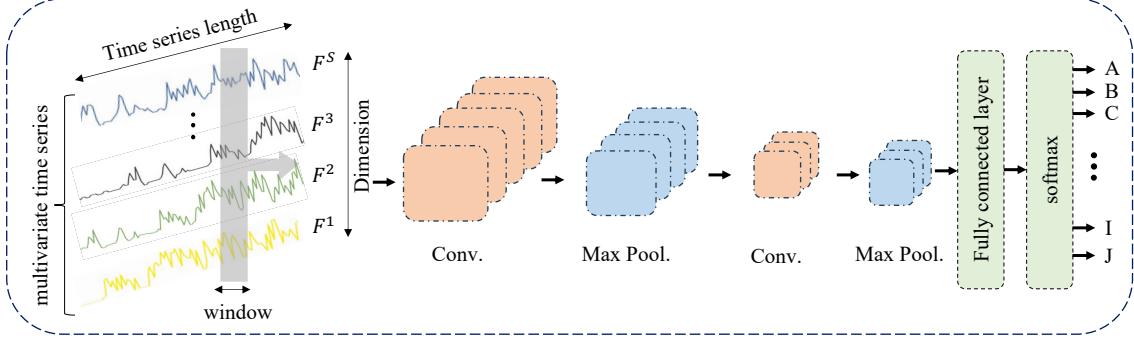


Figure 3.5: Schematic diagram of the CNN model containing two convolutional and maxpooling layers.

CNN. Although CNN models, as an important type of neural networks, are not specifically designed to capture the time dependencies of the sequential data, they have shown efficient results for time series classification [282]. We have implemented a CNN model containing two convolutional and max pooling layers followed by a fully connected layer and a softmax one for the output classification. Figure 3.5 presents the architecture of the model. As can be seen, the multivariate time series input data are processed through the sliding window technique and fed to the CNN model. The feature map output in each layer is calculated by Eq.3.2:

$$X_k^{(l)} = f\left(\sum_{i=1}^{n_{l-1}} W_{ik}^{(l)} * X_i^{(l-1)} + b_k^{(l)}\right) \quad (3.2)$$

where f is the activation function, i.e., ReLU function, $X_i^{(l-1)}$ is the output of i^{th} neuron at layer $l - 1$, $W_{ik}^{(l)}$ is the filter mapping, and $b_k^{(l)}$ is the bias.

DNN. As another basic model, we have used a DNN architecture. We employed stacked of fully connected layers. There are two hidden layers in our used baseline architecture. It is worth noting that, in our methodology, we did not feed the raw input signals directly into the DNN model, since DNNs require prior feature extraction. Instead, we chose to use the same set of extracted statistical features, comprising mean, standard deviation, and median, for classical machine learning approaches. This decision was motivated by a desire for simplicity and computational efficiency. By relying on statistical features, we aim to provide a summary of the distribution and characteristics of the data, allowing for an efficient and interpretable representation in the models.

LSTM. As discussed in Chapter 2.3, RNNs, as a major deep learning-based models, serve as a pivotal tool for effectively managing sequential data. Their distinctive capability to capture intricate temporal patterns within data makes them particularly suited for such tasks. However, RNNs main challenge is to capture long-term dependency as a result of their gradient vanishing/exploding problem. This limitation has been effectively addressed by the introduction of LSTM recurrent units [283]. We have implemented an LSTM-based recurrent network, including an LSTM layer and two fully connected layers followed by a softmax layer. LSTM is modeled through Eq. 3.3:

$$\left\{ \begin{array}{l} i_t = \sigma(W_i[a_{t-1}, x_t] + b_i) \\ f_t = \sigma(W_f[a_{t-1}, x_t] + b_f) \\ o_t = \sigma(W_o[a_{t-1}, x_t] + b_o) \\ \tilde{c}_t = \tanh(W_{\tilde{c}}[a_{t-1}, x_t] + b_{\tilde{c}}) \\ c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \\ a_t = o_t \odot \tanh(c_t) \end{array} \right. \quad (3.3)$$

where W is the weights, c is the cell state, and i , f , o are the input, forget, and output gates, respectively.

DeepConvLSTM. DeepConvLSTM [21] model contains two separable convolutional layers, and a pooling layer, followed by LSTM layer. This architecture combines CNNs and LSTM units. This hybrid approach leverages the strengths of both CNNs and LSTMs to effectively process and analyze sequential data. By incorporating convolutional layers, the model can capture spatial patterns and features within the data, while the LSTM units enable the model to capture temporal dependencies and long-range interactions.

Classical machine learning algorithms. As baseline, three classical machine learning methods including random forest, decision tree and KNN are used, which are shown to be efficient in driver identification [71, 281].

It is noteworthy that all deep learning models are implemented using Adam optimizer with the dropout of 0.5.

3.3 Performance Evaluation

In this section, we evaluate the performance of seven different models, mentioned in Section. 3.2.2, with respect to their ability to capture the temporal dependency in driver identification time series, the required feature selection, and classification performance. We have used the standard 10-fold cross validation technique for classification results such that, in each round, the data are split into 10 partitions/folds. Then, the models are trained based on the nine folds and tested on another remaining fold. For evaluation metrics, we employed two metrics:

- **The classification accuracy.** This metric measures the proportion of correctly classified instances out of the total instances in the dataset.
- **The weighted F1 score.** The F1 score is a measure that takes into account both precision and recall of a model results. It provides a balance between precision (i.e., the proportion of true positives among predicted positives) and recall (i.e., the proportion of true positives among actual positives). The weighted F1 score considers the F1 score for each class and calculates a weighted average based on the class distribution in the dataset.

Table 3.1 presents the 10-CV results for all seven models as well as their variations. The mean and the standard deviation of both multi-class classification accuracy and weighted F1 scores are shown employing the window size of 60. As can be seen, when feeding all features to the methods, classical machine learning models perform poorly since they are not able to learn the non-linear complex temporal dependency existing in the data. Their performance, however, improves by over 20% when statistical features are added. Conversely, the deep learning methods including CNN, and DeepConvLSTM gained better performance without any feature selection by CNN obtaining the highest accuracy and F1 score. Of particular significance among the outcomes is the outcome of LSTM model. It is noteworthy that when utilizing the complete set of features for the LSTM architecture, the achieved accuracy and F1 score were notably lower compared to employing a subset of features. This arises from the fact that employing all features in the LSTM model attempts to capture every temporal correlation existing between the features. However, in doing so, the model becomes overly complicated and unnecessarily complex.

Figures 3.6 and 3.7 illustrate the training accuracy and loss for deep learning models employing all 51 features. We present one iteration of 10-CV results in the figures for

Table 3.1: Evaluation results (window size = 60).

Model	Feature Set	Accuracy	F1 score
Random forest	all features	72.12 ± 0.64	72.86 ± 0.85
KNN	all features	67.22 ± 0.79	67.23 ± 0.32
Decision Tree	all features	69.11 ± 0.49	69.86 ± 0.45
Random forest	15 + statistical	90.05 ± 0.88	91.51 ± 0.08
KNN	15 + statistical	89.77 ± 0.83	89.34 ± 0.42
Decision tree	15 + statistical	85.63 ± 0.61	86.98 ± 0.19
CNN	all features	96.85 ± 0.43	96.78 ± 0.38
DNN	all + statistical	94.74 ± 0.34	94.53 ± 0.23
LSTM	all features	89.63 ± 0.51	89.56 ± 0.21
DeepConvLSTM	all features	95.06 ± 0.68	94.92 ± 0.72
LSTM-15	15 features	92.46 ± 0.25	92.83 ± 0.10

clarity reasons. As can be observed, the window size of sliding window technique has a moderate impact on the performance of deep learning methods. However, for both window sizes of 45 and 60, CNN and DeepConvLSTM exhibit better generalization capability, learning the local temporal dependency in driving behavior data. Furthermore, LSTM model shows slower learning process than other methods mainly because of its higher number of parameters. Comparing the test results shown in Table 3.1 and training accuracy and loss in Fig. 3.6, we can conclude that overfitting problem is avoided since we have employed dropout, i.e., a model averaging technique designed specifically for neural networks.

3.4 Discussion

The results observed in the previous section can be used as a guide to design novel models for the driving behavior identification task. From the obtained 10-CV results for different types of feature sets, we can conclude that blindly employing deep learning models does not guarantee better performance in terms of accuracy and F1 scores (please refer to random forest and LSTM results in Table 3.1). It should be noted that making the models more complex may not always yield higher accuracy and can even have reverse

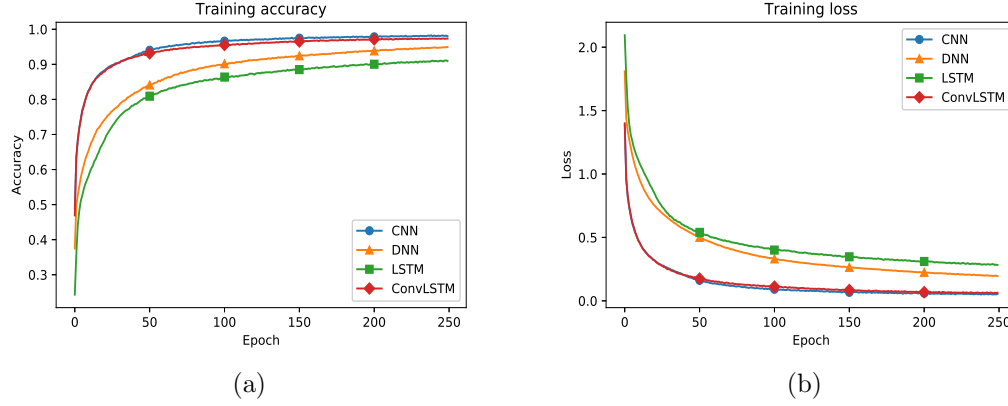


Figure 3.6: The training accuracy and loss (window size = 60).

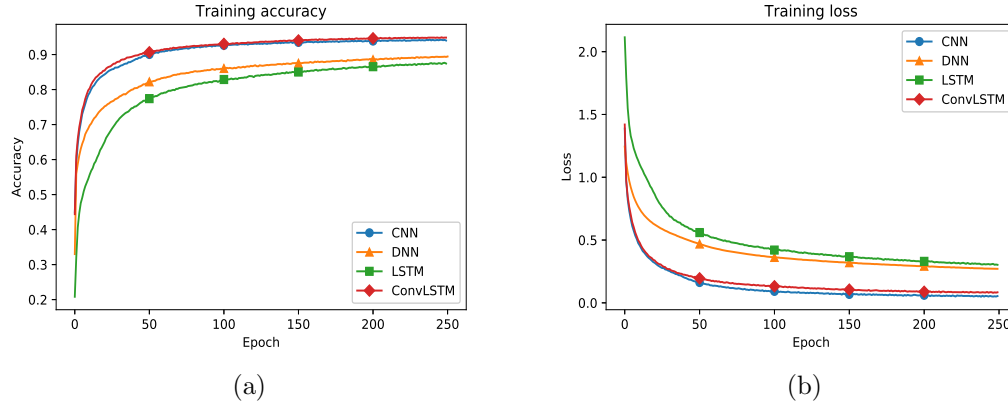


Figure 3.7: The training accuracy and loss (window size = 45).

results (e.g., LSTM with all 51 features). Furthermore, although the sliding window technique is promising, one may find it challenging to find the appropriate window size. Modern methods can be designed to explore the proper window size.

Moreover, overfitting can be always a problem while conducting the driver classification. The dropout technique [284] can be used to reduce the impact of over-emphasizing a single input dimension of layer l . Finally, there must be a trade-off between the amount of time spent on data pre-processing steps and the identification time since in situations such as car theft, real time analysis of driving behavior is essential.

3.5 Concluding Remarks

This chapter evaluated the performance of different deep learning and machine learning methods that have been designed for the driver classification problem. In contrast to the previous works, the performance was done with respect to the capability of the models in capturing the temporal dependency in driving time series data, the 10-CV results, and the required data processing. The evaluation results revealed that feature engineering and exploiting the statistical features for classical machine learning methods is essential. Furthermore, the results indicated that the window size in the sliding window technique has a moderate impact on the performance of models and merely using this technique does not guarantee high performance. We also provided insights for the design of new models for the driver identification task.

Chapter 4

DriverRep: Driver Identification through Driving Behavior Embeddings

In the preceding chapter, we evaluated the effectiveness of various classical machine learning and deep learning techniques for the driver identification task using a naturalistic driving dataset. As discussed, there has been a growing interest in non-intrusive identification methods that focus on analyzing driving behavior to uncover individual patterns. These supervised approaches, however, require a considerable volume of labeled data to have enhanced performance. Therefore, more cost-effective, accurate, and reliable models are needed to make practical and almost real-time applications possible.

To alleviate this problem, in the current chapter, we introduce a novel semi-supervised deep learning architecture, namely DriverRep, to extract the latent representations associated with each individual, called driver embeddings, which represent the unique driving characteristics of drivers. To this aim, we propose a fully unsupervised triplet loss that selects triplet samples from data in an unsupervised manner and extracts the embeddings using our proposed stacked encoder architecture. Dilated causal convolutions are used to make residual blocks of the encoder. To perform the task of driver identification, we leverage the classification accuracy of support vector machine on top of the obtained driver embeddings.

To the best of the authors' knowledge, this is the first work to propose a novel methodology to compute the triplet loss in an unsupervised way from the driving data and, consequently, to extract the driver embeddings through unsupervised representation

learning for the tasks such as driver identification and verification. Such methodology will reduce the necessity of having access to a considerable volume of labeled data for distinguishing the drivers. Moreover, the proposed methodology falls into the category of generative models in which including a new driver in the identification scheme depends only on the collected information from him/her. There will be no need to update the entire model.

The remaining of this chapter is structured as follows. In Section 4.1, we present our proposed method, namely DriverRep, for the task of driver identification. Performance evaluation and discussion are given in Section 4.2, followed by final remarks in Section 4.3.

4.1 The Proposed Model

In this section, we first describe the preliminaries used to formulate and solve the problem. We then explain the proposed methodology, namely DriverRep, and its deep neural network architecture.

The main objective of the DriverRep system is to identify the person controlling the vehicle based on their driving patterns using driving behavior analysis. This way, we emphasize the fact that the data based on which we define our model are extracted from sensors such as smartphones, IMUs, wearables, or GPS. We consider sensor data as time series, and we aim to determine the best mappings from the extracted features to the drivers. To formulate the task of driver identification, one can consider a collection of pairs $\{(I_1, L_1), (I_1, L_2), \dots, (I_N, L_N)\}$, where I_i and L_i are driving instances and labels (i.e., drivers), respectively. Also, each driving instance can be of type univariate or multivariate according to the number of extracted features. In most driving behavior analysis scenarios, we are faced with multivariate instances ($I_i = \{i^1, i^2, \dots, i^m\}$ with m being the number of dimensions/features) since we have various types of features such as vehicle speed, acceleration or brake pedal pressures, steering wheel angles, etc.

An overview of the DriverRep and its model architecture is illustrated in Fig. 4.1. Based on the definitions mentioned earlier, the main task of DriverRep includes analyzing the driving time series data and finding the best mapping from the features to the drivers through determining the underlying behavior representations for each driver. The DriverRep system can be divided into three phases, including data acquisition from different drivers, latent representation or embedding extraction from the data, and finally, using the obtained embeddings for driver classification task to identify the drivers. We explain the main parts of the DriverRep in the following.

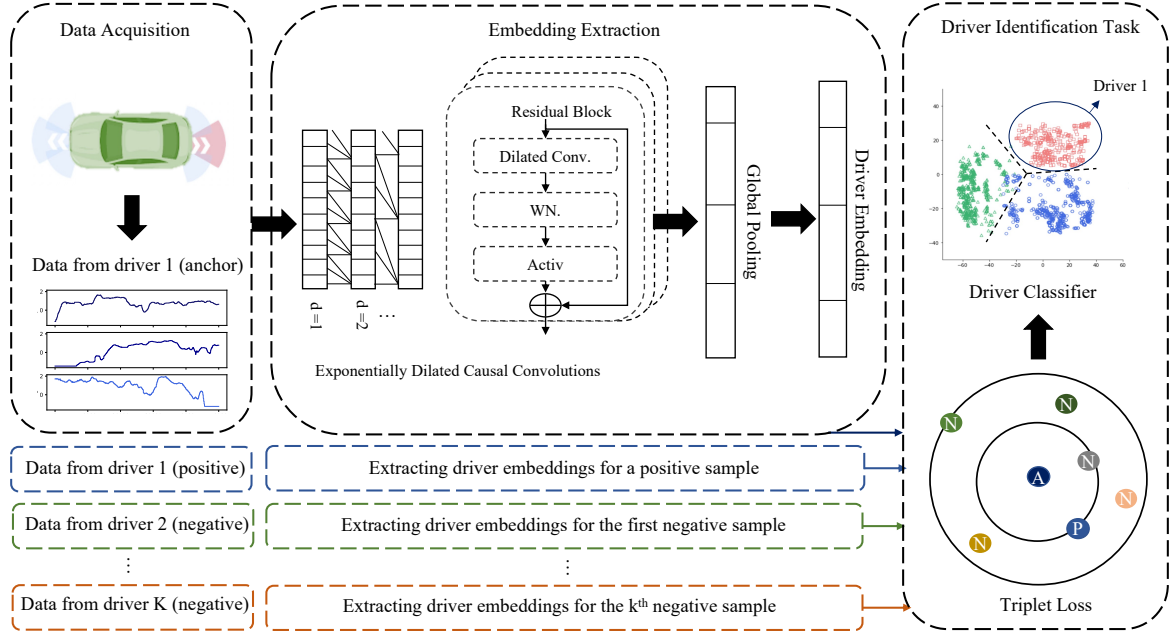


Figure 4.1: An overview of the proposed driver identification system, called DriverRep, and its model architecture.

4.1.1 Driver Embeddings

Embeddings or latent representations are, in their simplest form, multidimensional vectors that provide insights into the essential features of the data. If properly extracted, embeddings can also be advantageous for distinguishing data instances based on the distinctive underlying features within the data such as various faces in the face recognition task. In the DriverRep model, we tend to extract these latent representations for various drivers in order to gain valuable insights about their unique driving behavior and patterns which are further used to identify the person behind the wheel. The main advantages of using embeddings are that they can be transferable, and it is not necessary to update the entire model when a new class (drivers in our case) is added to the system. It is only required to compute the embeddings for the newly added driver. Inspired by [285, 286], we propose to compute an unsupervised triplet loss metric for driving time series for better capturing the underlying features. The concept of triplet loss assumes that similar data will present similar embeddings as well. In fact, triplet loss is used to bring the embeddings of the same instances closer and push farther the embeddings of different instances. Fig. 4.2 demonstrates an illustrative example of triplet loss training and its

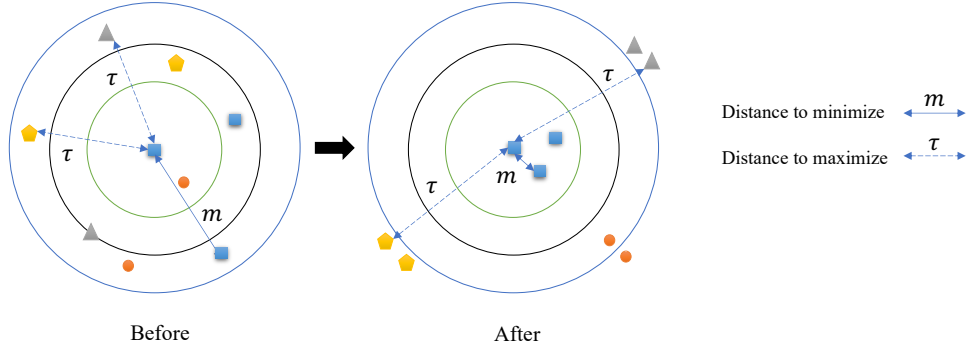


Figure 4.2: The triplet loss training illustrative example.

primary purpose, which is to make the samples of different instances far away from each other. As shown, after the training process, the distance between similar instances has decreased.

On the one hand, our main objective in this section is to make sure that similar driving time series gain similar driver embeddings without any supervision to learn these similarities so that we can tackle the challenge of sparse labeling in the driver identification task. On the other hand, although the concept of triplet loss helps to obtain such similarities [178], it requires that we provide pairs of similar and dissimilar inputs, which challenges unsupervised training. While the existing approaches for driver identification, which work based on supervised triplet losses ([164, 181]), consider the driving data is labeled, we propose an unsupervised time-based approach to choose these pairs that also consider the variable length of time series. The proposed triplet loss training is considered unsupervised as the labels have no participation in the training process.

Triplet loss requires tuples of triplet samples, namely *anchor*, *positive* and *negative*. The reason is to provide the system with pairs of instances that are similar (i.e., the *anchor* and the *positive* samples) as well as pairs of instances that are not (i.e., the *anchor* and the *negative* samples). As discussed earlier, this task is challenging in an unsupervised setting. Hence, in DriverRep, we propose to choose these triplet samples in an unsupervised manner based on a sub-sampling strategy. As described in Algorithm 1, considering a dataset of driving time series, we start by selecting one of the instances I_n which belongs to a specific driver. At this point, we choose a subsequence (consecutive) of this instance as the *anchor* sample. To choose the *positive* sample, we then select a subsequence of the anchor with a size smaller than the anchor itself. The reason behind this choice is the fact that any subsequence of any given driving instance is more similar to

Algorithm 1: Positive and negative sampling method.

Input: $\{I_1, \dots, I_N\}$: driving instances**Output:** triplet samples for loss calculation

```

1 for each epoch over the driving instance  $I_n$  do
2   for  $n \in \{1, \dots, N\}$  do
3      $s \leftarrow \text{size}(I_n)$ 
4      $s_{\text{positive}} \leftarrow$  a random number  $\in [1, s]$ 
5      $s_{\text{anchor}} \leftarrow$  a random number  $\in [s_{\text{positive}}, s]$ 
6     Select one anchor sample at random from subseries of  $I_n$  with size  $s_{\text{anchor}}$ 
7     Select one positive sample randomly from subseries of anchor with size
        $s_{\text{positive}}$ 
8      $\text{neg} \leftarrow$  desired number of negative sampling
9     for  $d \in \{1, \dots, \text{neg}\}$  do
10      Select  $n_d \in \{1, \dots, N\}$ 
11       $s_d \leftarrow \text{size}(I_{n_d})$ 
12       $s_{\text{negative}} \leftarrow$  a random number  $\in [1, s_d]$ 
13      Select a negative sample randomly from subseries of  $I_{n_d}$  with size
         $s_{\text{negative}}$ 
      end
    end
  end
end
return: All triplet (anchor, positive, negative) samples

```

that given instance rather than those selected from other driving instances. Considering another random time series, we can select a subsequence of that driving instance as a *negative* sample. We also select multiple *negative* samples randomly to improve the performance of the learned driver embeddings as well as the convergence of the loss training. It is important to note that when selecting *negative* samples randomly, there is a possibility of inadvertently including *positive* samples, as the selection process does not take labels into account. Consequently, choosing multiple *negative* samples helps mitigate the likelihood of inadvertently including *positive* samples among the randomly chosen *negatives*. It is worth mentioning that the size of the samples is chosen randomly so that the algorithm is flexible even if the length of all driving instances (i.e., time series) are not the same. However, in the case of having an equal length, one can initialize the

sample sizes such that all of them have the same value to make the training process more efficient. This process is considered unsupervised as the labels of the time series have no role in the process of the sample selection as well as training based on the optimization objective function discussed in the following. We define the optimization objective function according to the equation 4.2 where function f is a deep neural network with θ as its parameters. The overall computational cost is $O(neg.c(f))$, where $c(f)$ is the cost of evaluation and backward propagation through the deep neural network on time series data.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4.1)$$

$$J(\theta) = \log \sigma(f(anchor, \theta)^\top f(positive, \theta)) + \sum_{d=1}^{neg} \log \sigma(-(f(anchor, \theta)^\top d(negative, \theta))) \quad (4.2)$$

As discussed earlier, the task is to distinguish the *anchor* and *positive* samples from the *negative* ones. Figure 4.3 presents a clarifying example for three drivers selected from a real-world driving dataset (see Section 4.2.1). We employ the t-distributed stochastic neighbor embedding (t-SNE) [287] that is a well-established algorithm for the visualization of high-dimensional data. On the one hand, Fig. 4.3a demonstrates the initial driving instances for drivers A, B, and C. On the other hand, Fig. 4.3b visualizes the extracted driving embedding from the same drivers as those in Fig. 4.3a after running the proposed DriverRep architecture for 20 epochs using the proposed triplet loss objective function (See section 4.1.2 for the details on the architecture). The effect of triplet loss training can be observed such that the driving embeddings of the same drivers are closer to each other than to driving embeddings of the other two drivers.

4.1.2 DriverRep Architecture

In this section, we propose a scalable encoder architecture to learn the introduced unsupervised loss. The introduced encoder consists of stacked dilated causal convolutions in the format of deep convolutional neural networks, which accepts the time series with variant length and outputs the targeted latent representations of the drivers. The proposed architecture is motivated by the fact that we tend to capture the underlying features from the driving time series, be efficient and scalable, and be flexible about the length

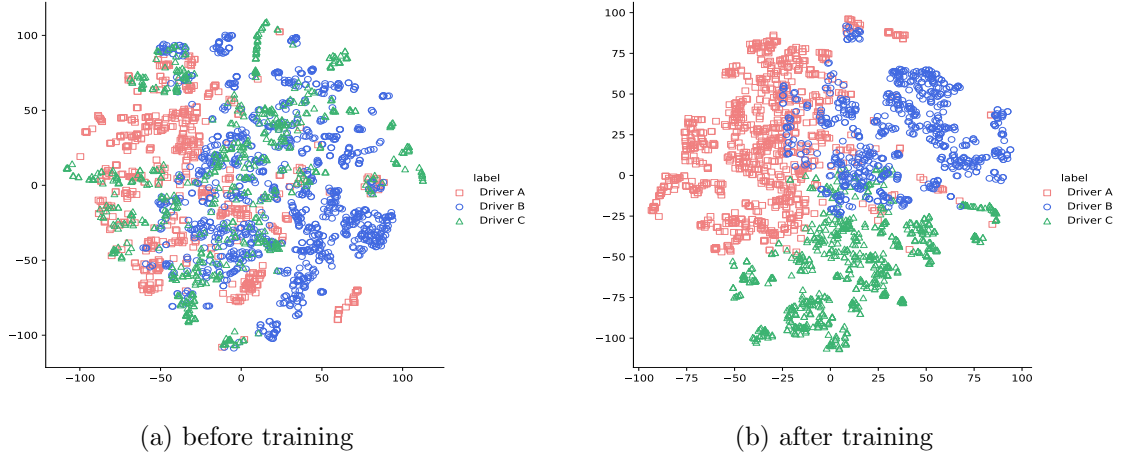


Figure 4.3: Driving instances before training and the obtained driver embeddings after training.

of the driving instances. The main building blocks of the DriverRep architecture are explained in the following.

One-Dimensional Convolutions

CNNs [118] are a type of neural network with the capability of extracting features from data being mostly used for two-dimensional (images) data. However, CNNs can be generalized to the one (1D) or three-dimensional data as well. In the following, we explain this generalization process for 1D convolutions for driving time series data. We start by describing the forward and backward propagation process of the 1D CNN. Given an input signal I and a filter F of size N_f , the convolution operation is given by:

$$(I * F)_i = \sum_{m=0}^{N_f-1} I(i+m)F(m) \quad (4.3)$$

The 1D CNN models include convolutional layers with the inputs and a number of filters and biases. In the case of signals (time series data), we can have an input with the size of N_i and a different number of channels C in the case of multivariate time series. In this situation the input is $I \in \mathbb{R}^{N_i \times C}$. Also, for each filter out of S filters, we have $F \in \mathbb{R}^{N_f \times C \times S}$. The output from such a convolution is:

$$(I * F)_i = \sum_{m=0}^{N_f-1} \sum_{c=1}^C I_{i+m,c} \cdot F_{m,c} + b \quad (4.4)$$

For the sake of argument and simplicity, we consider the case where the input is a univariate signal. Then, Eq.4.4 can be rewritten as:

$$(I * F)_i = \sum_{m=0}^{N_f-1} I_{i+m} \cdot F_m + b \quad (4.5)$$

To explain the forward and backward propagation in the 1D CNN, we use the following notations:

1. l is the number of the layer from 1 to L .
2. input x has the size of N_i (i as the iterator)
3. Filter values are considered as weights w with the size of N_f (m is the iterator)
4. w_m^l is the weight vector connecting neurons of layer l to those of layer $l - 1$
5. b_l is the bias
6. x_i^l is given by:

$$x_i^l = \sum_m w_m^l o_{i+m}^l + b^l \quad (4.6)$$

7. o_i^l is the output vector at layer l computed as:

$$o_i^l = g(x_i^l) \quad (4.7)$$

8. $g(\cdot)$ is the activation function.

In the forward propagation, the filter is solid across the input signal for the convolution operation. The convolution of the input layer at layer l is calculated using Eq. 4.6 and Eq. 4.7.

Moreover, to perform backpropagation, we need to compute the derivatives. First, we have to compute the $\frac{\partial E}{\partial w_{m'}^l}$ which demonstrates how the change in a single component of filter $w_{m'}$ can affect the loss function of the network E . While performing the forward pass in CNN, the convolution process makes sure that the component $w_{m'}$ participates in all the products operations. This means that the aforementioned component has an

effect on all components of the output/feature map. If we consider the filter size as N_f and the input size as N_i , then the feature map size will be $N_i - N_f + 1$. Therefore, the gradient component for each weight can be gained as the following using the chain rule:

$$\frac{\partial E}{\partial w_{m'}^l} = \sum_{i=0}^{N_i-N_f} \frac{\partial E}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial w_{m'}^l} \quad (4.8)$$

We first start with the $\frac{\partial x_i^l}{\partial w_{m'}^l}$ part. If we substitute the x_i^l with Eq.4.6, we have:

$$\frac{\partial x_i^l}{\partial w_{m'}^l} = \frac{\partial}{\partial w_{m'}^l} \left(\sum_m w_m^l o_{i+m}^l + b^l \right) \quad (4.9)$$

We then expand the summation in Eq.4.9 and take the derivatives for all components. As a result, we get the following:

$$\begin{aligned} \frac{\partial x_i^l}{\partial w_{m'}^l} &= \frac{\partial}{\partial w_{m'}^l} \left(w_0^l o_{i+0}^{l-1} + \dots + w_{m'}^l o_{i+m'}^{l-1} + \dots + b_l \right) \\ &= \frac{\partial}{\partial w_{m'}^l} \left(w_{m'}^l o_{i+m'}^{l-1} \right) = o_{i+m'}^{l-1} \end{aligned} \quad (4.10)$$

where the derivatives for all components will be zero except the one where the condition $m = m'$ satisfies.

For now, if we substitute Eq.4.10 in Eq.4.8, we have:

$$\frac{\partial E}{\partial w_{m'}^l} = \sum_{i=0}^{N_i-N_f} \frac{\partial E}{\partial x_i^l} \cdot o_{i+m'}^{l-1} \quad (4.11)$$

Consider δ_i^l as the gradients that will be generated through backpropagation step. Then, these gradients can be denoted as $\delta_i^l = \frac{\partial E}{\partial x_i^l}$, where $\frac{\partial E}{\partial x_i^l}$ shows how the change in a component x_i in the input will impact the loss function E . If we consider the part Q in the output which is affected by the input component x_i^l as a part bounded by $i' - N_f + 1$ to i' , then using the chain rule we have:

$$\frac{\partial E}{\partial x_{i'}^l} = \sum_{i \in Q} \frac{\partial E}{\partial x_Q^{l+1}} \cdot \frac{\partial x_Q^{l+1}}{\partial x_{i'}^l} = \sum_{i \in Q} \delta_Q^{l+1} \cdot \frac{\partial x_Q^{l+1}}{\partial x_{i'}^l}$$

Using the ranges of part Q , we can rewrite Eq.4.1.2 as:

$$\begin{aligned} \frac{\partial E}{\partial x_{i'}^l} &= \sum_{m=0}^{N_f-1} \frac{\partial E}{\partial x_{i'-m}^{l+1}} \cdot \frac{\partial x_{i'-m}^{l+1}}{\partial x_{i'}^l} \\ &= \sum_{m=0}^{N_f-1} \delta_{i'-m}^{l+1} \cdot \frac{\partial x_{i'-m}^{l+1}}{\partial x_{i'}^l} \end{aligned} \quad (4.12)$$

In Eq.4.12, $x_{i'-m}^{l+1}$ is equivalent to $\sum_{m'} w_{m'}^{l+1} o_{i'-m+m'}^l + b^{l+1}$. Therefore, expanding this part we are given:

$$\begin{aligned} \frac{\partial x_{i'-m}^{l+1}}{\partial x_{i'}^l} &= \frac{\partial}{\partial x_{i'}} \left(\sum_{m'} w_{m'}^{l+1} o_{i'-m+m'}^l + b^{l+1} \right) \\ &= \frac{\partial}{\partial x_{i'}} \left(\sum_{m'} w_{m'}^{l+1} f(x_{i'-m+m'}^l) + b^{l+1} \right) \end{aligned} \quad (4.13)$$

Further expansion of the summation and then taking the derivatives results in:

$$\begin{aligned} \frac{\partial x_{i'-m}^{l+1}}{\partial x_{i'}^l} &= \frac{\partial}{\partial x_{i'}} (w_{m'}^{l+1} f(x_{i'-m+m'}^l) + \dots + \\ &+ w_m^{l+1} f(x_{i'}^l) + \dots + b^{l+1}) = \frac{\partial}{\partial x_{i'}} (w_m^{l+1} f(x_{i'}^l)) \\ &= w_m^{l+1} \frac{\partial}{\partial x_{i'}} (f(x_{i'}^l)) \end{aligned} \quad (4.14)$$

Now, if we substitute Eq.4.14 in Eq.4.12, we get:

$$\frac{\partial E}{\partial x_{i'}^l} = \sum_{m=0}^{N_F-1} \delta_{i'-m}^{l+1} w_m^{l+1} f'(x_{i'}^l) \quad (4.15)$$

Dilated Causal Convolutions and Residual Learning

Dilated convolutions are initially introduced as a solution to support receptive fields expansion without coverage loss for tasks such as dense prediction [288]. Furthermore, dilated convolutions are well-suited for sequence modeling because of their ability to consider a longer history. We take advantage of dilated convolutions in DriverRep architecture as they allow the model to efficiently learn the connections from older information in time. We calculate the dilated convolution operator $*_{dil}$ as shown in Eq.4.16 where dil is dilation factor.

$$(I *_{dil} F)_i = \sum_{m=0}^{N_f-1} I(i - dil \cdot m) F(m) \quad (4.16)$$

The dilated factor is incremented exponentially as demonstrated in Fig. 4.4 (right). In other words, the convolution is calculated for every dil element from the input driving instance. Moreover, the used dilated convolutions are considered causal convolutions as there is no information leakage from future data elements.

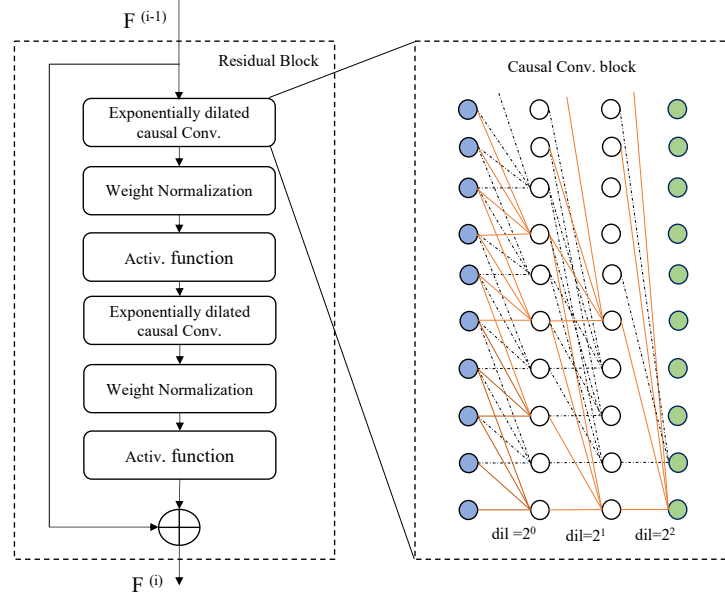


Figure 4.4: (left) Architecture of the i^{th} layer of the DriverRep model. (right) illustration of stacked dilated convolutions.

Furthermore, one of the problems while training deep neural networks is the degradation problem in which the training error becomes higher as the depth of the network grows [289]. The solution is to benefit from residual learning and the concept of skip connections [289] which cause the deep neural network to approximate $F(x) = H(x) - x$ rather than mapping $H(x)$. Skip connections in a residual block simply skip one or more layer(s) and get concatenated to the output of the block.

As can be seen in Fig. 4.4, the DriverRep architecture is based on residual blocks of stacked dilated convolutions. Each layer is constructed as a combination of exponentially dilated causal convolutions followed by weight normalization and the ReLU activation function. The output of the introduced encoder is the driver embeddings which are then fed to an SVM classifier for the task of driver identification. To implement the architecture, we have used the Pytorch library along with the list of the following hyper-parameters: learning rate of 0.001, decay rates of (0.9, 0.999), embedding size of 128, batch size of 10, and the Adam optimizer.

4.2 Performance Evaluation

In this section, we perform a series of experimental evaluations to assess the performance of the DriverRep. The quality of embeddings extracted by the DriverRep is evaluated based on the analysis of the driver identification prediction results. First, we briefly explain the datasets used in this chapter. We then provide a concise description of the designed experiments and discuss the achieved results.

4.2.1 Datasets

We conduct the performance evaluation over two different real-world driving behavior datasets. The HCRL dataset [71] includes 23 hours of data collected from CAN-bus signals through OBD-II for ten participants. There exist three types of data roads, including urban, motorway, and parking lots. All drivers have performed the task in a predefined path. However, no specific instructions have been given to the participants in terms of driving behavior or style. In this dataset, the driving length is approximately 46 km, and the sampling rate is 1 Hz. We employ the features related to driving behavior categorized into groups of steering, speed, fuel, pedals, and acceleration. These features are selected based on correlation analysis to choose the most contributing features that are highly associated with the interaction between the drivers and the vehicles. For example, driver's reactions can be reflected in accelerator pedal values or steering wheel [17]. A sample of this dataset is shown in Fig. 4.5 from driver A and including acceleration pedal value, vehicle speed, fuel consumption, long-term fuel trim bank, throttle position, intake air pressure, wheel velocity, torque, lateral and Longitudinal acceleration speed. No physical features related to drivers such as their seating positions or their heart rate as well as personal information such as drivers' gender and age are available for this dataset.

Moreover, the HCI-lab group [58] collect a dataset of real-world driving to evaluate the drivers' workload. Smartphones as well as wearable technologies and webcams are used to collect data about ten participants between the ages 23 and 57 with an average of 35 years old. There are seven males and three females in this dataset with different driving experiences. The data are collected from five different road types such as highways and tunnels and include almost 2.5 million records. Initially, the dataset contains various individual files for each driver, which is helpful in assigning collected data to drivers. We process each file and then integrate them into a single time series file having driving instances for all drivers and label each driving instance from 'driver one' to 'driver ten'. It should be noted that the labels are only used for the task of driver identification using

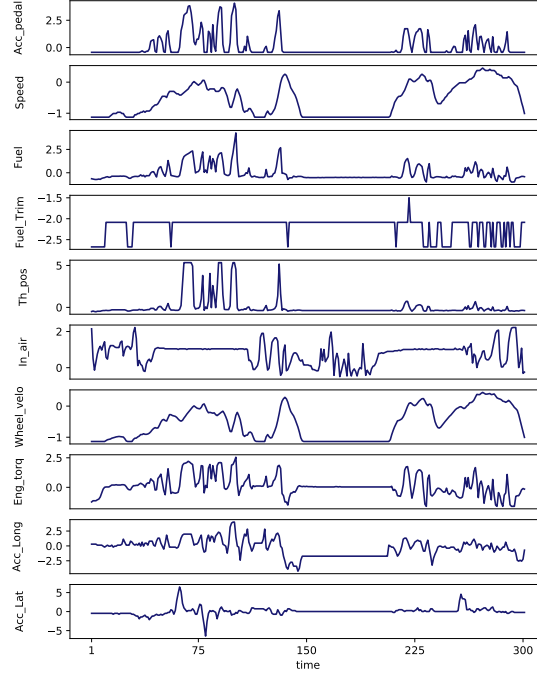


Figure 4.5: A sample of HCRL dataset from driver A.

SVM on the extracted driver embeddings. Different driving-related and physiological features, including acceleration (X, Y, Z), speed, SCR, Temp and heart rate (HR), HRV-LF, and ECG, are used. An example of the dataset is shown in Fig. 4.6 for three selected drivers using the speed signal. This dataset is selected to investigate the capability of the DriverRep in handling various types of data sources. Moreover, the length of data is not the same for all drivers, which makes it suitable to evaluate the ability of the DriverRep for variable input length as discussed in Section 4.1. According to the size of the datasets, the data split ratio for training and test is considered the standard [290] 8:2 for both datasets.

4.2.2 Multi-way Driver Identification Results

In real-world scenarios, a driver identification model should most likely perform the identification task over a small subset of drivers such as the members of a family or

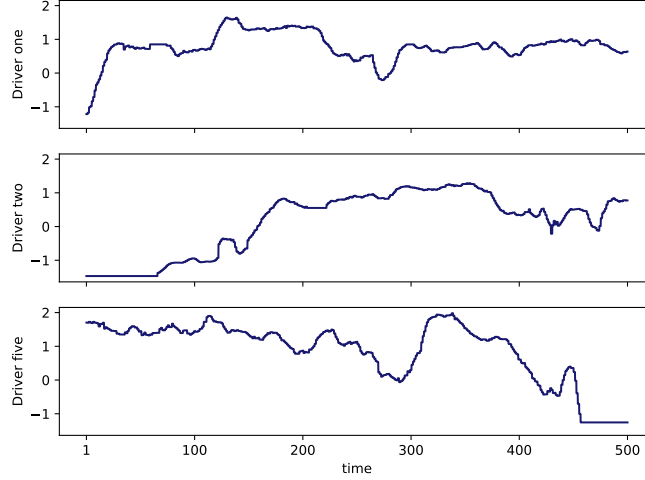


Figure 4.6: An example of speed signal from HCI-lab dataset for three drivers.

drivers sharing the same car in a fleet management company. For this purpose, we define an n -way identification experiment in which we perform the identification task over a subset of n drivers as a group for $n \in \{2, 3, 5, 10\}$. To set up this, for each value of n , we select the group of drivers to be present in the experiment from all possible group choices, repeat the experiment for each group, and report the average of the obtained results. For instance, in a three-way identification scenario and for a given driver, there can be a combination of $\binom{n-1}{2} = \frac{n-1!}{2!((n-1)-2)!}$ different group choices that include the given driver, where n is the total number of drivers.

Table 4.1 summarizes the obtained results of the DriverRep model for n -way identification over the two datasets, where $n \in \{2, 3, 5, 10\}$. The number of negative samples (see Algorithm 1) is selected as 5 and 10. Considering the HCRL dataset, for two-way identification, the DriverRep obtains 95.2% accuracy. The performance then declines to approximately 94%, 88%, and 75% for a three-way, five-way, and ten-way accuracy when $neg = 5$. The same trend can be observed in the results achieved for the HCI-lab using both neg values. This is because as the number of drivers increases, so does the possibility of having similar driving behavior among the drivers. In other words, considering an N -way identification, as the number of the labels (i.e., drivers in our defined problem) increases, it is more likely to come across more similar driving styles among the drivers, which makes it harder for the models to distinguish different drivers based on their

Table 4.1: Average driver identification multi-way accuracy of the DriverRep model.

N-way identification	<i>neg</i> = 5		<i>neg</i> = 10	
	HCRL	HCI lab	HCRL	HCI lab
Two-way	95.2%	96.3%	96.7%	96.8%
Three-way	93.7%	94.3%	95.7%	96%
Five-way	88.6%	89.5%	89.8%	92.3%
Ten-way	75.3%	79.2%	81.7%	85.1%

driving behavior. The observed behavior is consistent with the reported results by two state-of-the-art approaches ([164, 181]) over their extracted driving embeddings employing supervised triplet loss training. In [164], using acceleration signals and a real-world dataset, the authors report an accuracy of 23.87% for the twenty-five-way identification and an F1 score of 49.48% for two-way identification. Also, Yang *et al.* [181] achieved an accuracy of 83.1%, 62.5%, and 15% for the two-way, five-way, and fifty-one-way identifications, respectively. Moreover, the results shown in Table 4.1 represent the impact of the number of negative samples. A higher value of *neg* achieves higher average accuracy for both datasets. This is due to the fact that an increased number of negative values enhances the probability of considering various similarities and dissimilarities among drivers. As mentioned earlier, the deliberate choice of more negative samples serves the dual purpose of broadening the spectrum of distinctions among drivers and concurrently minimizing the likelihood of unintentionally including positive samples within our selection process. This strategy contributes to a more robust approach in capturing driver behavior.

Furthermore, it is worth mentioning that the model is not overfitted since the model performance is stable for training and test accuracies. For instance, consider the HCRL dataset and three-way classification for *neg*=10, the average training accuracy of the model is about 96.42%, while the average test accuracy, as shown in Table 4.1, is 95.7%. Also, consider the HCI-lab dataset and *neg*=10, the training results are 97.2%, 95.5%, 93.4%, and 87.3% for the two, three, five, and ten-way identification, respectively. Comparing these results with those of test data demonstrates that the overfitting problem has been avoided.

Figure. 4.7 represents the extracted driver embeddings from the multi-way identification experiment for both datasets. In consistent with the results observed in Table 4.1,

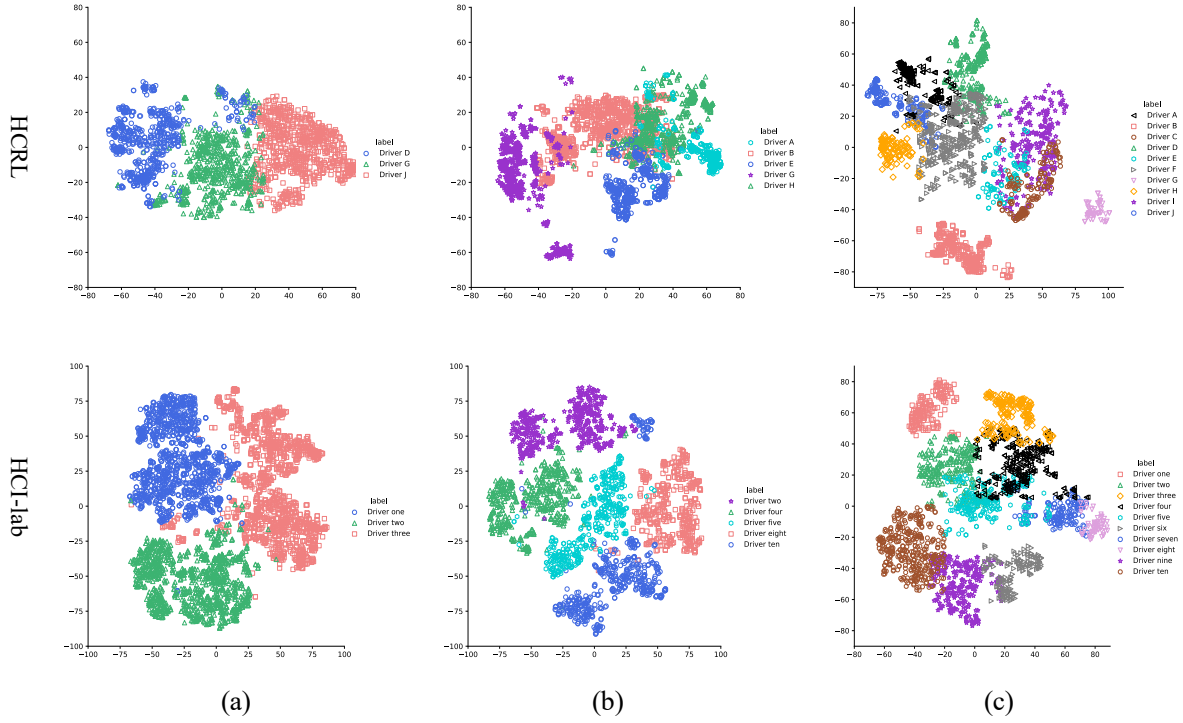


Figure 4.7: Extracted driver embeddings by the DriverRep for two datasets. (a) three-way identification (b) five-way identification and (c) ten-way identification.

as the number of drivers increases, it gets harder to distinguish between the drivers. Nevertheless, it is notable that some drivers are easier to identify, such as 'driver one' with an identification accuracy of approximately 98% in the HCI-lab, as they represent less similar driving behavior to other drivers and achieve higher identification results (see Fig. 4.8).

Moreover, Fig. 4.9 illustrates the extracted drivers' latent representations of four distinct driver pairs. In the first two rows, each column represents the results for the same two individuals. The first row demonstrates the original driving instances, and the second one shows the extracted representations. It can be observed that not all driver groups have the same difficulty level. For instance, drivers shown in Fig. 4.9a1 and 4.9a2 are easier to be distinguished than those presented in Fig. 4.9b1 and 4.9b2.

4.2.3 Comparison Results

In this section, we perform two different types of evaluations. First, we compare the results of our proposed methodology with state-of-the-art driver identification schemes.

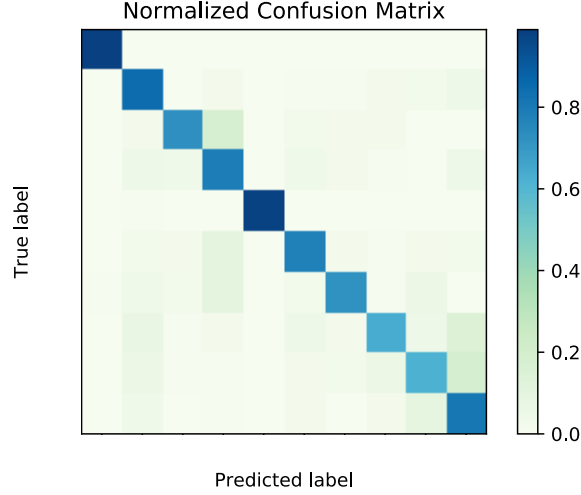


Figure 4.8: Confusion matrix of ten-way identification over the HCI-lab dataset.

We then compare the proposed model with benchmark time series classification models with respect to accuracy over multivariate driving time series classification and handling sparsely labeled data.

Driver Identification Schemes

We compare the proposed DriverRep identification scheme results with those of the existing benchmarks for driver identification. Point to be considered, this comparison is according to our obtained results for the HCRL dataset and those reported in the previous respective studies using their best settings.

Regarding the use of triplet loss and driver embeddings extraction for the task of driver identification, our results (96.7%, 95.7%, 89.8%) can be compared to those of Yang *et al.* [181], which report an average accuracy of 83.1%, 73.8%, and 62.5% for two-way, three-way and five-way identification, respectively. To extract the driver embeddings, the authors use several feature groups similar to ours (acceleration, speed, steering wheel, pedal values) in addition to distance information, turn indicators, and lane-related features [181]. They employ Gradient Boosting over the extracted driver embeddings. Also, Sanchez *et al.* [164] report an accuracy of 23.87% for a group of 25 drivers using triplet loss training. Our results are also comparable with other driver identification schemes that do not extract driver embeddings. Using a set of similar features to ours (steering

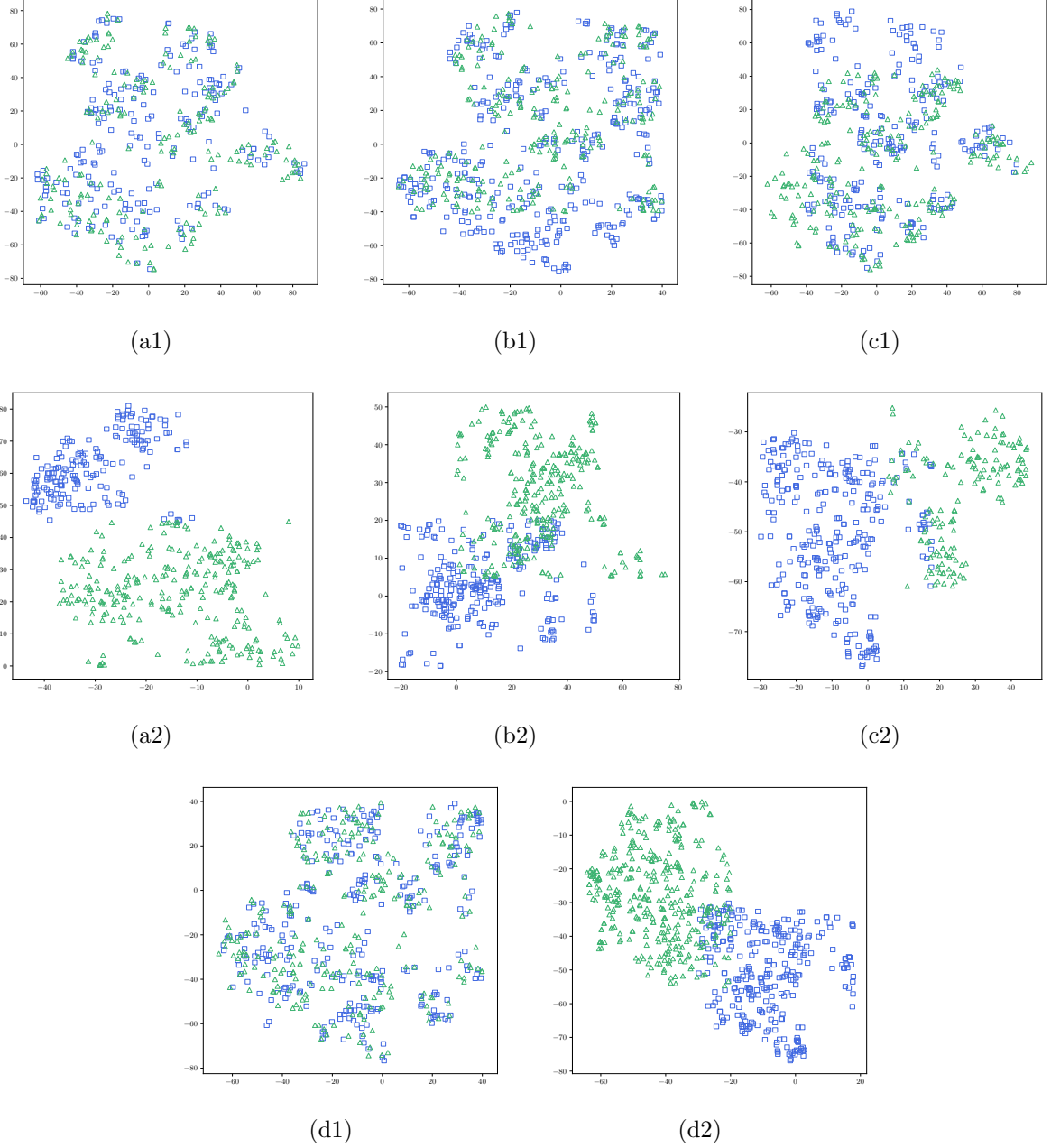


Figure 4.9: Comparison between the instances of four driver pairs. Each alphabetical letter represents the results for the same two drivers. Parts 4.9a1- 4.9d1 demonstrate the original driving instances, i.e., the input. Parts 4.9a2- 4.9d2 illustrate their extracted latent representation.

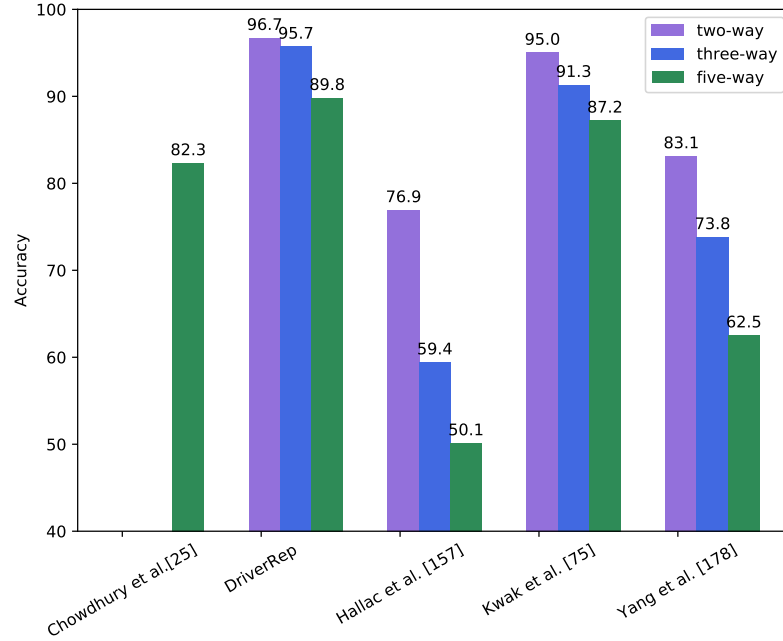


Figure 4.10: Comparison of different driver identification schemes in terms of accuracy.

wheel, speed, engine RMP, throttle position, engine torque, pedal values, and acceleration), Hallac *et al.* [160] report an average accuracy of 76.9%, 59.4%, and 50.1% for a two-way, three-way, and five-way identification, respectively. The reported results by Chowdhury *et al.* [25] over 137 extracted features from the GPS data indicate an average accuracy of 82.3% for four/five-way identification (each group in their evaluation setting is made up of 4 or 5 drivers). Recently, the scheme proposed by Kwak *et al.* [75] achieves an average accuracy of 95.07%, 91.3%, and 87.2% for a two-way, three-way, and five-way identification using the SVM classifier and a set of features similar to ours (fuel, throttle position, pedal values, intake air pressure, vehicle speed, torque, steering wheel, and fuel trim) in addition to yaw rate as well as transmission, brake and engine-related features. Fig. 4.10 summarizes the results for different driver identification schemes whose details are discussed above. The two-way and three-way identification results are not available for the first method in the graph. As can be seen, our obtained results are comparable with recent and state-of-the-art driver identification schemes.

In another experiment, we compare the results obtained from our model with those obtained from replicated driver identification schemes over the HCRL dataset. Point to be considered, the results are not fully representative of the original works [160, 181] as

the features used (See Section 4.2.1) are as close as possible to the original works but not the same due to availability limitations. For the first model, similar to the work by Yang *et al.* [181], we use temporal convolution combined with Haar wavelet transform to extract the driving embeddings and Gradient Boosting model to classify the drivers. Parameter setting is Adam optimizer, decay = 0.975, and embedding size of 64 for the deep learning method, 30 leaves, 100 trees, depth of 12, and a feature fraction of 0.8 for the classifier obtained from a grid search. The results demonstrate an average accuracy of 88.7%, 77.6%, and 67.5% for the two, three, and five-way identification, respectively. Moreover, we employ both extracted Haar wavelet and statistical features such as mean and standard deviation of the driving sensors for the second model. Similar to the work by Hallac *et al.* [160] we define the window intervals based on the turns. The results indicate an average accuracy of 81.2%, 64.7%, and 58.8% for the two, three, and five-way identification over five turns, respectively.

Time Series Classification

In another experiment, we apply several well-established time series classification models over the HCI-lab dataset and compare the accuracy results with those of the DriverRep. For this purpose, we reimplement four benchmark deep learning methodologies including fully convolutional networks (FCN) [291], MLP, ResNet [282], CNN as well as the DTW_D algorithm [292], and k-nearest neighbor with Euclidean distance as a baseline. For the ResNet model, we implement three blocks of residual blocks consisting of the 1D convolutions, batch normalization, batch size of 10, and ReLU activation. The output of the blocks is then fed to a global average pooling followed by a fully connected layer with softmax activation. For the MLP model, we use four layers of fully connected layers and a softmax classifier as an output layer. The batch size and dropout rates are 10 and 0.5, respectively. In FCN, we implement five layers of fully connected convolutions as well as batch normalization and ReLU activation followed by an average global pooling layer. Table 4.2 represents the optimized hyper-parameters by grid search, for all the deep learning approaches in this experiment.

The evaluation results are summarized in Table 4.3. As demonstrated, the DriverRep outperforms FCN, CNN, and MLP deep learning approaches. Moreover, relative improvements of 27% and 10% are achieved when compared with baseline and DTW_D models, respectively. It is also worth mentioning that a random classifier (uniformly random prediction) achieves an accuracy of 10% (1/10 for ten drivers). In comparison with the ResNet, it should be kept under consideration that this model, as a supervised

Table 4.2: List of optimized hyper-parameters for the deep learning approaches where b represents the number of blocks.

Model	Layers	Act.	Optim.	epochs	Learning-rate
CNN	3	ReLU	Adam	400	0.001
DriverRep	11 (3b)	ReLU	Adam	400	0.001
FCN	5	ReLU	Adam	400	0.001
MLP	4	ReLU	Adam	500	0.9
ResNet	11 (3b)	ReLU	Adam	400	0.001

Table 4.3: Prediction accuracy (ten-way) of different time series classification models over the HCI-lab dataset.

Model	Accuracy	F1-score
ResNet	92.88 ± 0.18	91.63 ± 0.14
DriverRep ($neg = 10$)	85.08 ± 0.02	85.11 ± 0.26
FCN	81.36 ± 0.31	81.24 ± 0.27
CNN	80.26 ± 0.04	80.98 ± 0.35
MLP	77.62 ± 0.11	78.08 ± 0.02
DTW _D	77.30 ± 0.02	77.61 ± 0.01
Baseline	67.15 ± 0.16	67.03 ± 0.15

approach, achieves higher accuracy than the DriverRep (SVM classifier on top of the driver embeddings that are extracted in an unsupervised manner) when tested on the fully labeled dataset. In our next experiment, we highlight the considerable potential of the DriverRep in handling sparsely labeled data in compared to the ResNet.

We then perform a sparse labeling experiment to investigate the capability of the extracted driver embeddings by the DriverRep in handling sparsely labeled data. Fig. 4.11 demonstrate the accuracy of both our and the ResNet approaches while varying the size of the labeled data. The error bars in this figure show the standard deviation for multiple runs. As can be seen, the driver embeddings can be successfully extracted from datasets with small portions of the labeled data, outperforming the supervised ResNet method. For instance, when applied on 2% of the labeled data, the DriverRep obtained

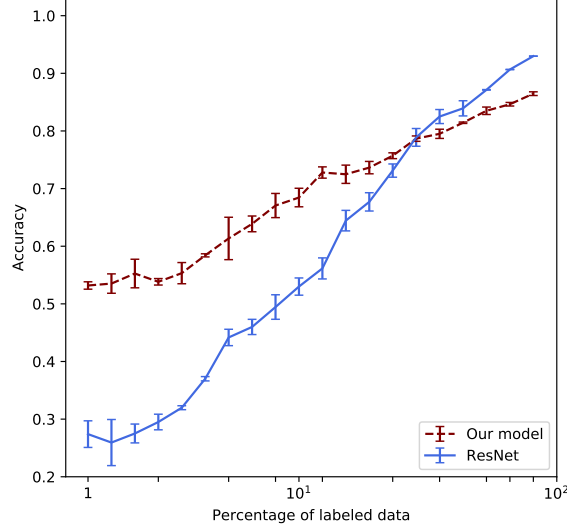


Figure 4.11: Prediction results of sparse labeling.

an accuracy of more than 50% in comparison with almost 25% for the ResNet method. This indicates the advantage of unsupervised triplet loss training and proposed encoder architecture in the DriverRep model against the ResNet model.

4.2.4 Ablation Analysis

In this section, we investigate the correlation between embedding size and model performance, exploring how variations in embedding dimensions impact the effectiveness of DriverRep. Additionally, we conduct a comprehensive component ablation study to investigate the effect of introduced architecture on our proposed model.

Embedding Size

We also conduct two different model analysis experiments. First, we design an experiment to assess the performance of our model when different embedding sizes are selected. The size of the extracted embeddings demonstrates how well the latent representations retain the information and also the complexity level of the driving characteristics of the drivers. Fig. 4.12 compares the prediction accuracy of the proposed model over two datasets when embedding sizes of 32, 64, and 128 are used. As can be observed, the proper embedding

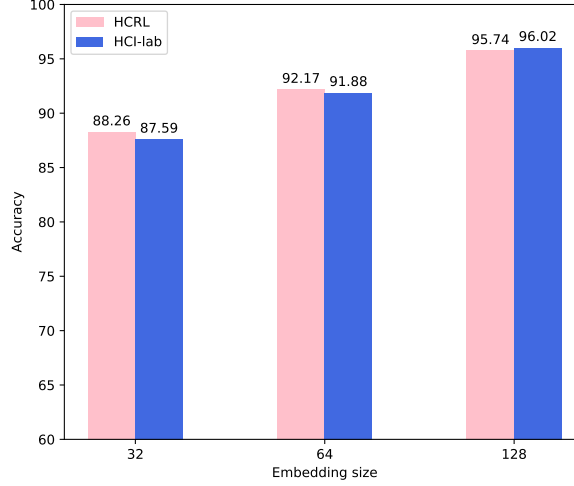


Figure 4.12: The effect of different embedding sizes on the performance (three-way) of the DriverRep with $neg = 10$.

size has a significant impact on the performance of the model. According to the obtained results over the selected datasets, a size of 128 is able to capture the underlying features of the driving time series entirely. In other words, a driver driving behavior can be demonstrated with 128 dimensions.

Embedding Extraction Models

In this part, we perform a model ablation study by which we aim to assess the performance of the introduced encoder architecture as well as its significance in shaping the DriverRep. More specifically, let us consider the DriverRep architecture and its main components, demonstrated in Fig. 4.1. To perform the ablation study, we change the architecture of embedding extraction encoder component and obtain the driver identification results. We extract the latent representations for all drivers using four different recurrent units. We will explain each employed recurrent units in the following.

Recurrent Neural Networks. are introduced as a powerful DL tool for sequence modeling and handling temporal dependencies that exist in the data [293]. The main feature of the RNNs is that the output of the previous steps is also employed as the input of the current step. There are different types of RNNs based on the fact that which of the input or output is sequence data and if the length of the input and output is the

same or not. These types indicate how the main components (i.e., recurrent units) can be put together. Among all the existing types, many-to-one is of interest in this research as it is suitable for classification tasks where the input is time-series data, and the output is a label. We implement a Vanilla RNN-based encoder.

Long Short-Term Memory. is a well-established recurrent unit introduced by Hochreiter and Schmidhuber [294]. The utilized sophisticated gates in its architecture make memorization of past information easier. LSTM consists of three different gates including forget (f), input (i), and output (o) gates. The characterizing equations of LSTM architecture are as follows:

$$\begin{cases} i_t = \sigma(W_i[a_{t-1}, x_t] + b_i) \\ f_t = \sigma(W_f[a_{t-1}, x_t] + b_f) \\ o_t = \sigma(W_o[a_{t-1}, x_t] + b_o) \\ \tilde{c}_t = \tanh(W_{\tilde{c}}[a_{t-1}, x_t] + b_{\tilde{c}}) \\ c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \\ a_t = o_t \odot \tanh(c_t) \end{cases} \quad (4.17)$$

where W and c are the weight vector and the cell state, respectively. We implement a three-layer LSTM-based encoder with a learning rate of 0.001 and a decay rate of 0.0001.

Gated Recurrent Unit. is another type of recurrent unit introduced by Cho *et al.* [295]. As opposed to the LSTM, GRU contains two gates, making training much faster and having fewer parameters. The difference is that the step of forgetting old information and loading the new ones is done in a single gate. The forward-propagation equations for the GRU are denoted as:

$$\begin{cases} u_t = \sigma(W_u[c_{t-1}, x_t] + b_u) \\ r_t = \sigma(W_r[c_{t-1}, x_t] + b_r) \\ \tilde{c}_t = \tanh(W_{\tilde{c}}[r_t \odot c_{t-1}, x_t] + b_{\tilde{c}}) \\ c_t = u_t \odot \tilde{c}_t + (1 - u_t) \odot c_{t-1} \end{cases} \quad (4.18)$$

where u , r , W , and c are update gate, reset gate, weights, and cell state, respectively. We have developed a three-layer GRU-based encoder with a learning rate of 0.001 and a decay rate of 0.0001.

Prototypical Recurrent Unit. is a recent variant of recurrent units proposed by Long *et al.* [296]. They introduced PRU as an alternative to LSTM and GRU networks

Table 4.4: Driver identification accuracy scores for different encoder architectures with respect to three different representation size on HCI-lab.

Model	Three-way			Five-way		
	32	64	128	32	64	128
RNN	54.78	63.32	68.53	51.34	58.27	63.13
LSTM	72.84	75.09	83.82	65.73	71.19	79.64
GRU	81.31	84.06	89.78	75.96	81.83	85.04
PRU	80.20	84.83	90.36	75.44	79.63	85.32
DriverRep	87.59	91.88	96.02	81.03	88.62	92.28

that captures the major components while preserving simplicity. Similar to LSTM and GRU, PRU has a gradient freeway, making it possible to avoid the vanishing gradient problem. The main equations for the PRU unit can be summarized as below:

$$\begin{cases} u_t = \sigma(W_u[c_{t-1}, x_t] + b_u) \\ \tilde{c}_t = \tanh(W_{\tilde{c}}[c_{t-1}, x_t] + b_{\tilde{c}}) \\ c_t = u_t \odot \tilde{c}_t + (1 - u_t) \odot c_{t-1} \end{cases} \quad (4.19)$$

We also choose to implement PRU since it has been shown [296] that PRU has the minimal complexity among these recurrent units and performs comparably to LSTM and GRU over several tasks. This way, its performance can also be evaluated on driver identification accuracy. We implemented three layers of PRU with a learning rate of 0.0001 and decay (0.9, 0.9999).

Furthermore, We report the driver identification accuracy for comparison models in Table 4.4. The accuracy results are obtained for two different group sizes as well as three representation sizes. As illustrated, the DriverRep encoder performs better than recurrent-based encoders for all group and representation sizes. This suggests the capability of the introduced approach in capturing the essential driving characteristic of each individual. Moreover, comparing the accuracy results for several RNN-based encoders, we can see that RNN demonstrates a poor performance for all group and representation sizes. This highlights the inherited problem within such architecture, i.e., gradient vanishing. Furthermore, the results obtained for all encoders using different embedding sizes are consistent with the results achieved in embedding size part, confirming that the

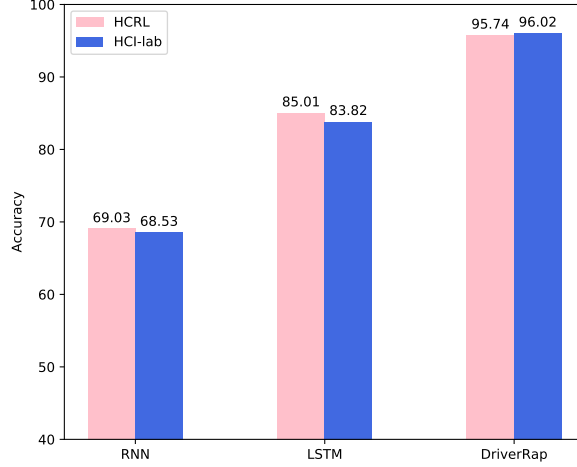


Figure 4.13: The model ablation study results (three-way) for the encoder architecture of the DriverRep ($neg = 10$).

representation dimension can considerably impact the effectiveness of models. Considering our model, the identification accuracy increases by 10% when the representation dimension is 128. This can be specific to the complexity of the used HCI-lab dataset, where the distinctive driving behavior of drivers can be summarised using 128 representation dimension. Another point is that when the number of drivers within a group increases, so does the complexity of the problem, resulting in a reduction in accuracy for all comparison models.

Figure. 4.13 represents the evaluation results when different encoders are applied for the driver identification task. Based on these restricted experiments, we observe that the DriverRep encoder outperforms the other two encoders by a large margin, which indicates its capability in capturing the underlying driving characteristics within the data compared to RNN or LSTM as an encoder.

Furthermore, we qualitatively assess the effectiveness of TC and RNN in encoding extraction. We visualize the extracted representation from driving samples of four drivers using TC (Fig. 4.14a) and RNN (Fig. 4.14b). The same four drivers are used for both comparison models. As can be seen, TC successfully brings the driving samples of the same drivers closer, and distinguishes the drivers. This justifies TC’s higher driver identification accuracy presented in Table 4.4. Moreover, if we consider computational

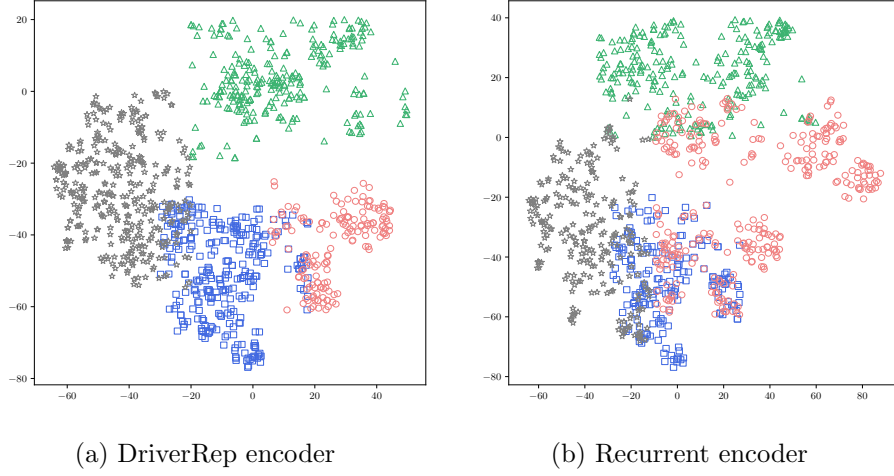


Figure 4.14: Comparing extracted driver embeddings by DriverRep and RNN for four drivers.

complexity, the inference time of TC is about 0.842ms while it takes 4.701ms for RNN. The reason is that, unlike recurrent-based networks where it is needed that subsequent units wait for the results of the previous unit, convolution operations can be performed in parallel.

4.3 Concluding Remarks

Taking into account the unique driving characteristics of the individuals for distinguishing between drivers has been actively investigated in recent years. However, the existing methods suffer from the need for the availability of a large amount of labeled data as well as the scalability of the models for practical scenarios. In this chapter, we address these challenges and propose the DriverRep method to, first, efficiently extract the driver embeddings from the data in an unsupervised manner and, second, use these embeddings to distinguish between the drivers within a specified group. Overall, we can conclude the following points based on the observed results of various experimental evaluations. First, the DriverRep is able to successfully capture the underlying driving characteristics within the data. The identification results obtained over the extracted driver embeddings by the DriverRep are considerable in comparison with the state-of-the-art models on the driver embeddings extraction. The DriverRep also outperforms benchmark driver identification schemes that are applied directly on the data (i.e., no driver embeddings

are extracted). Second, compared to the time series classification approaches, the DriverRep results over a dataset of ten drivers represent a relative improvement of 27% over those of the baseline approach and higher performance than those of three deep learning approaches, including FCN, MLP, and CNN. Finally, the DriverRep outperforms the supervised ResNet approach when applied to the highly sparsely labeled data, which tackles the challenges outlined earlier. The ablation analysis results indicated the potential of DriverRep architecture in extracting driver embeddings compared to recurrent unit-based encoders.

Chapter 5

Siamese TCNs for Driver Identification using Driver Steering Behavior Analysis

In this Chapter, we address the data dependency and scalability challenges for both driver identification and impostor detection tasks. We design a novel deep learning-based system to perform these tasks through analyzing the steering behavior of the drivers. Steering behavior has proved to be useful in some other tasks such as detecting drunk drivers [49], steering evasion control [297], or the design of advanced ADAS for lane-keeping or cornering [298]. In the present chapter, however, we analyze steering behavior in different scenarios with the purpose of capturing distinguishing characteristics among drivers and employ such characteristics to identify the person behind the wheel. As an illustrative example, consider the left turn maneuver shown in Fig. 5.1 from the dataset. We compare the steering wheel angle of three different drivers at the same turn. After keeping the steering wheel straight at the intersection, the drivers started to turn the steering wheel to perform the turn maneuver. Driver A turns earliest while drivers B and C do not move the wheel until reaching the peak of the turn. Driver B also demonstrates a sharper steering behavior. These are the kinds of unique features in the driving behavior of drivers that we aim to capture with the proposed system to, first, identify the driver and then detect any potential impostors.

More specifically, in this chapter, we develop a novel deep learning-based system architecture to identify drivers out of a given set of individuals through modeling their steering behavior. This way, unnecessary and extensive data collection can be avoided

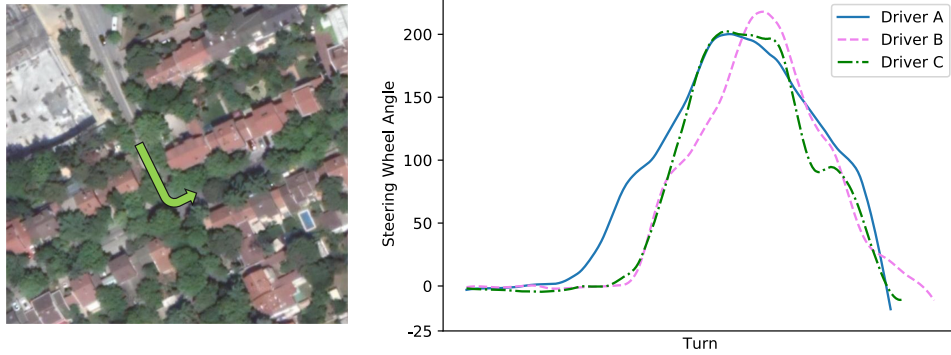


Figure 5.1: Steering wheel angles (in degrees) of three different drivers at the same turn.

and the data dependency of the identification models can be reduced. We also design a novel methodology to address driver verification as a comparison problem using the combination of the developed architecture and Siamese networks concepts to verify the driver while having access to a short interval of driving data. We further extend the proposed driver verification to tackle impostor detection in an open-set setting. The proposed methodology addresses the generalization challenge and eliminates the need for re-training the models for different types of impostors.

The remainder of this chapter is structured as follows. In Section 5.1, we discuss an overview of the proposed system and the used dataset. We explain our proposed driver identification and impostor detection approaches in Section 5.2. The experimental evaluation results are demonstrated and discussed in Section 5.3. Finally, we discuss the conclusion in Section 5.4.

5.1 System Overview

In this section, we explain the high-level structure of the proposed system and provide insights into the dataset. The system includes two main modules, one for driver identification and one for impostor detection as shown in Fig. 5.2. The primary step before conducting any analysis is to make sure the data is of high quality and to perform any necessary preprocessing steps. In the preprocessing sub-module, we look for any fundamental error points caused by data recorder malfunction. These errors include impossible vehicle kinematics such as when all driving signals suddenly flatline at a specific time.

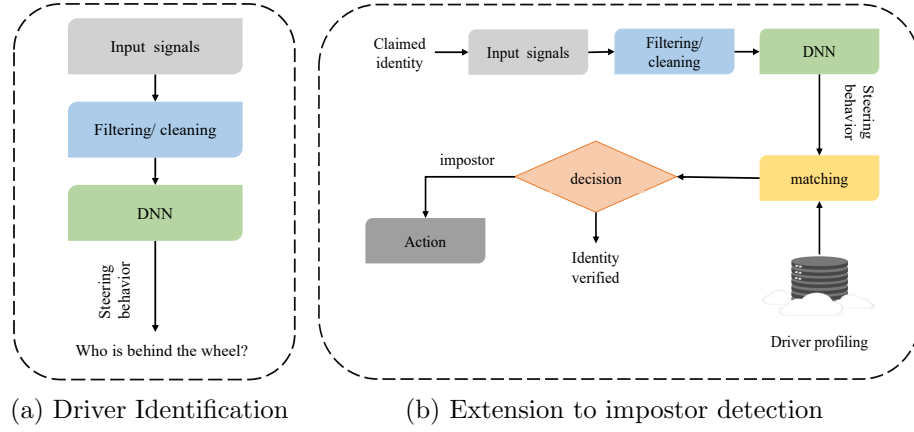


Figure 5.2: General flow of the high-level driver identification system architecture.

Significant irregular patterns or abnormal points are detected and replaced with a value inferred from close data points. We then feed data to the proposed deep neural network architecture where we analyze the steering behavior of the drivers. The results can then be used either to perform a classification task to recognize the individual behind the wheel or to detect an impostor. In an impostor detection scenario, the proposed system analyzes the new data coming from an impostor and compares it with previous results from driver profiling of the authorized drivers. An appropriate action such as warning and informing the owner of the vehicle, the company, or authorities can be taken if an impostor is detected.

We analyze an anonymized dataset collected by VPALAB group, namely UYANIK [69], in which drivers are asked to drive on real roads in Istanbul, Turkey. The initial dataset contains various types of data including the video recording of the drivers and the road and audio recordings as well as vehicle signals, and has the total size of more than 9 TB. Twenty four different sensor data is collected using CAN-bus, IMU, GPS, cameras, and the laser scanners. The original CAN-bus data contains the data related to vehicle speed, steering, engine RPM, yaw rate, and wheel speeds. In this chapter, we take advantage of three features related to the steering and lateral position of the vehicle including steering wheel angles, steering wheel relative speed, and yaw rate. These three signals are extracted from the CAN-bus data of the vehicle using a special software with the sample rate of 32 Hz. Steering wheel angle and relative speed are read in degrees and degrees/second, respectively. Table 5.1 summarizes the main characteristics of the subset of data used in this chapter. As illustrated, the used data covers a wide variety

Table 5.1: The main characteristics of the used subset of UYANIK dataset.

Feature	Value
Drivers	95
Source	CAN-bus
Used Signals	Steering wheel angles, steering wheel relative speed, yaw rate
Average trip length	25 km / 47:68 m
Road types	City ways and motorways and highways

of traffic and road conditions such as city ways and motorways. All three signals are sampled at 32 Hz for all the drivers and, thus, eliminate the need for synchronization. Figure 5.3 demonstrates total driving duration in seconds for each driver in the system. Longer trip lengths are the result of heavier traffic. Furthermore, illustrative histograms of employed signals, obtained from two randomly selected groups of drivers, are shown in Fig. 5.4, representing different driving styles among drivers.

5.2 Proposed Methodology

In this section, we will explain the main architectures, methodologies, and strategies in the proposed system for driver identification, verification, and impostor detection.

5.2.1 Driver Identification

Driver identification is the task of identifying an individual behind the wheel among a closed-set group of drivers. Note that we will explore identifying drivers in an open-set setting later in Section 5.2.3. We consider this task as the classification problem in which the drivers, denoted as $D = \{d^1, d^2, \dots, d^n\}$, are the primary classes with $n = 95$ in our case. We design a deep learning-based system architecture to analyze windowed segments of driving data from drivers and capture the main underlying and unique characteristics within the data to identify the individual behind the wheel. Figure 5.5 illustrates our main system architecture. As shown, the input of the system is the driving time series. The number of the features in the time series are considered as channels. This data is then analyzed through a deep neural network including layers of stacked blocks. To classify the drivers, the output feature maps are passed through a fully connected layer

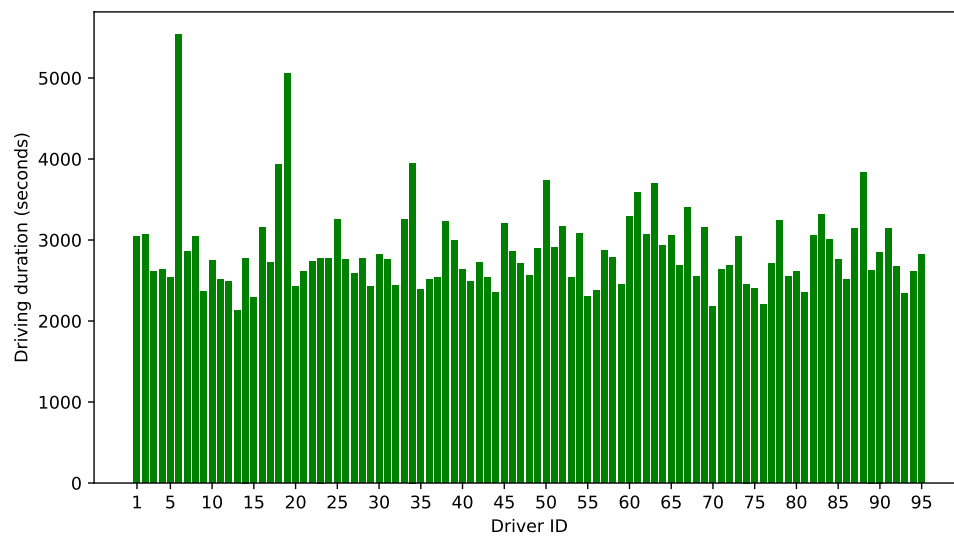


Figure 5.3: Total driving duration for each driver in the used dataset.

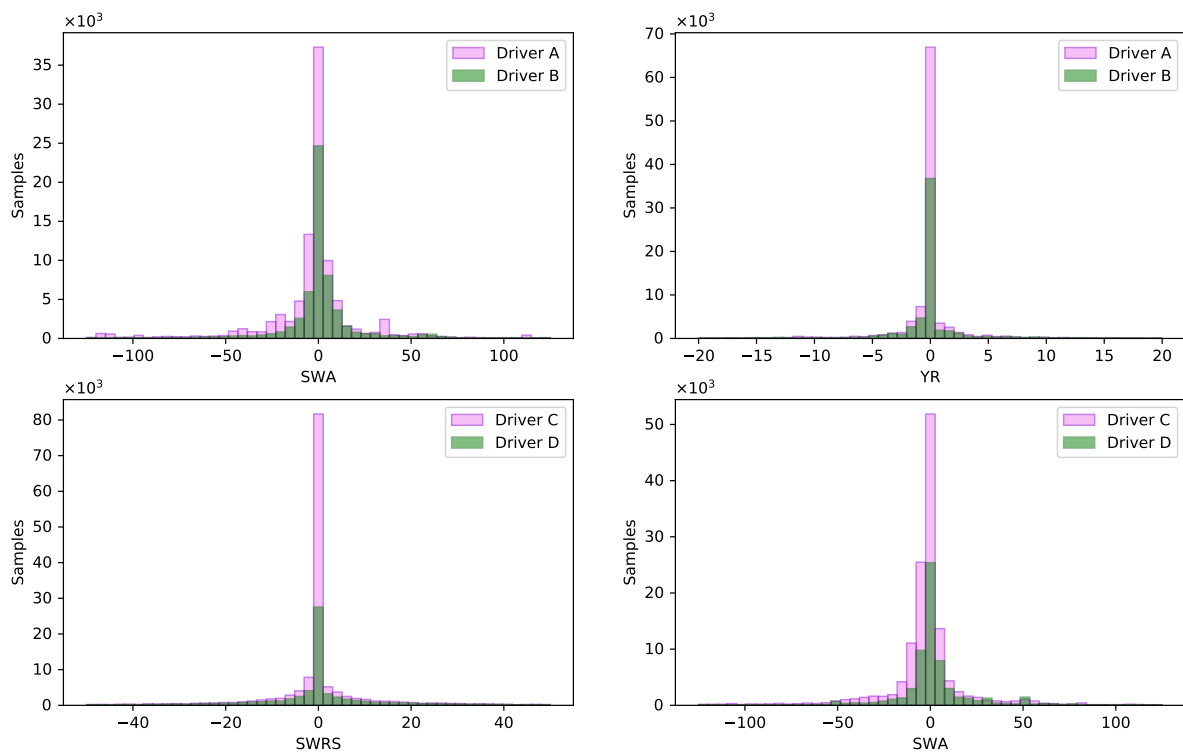


Figure 5.4: Histogram of steering wheel angles (SWA), yaw rate (YR) and steering wheel relative speed (SWRS) for two groups of drivers.

and softmax. We further explain different components in detail. For the driver d^i , a windowed segment of data can be defined as follows:

$$W_j^i = [\vec{V}_1, \vec{V}_2, \dots, \vec{V}_m] = \begin{Bmatrix} v_1[0] & v_2[0] & \dots & v_m[0] \\ v_1[1] & v_2[1] & \dots & v_m[1] \\ \vdots & \vdots & \ddots & \vdots \\ v_1[s] & v_2[s] & \dots & v_m[s] \end{Bmatrix} \quad (5.1)$$

where s , v , and m are the length of the window, the vector of features (i.e channels), and the number of the features, respectively. The length of the window is defined as the length of data in seconds we look ahead. The input of the system is the extracted segments from the total trip of the drivers. Figure 5.6 illustrates the data sampling process for the steering wheel angles signal of driver d^{46} . Driving instances are sampled with overlapping windows. Then, the data for driver i and all drivers in the system can be shown as (Eq.5.2).

$$z = \begin{Bmatrix} W_1^i \\ W_2^i \\ \vdots \\ W_j^i \end{Bmatrix}, Z = \begin{Bmatrix} W_1^1 & W_1^2 & \dots & W_1^n \\ W_2^1 & W_2^2 & \dots & W_2^n \\ \vdots & \vdots & \ddots & \vdots \\ W_j^1 & W_j^2 & \dots & W_j^n \end{Bmatrix} \quad (5.2)$$

In each window, the joint probability of the data over one channel $v_i = \{v_1, v_2, \dots, v_s\}$ can be computed as the following:

$$p(v) = \prod_{t=1}^s p(v_t | v_1, \dots, v_{t-1}) \quad (5.3)$$

which indicates that each record is conditioned on the data totally from the past. To model this conditional probability distribution and fully capture the underlying characteristics within the data, we use causal convolutions [299] as the primary component of our system. The main feature of the causal convolutions is that, at a certain time t , the output is only dependent on the input data that is before that time. This way, we make sure that the model is not ignoring the ordering inherent in the driving data since the probability generated by the model at time t cannot have leakage from future $v_{t+1}, \dots, v_s$. It should be noted that fully capturing long-term temporal dependencies requires either larger receptive fields (i.e., the set of points from the original input of the network that can affect a specific point in the output) or more layers in the network. To avoid the large number of layers and reduce the computational complexity, we also

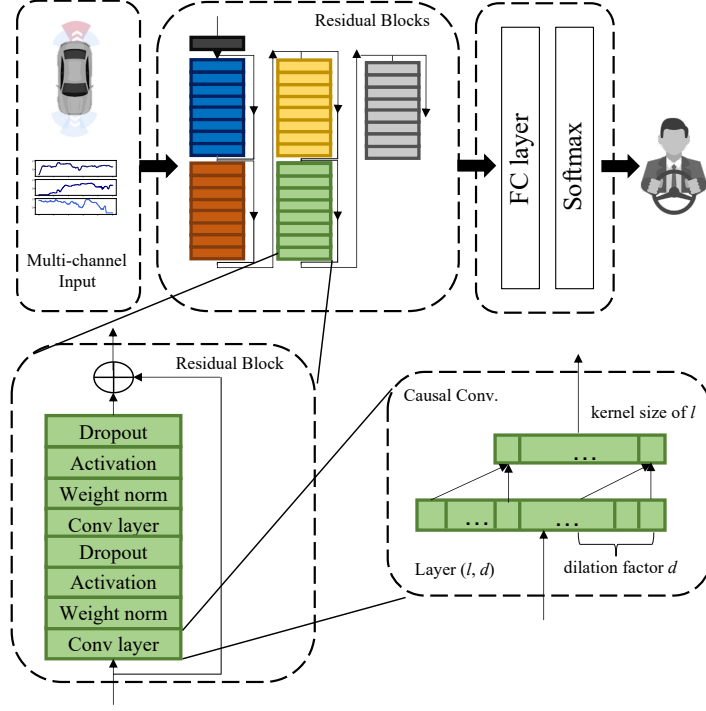


Figure 5.5: Architecture of the deep neural network for driver identification.

use the dilation concept which allows the filter of a convolution operation to be applied over a wider area. Dilation can be defined as the distance between the elements of the input that can have effects on a specific point in the output. This is due to the fact that this type of convolutions skips the input values with a certain amount (dilation) which can be increased exponentially [300]. Considering an input of I and the filter of k , the feature map f at time t can be defined as:

$$f(t) = (I *_d k)(t) = \sum_{l=0}^{l-1} k(l)I(t - d.l) \quad (5.4)$$

where l and d are the size of the filter and dilation factor, respectively. As shown in Fig. 5.5, the basic module of the proposed architecture is stacks of convolution layers followed by weight normalization, ReLU activation, and spatial dropout. In each block, skip connections (i.e., the input skips one or more layers and is concatenated with the output) are used to avoid degradation problems. It has been shown that skip connections [289] can help train and stabilize the network, especially when the input is long. The feature maps in the last layer are then fed to a softmax layer to be classified.

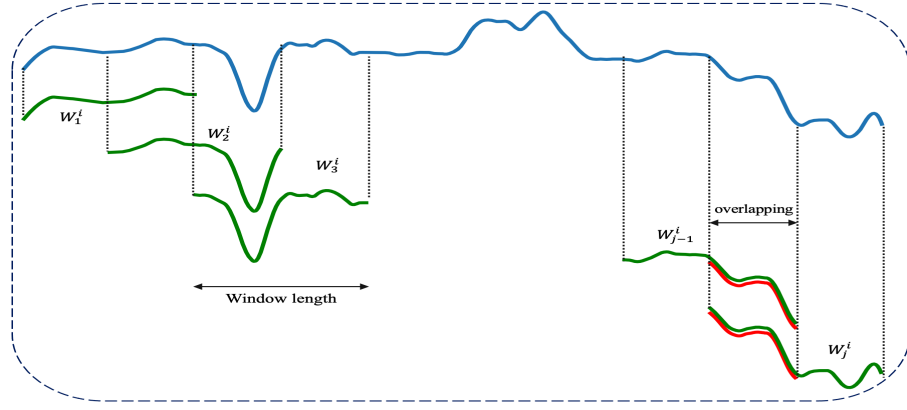


Figure 5.6: Illustration of the window sampling technique for a sample signal.

5.2.2 Driver Verification

In the task of driver verification, also known as driver authentication, the purpose is to verify or authenticate the drivers at the time of driving. This task differs from driver identification in the sense that, in this case, the problem is a one-by-one problem in which the aim is to accept or reject an identity. In other words, the primary aim is to check if the vehicle is driven by driver i . For this purpose, we propose considering the driver verification task as a comparison problem in which we intend to determine whether two drivers are the same in terms of comparing their steering behavior. To do so, we propose combining our temporal architecture with Siamese network concepts. As explained in Chapter 2, Siamese networks refer to any desirable identical networks that are used to compare two particular input data and have been mainly used in tasks such as image verification.

In contrast with the driver identification module where we use the proposed system to classify the drivers, we plan to employ the deep neural network to learn a similarity function. This similarity function gets data from two drivers and determines if the analyzed steering behavior belongs to the same people or not. Figure 5.7 represents the proposed system architecture for driver verification. The driving data from two different or same drivers are the inputs of the system. The networks shown on the top for driver i and bottom for driver k are completely the same and have the same building blocks as shown in Fig. 5.5. We can train the network for one driver and have an output of encoding that is a vector of features or latent representations associated with that driver. This encoding is denoted as $E(d^i)$. The same network can be used to get the encoding

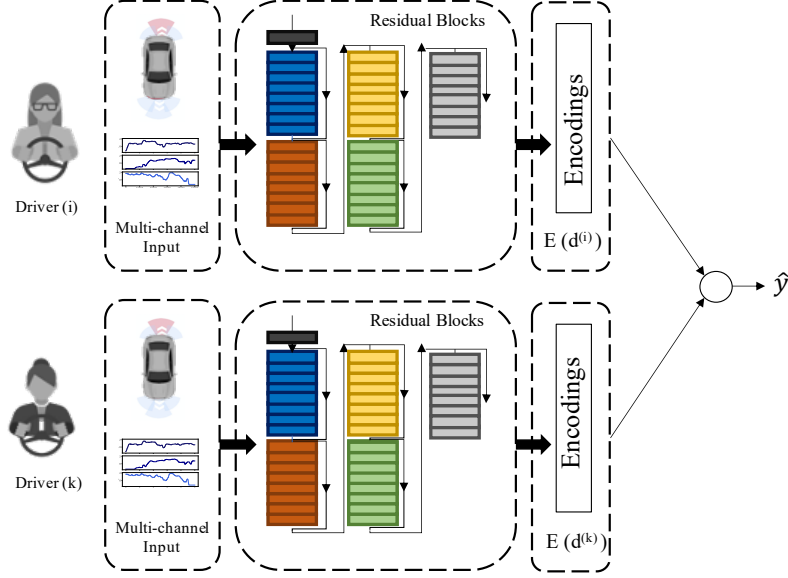


Figure 5.7: System architecture for the driver verification task.

for the second driver $E(d^k)$. After calculating these two encodings, we pass their distance to a logistic regression unit to decide if drivers i and k are the same person or not. As an example, imagine the encoding has the dimension of r , then the difference between the driving behavior of two drivers can be calculated using Eq. 5.5, where q is the number of components in the encodings and d^i and d^k are the data from both drivers. To calculate the L1 distance between the encodings, we need to take the sum over the element-wise difference between the absolute values of the encodings.

$$f(d^i, d^k) = \sum_{q=1}^r |E(d^i)_q - E(d^k)_q| \quad (5.5)$$

These $E(d^i)_q$ and $E(d^k)_q$ can be treated as the features that are later on passed to the logistic regression unit whose target output ($\hat{y}$) is calculated as:

$$\hat{y} = \sigma\left(\sum_{q=1}^r W_q |E(d^i)_q - E(d^k)_q| + b\right) \quad (5.6)$$

where w_q and b are the parameters of a normal regression and σ is the sigmoid activation. The output of this logistic regression unit is one if both of the drivers have the same identity. In contrast, the output will be zero if drivers are different. This way, we adapt the proposed architecture to perform the verification task as a comparison

problem.

5.2.3 Impostor Detection

We define impostor detection as an extension of the driver verification task where we aim at specifying if the person behind the wheel is among the verified or authorized drivers. In fact, the impostor detection problem can be considered as an open-set setting driver identification in which we try to understand if the driver is from outside of a group of known or authorized drivers. It is worth mentioning that, the driving instances of the impostor is only available during testing.

We come up with a solution for the driver verification task by considering it as a comparison problem in which the steering behavior of two drivers is compared and the result is whether these two drivers are the same person or not. We take advantage of the same strategy to address impostor detection where we compare the steering behavior of the impostor with that of all registered or authorized drivers. Similar to the verification task, we use the exact proposed network for all drivers. Figure 5.8 demonstrates the system architecture to detect the potential impostors. As shown, the data from the current driver can be used to calculate the similarity function between their data and that of all other drivers in the system. We can then compare the results of the similarity function to check if he/she is registered. The optimal value of the decision threshold is set as 0.23 i.e., if the distance is more than this value the drivers are considered different. This decision threshold is optimized using a grid search (within the range of (0, 0.5)) over cross-validation ($k=3$) on the training data. If the driver is none of the drivers in the database of the system, then the model should output small similarity numbers for all the comparison pairs it is making. Therefore, the system can determine the driver is none of the drivers registered in the database. In comparison with the classical models in which a new class is added to the system design to classify impostors and the authorized drivers, the proposed methodology has a significant advantage, i.e., the accuracy of the system is not dependent on whether the current impostor demonstrates the same driving behavior as the one defined in the new class.

5.3 Evaluation Results and Discussion

In this section, we perform a series of experimental evaluations to assess the performance of the proposed architectures. We provide concise explanations of the conducted

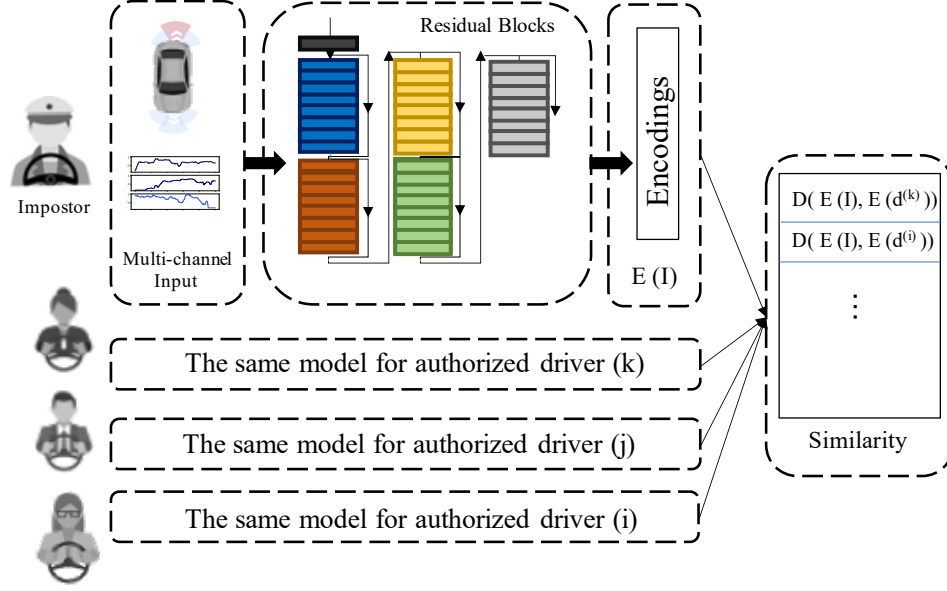


Figure 5.8: System architecture for impostor detection task.

experiments and discuss the obtained results.

5.3.1 Evaluation Results for Identification

We evaluate the performance of the proposed system architecture over the naturalistic dataset which we explained in Section 5.1. We analyze a short interval of steering behavior of the drivers to capture their unique characteristics. For all experiments (except the window length analysis one), the window length is set as 30s. The reason is, first of all, for several real-world applications of driver identification, it is desirable or even required to predict the driver's identity from a short interval of driving. Also, it should be reminded that the employed dataset is collect at 32Hz. Therefore, the window length of 30s represents an input size of approximately 1000 data points, large enough to contain two maneuvers. The effect of the window length on identification accuracy is further discussed at the end of this section. We perform cross-validation to avoid bias toward a particular traffic condition or part of a trip. At each fold and for each driver, the data is divided in 80:20% for training and testing pools which will be truncated according to the model size and prediction window. The prediction window and the model size are 30s, and 5, respectively. It should be noted that model size of 5 is equivalent to 90s of data in our scenario since the window size is 30s with 15s of overlapping. In other words, the

Table 5.2: Hyper-parameter settings for evaluations.

Model	Hyper-parameter	Value
Ours	Optimizer	Adam (0.0004, 0.975)
	Dropout	0.5
	Epochs	250
	Blocks	3
LSTM/GRU	Optimizer	Adam (0.01, 0.975)
	Dropout	0.4
	Epochs	250
	Layers	3
Siamese-LSTM	Optimizer	Adam (0.001, 0.975)
	Dropout	0.5
	Epochs	250
	Layers	4

amount of data used for the training at each time is 90 seconds. We employ Pytorch to implement all deep neural networks.

To evaluate the performance of the proposed system architecture, we compare the identification results of our system with those of two other recurrent deep learning approaches as well as several driver identification benchmarks. Table 5.2 reports the hyper-parameter settings for all deep learning-based approaches. The best settings are captured using a grid search over cross-validation ($k=3$) on the training data (divided into train and validation). It is worth mentioning that the evaluation setting including the amount of data used for training, the prediction window, number of drivers per group, and number of driver groups is the same for all comparison models as mentioned above. This evaluation setting is different from original settings of replicated methodologies. Furthermore, the feature extraction and preprocessing steps are adapted from the original studies. The comparison methods are as follows:

1. **LSTM.** We train a single LSTM model to predict the current driver of the vehicle. We implement 3 layers of LSTM recurrent units followed by a fully connected layer and softmax for classification. We use the same signals mentioned in Table 5.1.
2. **GRU.** A single GRU-based RNN is trained on the data. Hidden units are selected

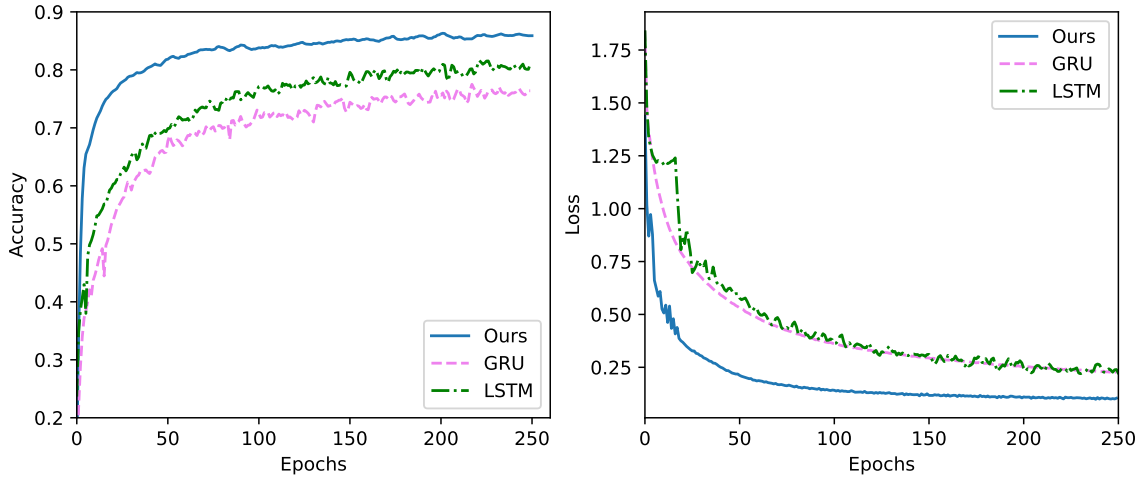


Figure 5.9: Comparison results on driver identification ($n = 3$).

in a way to maintain the total network size approximately as LSTM since the architecture is more complex. The signal sets mentioned in Table 5.1 is used.

3. **Classical ML.** [172]. We implement an ensemble of classical machine learning methods, including SVM ($c = 0.025$), GB (depth of 6), ET (depth of 8), and AdaBoost (rate of 0.75).
4. **GMM.** [157]. A GMM model is trained on the data using 3 and 0.5 as the number of components and α , respectively.
5. **Feedforward NN.** [171]. We implement a single feedforward network with one hidden layer followed by softmax for classification.
6. **Siamese-LSTM.** [168]. We train a 3-layer LSTM model followed by a single attention layer to execute identification evaluation.
7. **Ours.** The proposed system architecture is described in Section 5.2.1 in detail. We implement 3 blocks of stacked causal convolutions to better capture the underlying temporal dependency within data.

For this experiment, we evaluate the performance of all 95 drivers divided into groups based on ($n = \{2, 3, 4, 10, 15, 20, 30, 60\}$) number of drivers. For a certain group of drivers including n number of drivers, there exist $\binom{95}{n} = \frac{95!}{n!(95-n)!}$ possible combinations which

make it harder to calculate the results. Since the number of combinations for different values of n are high such as those for $n = 4$ (i.e., more than 3 million combinations), we randomly select 1000 subsets to perform the evaluation. Then, the achieved results for distinct groups having the same number of drivers are averaged and reported. For each group, we report three metrics including top 1, 3, and 5 accuracies. A top- k accuracy refers to the accuracy of a model in a way that the target driver is among the k highest probabilities. For instance, top-5 accuracy represents the success rate of a model if the target driver is among the group of 5 drivers having the highest prediction probability.

Table 5.3 summarizes the achieved results for all methods and groups of drivers. As demonstrated, the proposed system outperforms other comparison driver identification models as well as two recurrent neural networks. All three types of accuracy rates from our system are consistently higher than those of other methods across all driver groups. The reason can be the inherent capability of the proposed architecture in analyzing deeper dependencies in time in comparison with recurrent networks. An example of convergence results for training top-1 accuracy and loss for the top three approaches is shown in Fig. 5.9. As illustrated, the proposed method not only converges faster to the optimum results in comparison with LSTM and GRU but also achieves better final accuracy and loss for the target task. Furthermore, regarding the computational complexity, the inference time for our proposed model, GRU, and LSTM is 0.581, 1.996, 2.403ms, respectively. Moreover, the results shown in Table 5.3 demonstrate that the accuracy drops as a result of an increase in the number of the drivers across all models. Considering our model, having high top-1 accuracy of almost 90.4% for $n = 2$, it then reaches to about 25.8% for 60 drivers. This result is still better than those of LSTM, GRU, and random identification which are approximately 17.4%, 20%, and $1/60 \simeq 2\%$, respectively. Figure 5.10 shows the normalized confusion matrix of top-1 accuracy for $n = 60$ drivers. It is also worth mentioning that for any number of drivers n , the top- k accuracy is 100% when $k = n$ since the target driver will definitely be among the k top predicted probabilities. The obtained results in this section are also compatible with those reported by other driver identification systems that employ similar methodologies but have higher data dependency or complexity: An average accuracy of almost 70% for one group of 25 drivers [164] and an accuracy of about 83.1% and 15% for groups of 2 and 51 drivers [181].

Figure 5.11 illustrates the Cumulative Match Characteristic (CMC) graph for the proposed system when $n = 30$. The CMC curves demonstrate the top- k rank accuracy rates for different values of k for closed-set identification settings. As shown, starting with

a top-1 accuracy of approximately 68%, the top-k accuracy results then rise considerably as the rank increases. The improvement, however, is the greatest at the small numbers such as top-2 with a growth of almost 19.8%.

Table 5.3: Average accuracy results from 1000 combinations per group size for driver identification.

Drivers	Metrics	Models					
		Ensemble [172]	GMM [157]	Feedforward NN [171]	Siamese-LSTM [168]	LSTM	GRU
n=2	Top-1	44.38	45.67	68.52	76.31	80.47	84.23
	Top-5	90.36					
n=3	Top-1	42.07	44.54	66.88	73.07	78.32	82.68
	Top-5	87.26					
n=4	Top-1	41.23	40.92	63.72	72.40	76.47	80.02
	Top-5	96.38					
n=10	Top-1	38.94	39.98	58.73	69.61	73.46	75.24
	Top-5	80.04					
n=15	Top-1	42.38	43.44	60.26	75.30	86.98	90.38
	Top-5	94.35					
n=20	Top-1	45.64	47.12	60.90	88.75	89.92	92.00
	Top-5	94.35					
n=30	Top-1	30.47	35.87	43.23	66.74	69.27	74.73
	Top-5	79.82					
n=60	Top-1	36.56	40.63	43.48	72.94	79.03	85.06
	Top-5	90.86					
n=100	Top-1	39.04	41.94	43.48	76.86	88.62	91.82
	Top-5	94.01					
n=150	Top-1	23.68	30.17	35.8	62.13	65.75	67.18
	Top-5	73.80					
n=200	Top-1	30.57	35.53	37.94	63.54	74.80	80.46
	Top-5	86.44					
n=300	Top-1	32.60	37.12	37.94	70.14	86.17	89.87
	Top-5	92.53					
n=400	Top-1	9.40	21.75	30.00	57.06	59.01	62.78
	Top-5	67.82					
n=600	Top-1	11.62	22.32	30.28	59.29	69.85	77.12
	Top-5	86.23					
n=800	Top-1	13.98	26.54	31.54	63.61	84.37	89.98
	Top-5	90.01					
n=1000	Top-1	3.06	7.28	10.25	12.92	17.44	19.99
	Top-5	25.75					
n=1200	Top-1	5.21	9.46	11.04	14.13	23.50	26.53
	Top-5	29.12					
n=1400	Top-1	8.45	12.08	11.92	15.58	29.21	29.40
	Top-5	33.72					

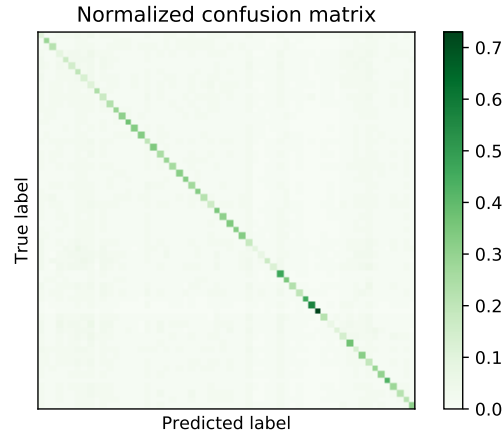


Figure 5.10: Normalized confusion matrix for driver identification ($n = 60$).

In another experiment, we compare the performance of the driver identification method using different signals including vehicles pedals, three accelerations signals, and vehicle speed. The model size and prediction window is set as 5 and 30s. We then report the average accuracy for 100 groups of 30 drivers ($n = 30$). The results indicate an average top-1 accuracy of 52.55%, 58.21%, 63.27% when only vehicle velocity, only pedal signals, and only accelerations are employed. It is worth noting that when comparing the results of different studies, various important points need to be considered, including the prediction window (i.e., the interval length of the driving data to identify the driver), the number of extracted features, and the number of distinguished groups of drivers for the reported group sizes.

We also examine how the selected length of the window can affect the performance of the proposed approach in terms of accuracy. As illustrated in Fig. 5.12, the longer the length of the window, the higher the top-1 accuracy. The shades indicate standard deviation. Nevertheless, it should be noted that the increase in the accuracy will be up to a point depending on the amount of data available. The reason is that as we increase the window length for sampling, the number of driving instances decrease. Considering a window length of 40s, the system can achieve an accuracy of over 90%. As described earlier, 30s is selected to maintain the trade-off between the real-world functionality of the system and its capability to analyze the steering behavior within the window. Due to the fact that the used data is collected at 32Hz, thirty seconds is large enough to analyze different driving styles over more than one driving maneuvers. For instance, the sample shown in Fig. 5.1 illustrates a left-turn maneuver that takes almost 12.5s.

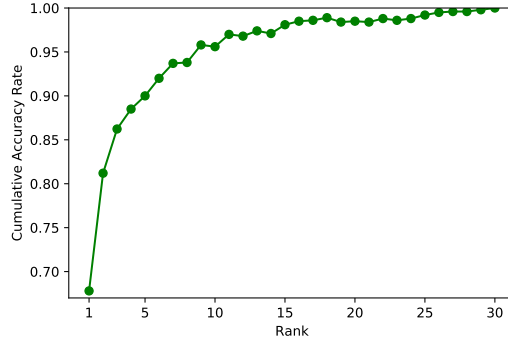


Figure 5.11: Normalized CMC curve for identification ($n = 30$).

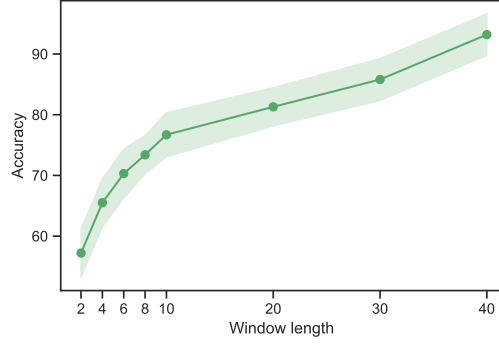


Figure 5.12: Effect of window length on accuracy ($n = 3$).

5.3.2 Verification and Impostor Detection Results

In this section, we present experimental results for driver verification and impostor detection. We evaluate the proposed methodology for driver verification using the following five metrics.

- **Precision** calculates the proportion of driving instances that belong to the target driver among all those associated to it, which can be denoted as:

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (5.7)$$

- **Recall** computes the proportion of driving instances that are correctly verified among all true authenticated drivers:

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (5.8)$$

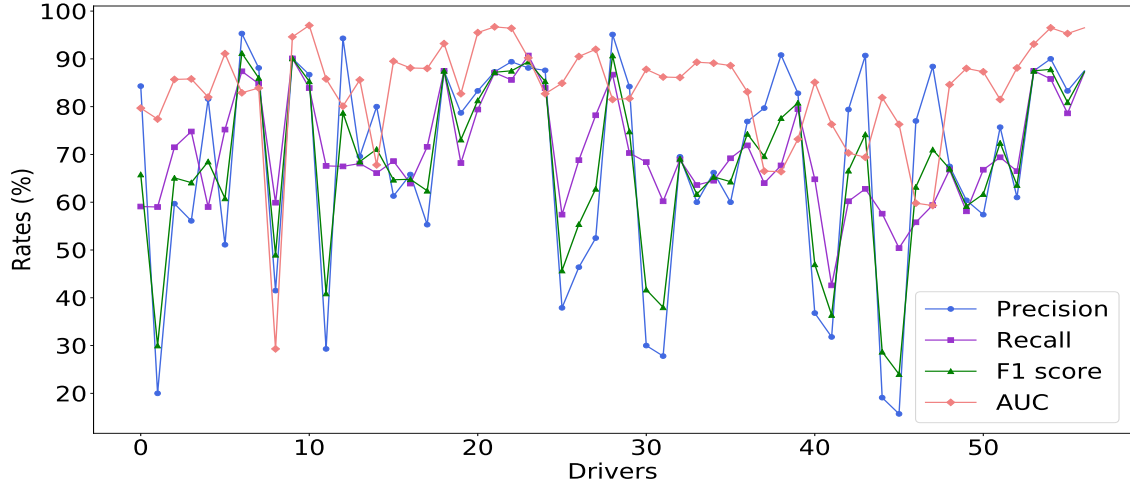


Figure 5.13: Evaluation results for 57 drivers in terms of four metrics.

- **F1 score** can be calculated as follows:

$$2 \frac{Precision * Recall}{Precision + Recall} \quad (5.9)$$

- **AUC and EER** measure the area under the curve and equal error rate that indicate how well verification systems are able to distinguish between the drivers.

The performance evaluation results across 57 drivers are obtained as the precision of 68.09%, recall of 70.53%, F1 score of 69.29%, AUC of 83.30%, and EER of 24.7%. The model size and prediction window for this experiment is 5 and 30s, respectively. The obtained result indicates that the extracted encodings for individuals can be particularly useful for verifying the drivers. Figure 5.13 demonstrates achieved verification results per driver present in the system. As can be seen, the system achieves high verification results for several drivers including drivers 7 and 55 to mention a few with the precision of 95.3% and 90.0%, recall of 87.4% and 85.8%, F1 score of 91.2% and 87.8, and AUC of 82.5% and 96.5%, respectively. Moreover, the system has difficulty verifying a small number of drivers such as driver 46 with the precision of 15.7%, recall of 50.4%, and F1 score of 24.0%. The Receiver Operating Characteristics (ROC) curve for driver verification is shown in Fig. 5.14. ROC curves represent how well models are able to distinguish between the classes. Our proposed system can achieve an AUC of 83.30% and an EER of 24.7%.

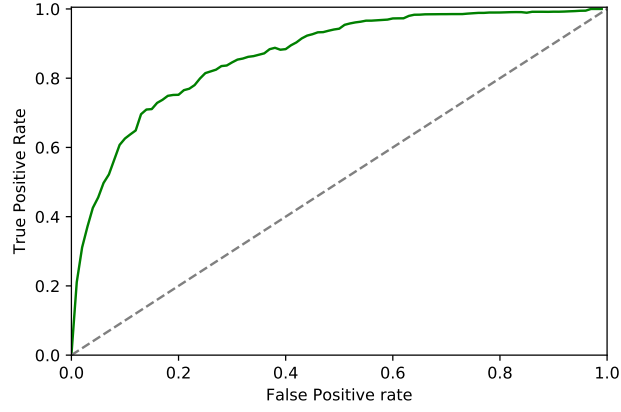


Figure 5.14: ROC curve for driver verification.

We can compare the obtained results from our introduced driver verification system to those (precision: 79.8% , recall: 69.1%, F1 score: 74.1%, AUC: 78.8%, and EER: 28.9%) of another state-of-the-art model [164]. However, it is noteworthy that there exist several major differences between the two methodologies. First, we take advantage of one-dimensional temporal convolutions in our system architecture, hence reducing the computational complexity and omitting extensive preprocessing steps including the need for transforming the data into images, which is adopted by the comparison model. Second, the results for the comparison model [164] are achieved for a group of 25 drivers based on smartphone accelerometer signals, whereas we perform the verification task over the steering behavior of the drivers.

To analyze the obtained results in more detail and investigate the existing relationships among drivers, we visualize driver encodings (i.e., vector features of latent representations) extracted for the drivers. To this aim, we use the well-established data visualization algorithm, namely t-distributed stochastic neighbor embedding (t-SNE) [287]. In Fig. 5.15, we present the extracted encodings (in two-dimensional latent space) for two sample groups of drivers each of which contains driving instances from two distinguished drivers. The encodings for the two selected drivers (d^2 and d^{49}) that the system finds hard to distinguish are shown on the left. Lower verification performance can be due to the fact that the encodings of these two drivers are clustered in the same region of the latent space, indicating high similarities in their steering behavior. The right subplot of Fig. 5.15 represents the encodings for two drivers (d^8 and d^{35}) for which the system has achieved higher results.

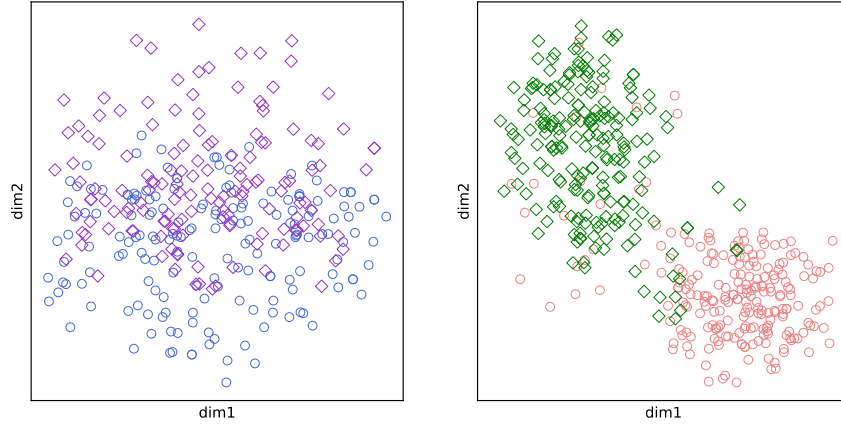


Figure 5.15: Two-dimensional projection of the encodings for drivers that the model had (left) and had not (right) difficulty verifying.

We also conduct an experiment to assess the performance of our introduced impostor detection approach. Recall that, in this problem, the primary objective is to check whether the individual who is driving the vehicle at the moment is within a group of authorized drivers or he/she is an impostor. To set up this experiment, it is required to consider a group of drivers as authorized drivers and a driver as an impostor. Considering n as the number of drivers within a group, there are $\binom{94}{n}$ possible combinations to select from (one driver selected as an impostor is excluded from training). Furthermore, considering each authorized driver within the group denoted as d , there are $\binom{93}{n-1}$ possible selections that include driver d and $\binom{93}{n}$ possible choices that do not include driver d . To reduce the high number of test variations for each group size, for each driver, we randomly select half of those that contain the driver and half of those that do not.

We evaluate the impostor detection system using four different values of $\{2, 3, 4, 5\}$. We also compare the obtained results from our system with those of a replicated impostor detection benchmark [171], which is a single layer feed-forward network (15 neurons). Table 5.4 summarizes the evaluation results for different groups of drivers. As can be seen, the proposed methodology outperforms the benchmark across all shown groups of the authorized drivers. For both models, the higher the number of drivers within the group, the lower the accuracy. For example, starting with an accuracy of approximately 73% for $n = 2$, the performance then drops 12.4% to almost 61% for $n = 5$. Furthermore, in another attempt for impostor detection, Yang *et al.* [181] report an accuracy of 83.1%, 73.8%, 67.3%, 62.5% for $n = 2, 3, 4$ and 5, respectively. Nevertheless, they use 31 different

Table 5.4: Evaluation results of an open-set setting impostor detection for different number of drivers.

Drivers	Models	
	Our proposed	Benchmark [171]
n=2	73.3	69.9
n=3	68.7	62.1
n=4	61.3	59.7
n=5	60.9	54.6

driving/driver-related signals. These drops are because of the fact that the system should handle more styles of steering behavior when there are more drivers in the systems. This fact will make it harder for the systems to distinguish verified drivers from impostors. Compared with driver identification and verification results, the models demonstrate a lower performance as, in this scenario, the impostor driving data (30s of driving data) is only available during the detection time.

5.4 Concluding Remarks

In this chapter, a deep learning-based driver identification system architecture is proposed that benefits from driving behavior analysis in general, and steering behavior in particular. The driver verification and impostor detection tasks are also addressed using a novel strategy based on extracting latent representations. Overall, we can summarize the following conclusion points based on the obtained results in this chapter:

- The proposed system architecture is able to successfully capture the distinguishing characteristics underlying within the data. The steering behavior of the drivers is analyzed to find these unique features and identify the drivers. The identification results on a dataset of 95 drivers revealed that the proposed methodology outperforms state-of-the-art approaches in terms of top-1, top-3, top-5 accuracy rates. One of the main advantages of the proposed system architecture is benefiting from one-dimensional temporal convolutions that can, first, eliminate the need for extensive preprocessing steps such as transforming the one-dimensional data to

two-dimensional data, and second, reduce the computational complexity by factor of 3 or 4 in comparison with recurrent neural networks architectures.

- The driver verification method is proposed based on the combination of our system architecture and Siamese networks concept. It is shown that the proposed approach satisfactorily manage to extract driver encodings and later use them to verify the individuals. The experimental results demonstrate the effectiveness of the proposed approach. The main advantage of the proposed approach is its suitability for scenarios in which there is a small number of driving instances (average of 25) available for each driver within the system. To verify a driver, a short interval of 30s is employed.
- The extension of the driver verification approach is used to detect any potential impostors based on driver encodings in an open-set setting. In comparison with a replicated benchmark method, it is disclosed that the proposed strategy can achieve higher performance.

Part II

Agents Behavior Prediction for Autonomous Vehicles

Chapter 6

Driver Intention and Vehicle Behavior Prediction at Unsignalized Intersections

In the preceding chapters, our focus was on the analysis of driving behavior for ADAS. Through characterizing driving behavior, we enhanced data dependency and the generalization capabilities of driver identification approaches. As the autonomy of AVs evolves from ADAS to full automation, our primary objective shifts towards operating without human intervention. In this context, developing situational awareness becomes of utmost importance. Accurate and reliable prediction of the future intentions and behaviors of nearby agents, along with a comprehensive understanding of the surrounding environment, holds the promise of greatly enhancing the quality of motion planning for automated vehicles, particularly in uncertain conditions. This predictive capability forms a crucial foundation for ensuring safe and effective traffic management in real-world deployments. However, this task becomes notably more complicated within highly dynamic and complex scenarios, such as unsignalized intersections where no lights exist to control vehicles behavior, or there are not multiple lanes for the vehicles to anticipate drivers' future intentions based on the lane in which they are driving. In these contexts, anticipating drivers future intentions and behaviors demands innovative approaches, enabling safer and more efficient navigation through such complex scenarios.

This chapter addresses multiple pivotal aspects of improving driver intention and behavior prediction at unsignalized intersections for AVs. Our key objectives in this chapter are summarized as follows:

- **Enhancing Driver Intention Prediction.** We propose a novel deep learning-driven approach that focuses on predicting driver intentions at unsignalized T-junctions. Utilizing the power of Temporal Convolutional Networks (TCNs), we leverage sequences of trajectory data to classify a driver’s intended maneuver seconds before its execution. Notably, to the best of our knowledge, this work is the first to benefit from the sequence modeling ability of the TCNs for driver intention prediction task.
- **Advancing Interaction-Aware Behavior Prediction.** We present an innovative framework that revolves around interaction-aware behavior prediction for AVs. Taking advantage of the capabilities of Graph Convolutional Networks (GCNs), our model effectively captures the complex connections between vehicles. This interaction-driven approach is extensively evaluated using a large-scale real-world dataset, comprising 14,867 vehicle trajectories. This effort adds to better comprehension and anticipation of complex driving situations, an essential step towards safe and effective deployment of autonomous vehicles.
- **Boosting Multimodal Vehicle Behavior Prediction.** We introduce a novel deep learning-based methodology to anticipate vehicles behavior at unsignalized intersections. The method provides multimodal outputs to take into account the inherited uncertainty and multimodality nature of vehicles behavior. Our proposed model works based on dilated convolutional networks in combination with a mixture density layer. We then cluster various existing mixes into possible behaviors that are ranked based on probability. We assess the performance and generalization capability of our vehicle behavior prediction model using several metrics over a large naturalistic dataset containing more than 23800 vehicle trajectories.

The rest of this chapter is organized as follows. Section 6.1 outlines our proposed driver intention prediction model, along with the problem statement, dataset used, data preprocessing, and evaluation outcomes. In Section 6.2, our behavior prediction model is introduced, addressing the challenge of social reasoning in complex scenarios. The dataset used and evaluation results are also covered. Section 6.3 explains our novel multimodal behavior prediction model, along with the evaluation setup and results obtained. Finally, Section 6.4 offers concluding remarks and a summary.

6.1 Driver Intention Prediction

Replacing human intuition and experience with autonomous intelligent inference systems is exceptionally challenging. Safe deployment of autonomous vehicles on public roads requires the ability to understand the current surrounding states and predict the future intentions of other road users, such as pedestrians, human-driver vehicles, and other autonomous vehicles. Consequently, driver intention prediction plays a vital role in increasing the effectiveness of autonomous vehicles as well as facilitating the interaction between them and human-driven vehicles. Driver intention of a target vehicle can be anticipated using either the data collected by its own sensors and then transferred to the surrounding autonomous vehicles using any available vehicular communication systems [301–304], or the data collected using a perception systems [305]. Several studies have addressed driver intention prediction task in different environments, including lane change detection [306] in highway scenarios or maneuver classification [307] at signalized intersections. Nevertheless, predicting driver intention appears to be substantially challenging at unsignalized intersections where there are no traffic lights to control the flow, or there exists only one lane (i.e., there is no turning lane).

This section addresses driver intent prediction at unsignalized T-junctions where there are two possible intentions for each origin position. The main objective is to anticipate driver intention as early as possible while the vehicle is approaching the T-junction. We propose to analyze a short sequence of target vehicle trajectory data (i.e., speed, headings, and positions) and predict the future intention of the driver using TCNs. As a result, we designed a deep learning-based architecture using causal convolutions to serve this purpose. Figure 6.1 demonstrates the understudy T-junction located at Sydney, Australia. Different observations of vehicles trajectories are shown.

6.1.1 Problem Formulation

In this section, we illustrate and formulate the driver intention prediction problem. We further describe the dataset used for the evaluation as well as the required data processing steps.

We consider the intent prediction problem as a classification task where the labels are the future intention of the drivers according to the maneuvers understudy. Moreover, we consider the trajectory as a time series. In other words, instead of considering each data point individually, we will analyze a short interval of data points (i.e., consecutive observations) to better capture the underlying features within trajectory data. As a



Figure 6.1: A sample of vehicles trajectories within the unsignalized T-junction.

result, we formulate intention prediction task as finding the best mapping from target vehicle data V to the intentions I using a collection of data pairs denoted as $D = \{(V_1, I_1), (V_2, I_2), \dots, (V_K, I_K)\}$ where K is the number of existing samples for vehicle maneuvers or driver intentions. Based on this definition, a sample k of the training data can be shown as $Sample_k = (\{v_{k1}, v_{k2}, \dots, v_{kn}\}, I_k)$ where n is the number of time steps for a sample maneuver. The length of the samples can be different for each maneuver according to the number of collected observations. It is also worth mentioning that each v_n includes the records of all used features, denoted as F , at time step n which can be denoted as $v_{k1} = \{f_1, f_2, \dots, f_F\}$.

In the following, we will explain the T-junction understudy as well as the feature settings for the employed dataset.

Dataset and Data Preprocessing

The naturalistic driving dataset [278] used in this chapter is collected in Sydney, Australia. Note that drivers drive on the left side of the roads in Australia. The side of the road the target vehicle is driven on is important for correct analysis of the driving behavior of drivers, especially for turning maneuvers. The released original dataset contains three long-duration trajectories driven by three different drivers. Each trajectory includes data about the position of the vehicle, its speed, heading, and the recorded timestamps that are collected through GNSS, IMUs, and wheel encoder sensors. There

Table 6.1: Extracted maneuvers according to approaching direction-destination pair. For instance, the term East-South refers to east as origin and south as destination

Drivers	Maneuvers						Total
	East-South	East-West	West-South	West-East	South-East	South-West	
One	10	13	10	13	11	11	68
Two	14	11	11	10	14	10	70
Three	9	10	10	10	10	10	59

were no special instructions for the driving behavior/style of the drivers. The only required instruction was the number of times drivers should perform each maneuver passing the T-junction.

As the first step of the data preprocessing, we have limited the trajectories data to the desired T-junction area shown by a green circle in Fig. 6.1 for all drivers. For this purpose, we have selected data within 60 meters from the center of the T-junction for each direction and discarded those outside the shown circle. This distance is chosen in a way that guarantees the driver is not in the middle of the maneuver execution. We then manually extracted different types of maneuvers and labeled them accordingly. There exist two possible driver intentions for each direction since the desired intersection is a three-way junction, adding up to 6 maneuvers in total. For instance, considering the driver is approaching the intersection from the south, the driver can either turn left or right. Each maneuver is labeled according to the direction from which the vehicle approaches and exits the T-junction. Figure 6.2 demonstrates the extracted maneuvers (i.e., going straight and turning to the right or left) within the area of interest for all three trajectories. Northward and eastward-measured distances are calculated relative to the center of the T-junction. There are 65, 68, and 64 maneuvers whose destination is west, east, and south, respectively. Table 6.1 provides details about the number of extracted maneuvers grouped by the driver of the target vehicle. As summarized, each driver performs all six maneuvers more than ten times (with the exception of driver three for the east-south one).

Moreover, as shown for all three trajectories in Fig. 6.2 (note that the data are rotated for visualization purposes), drivers have started to commit a specific maneuver from approximately $d = 20$ meters before the center of the T-junction in the used dataset. The distance d can be employed for further analysis and evaluations. Since this distance appears to indicate a threshold where a specific maneuver has started, classification prior

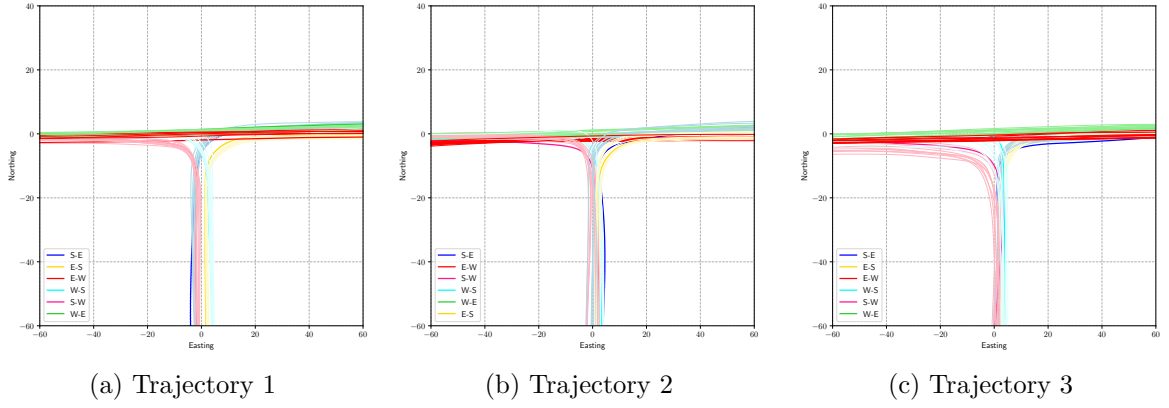


Figure 6.2: Extracted maneuvers from each trajectory according to the relative position from the center of the T-junction (i.e., (0.0) refers to the center of the intersection) denoted by origin-destination pair. The first sample for each maneuver is shown in bold color. The relative positional data are rotated for visualization purposes.

and after d is of utmost importance. Also, Fig. 6.3 illustrates different used features (i.e., the positions, heading, and velocity) for all extracted maneuvers originated from the south direction of the T-junction. Again, positional data are rotated for visualization only. As discussed earlier and shown in Fig. 6.3a from approximately 20 meters before the T-junction center, relative positions are separating, showing the left or right turning maneuver. The speed observations relative to the distance of 20 meters of the center for the same maneuvers are shown in Fig. 6.3b. The negative distances demonstrate that the driver of the vehicle is moving toward the intersection. A vehicle coming from a south origin should reduce its speed before making a turn, and it is quite likely to stop to yield right of way for both maneuvers, making it harder to predict the drivers intention. The headings values, however, as illustrated in Fig. 6.3c, diverge considerably after the distance d from the T-junction center.

6.1.2 Methodology

As described in Section 6.1.1, we consider the driver intent anticipation problem as a classification task in which the past vehicle trajectory for a specific maneuver is the input of the model, and the future intention of the drivers are the labels/outputs. To anticipate the intention of the driver, we need to model the long-term temporal dependencies of the driving behavior and the vehicles maneuver. To do so, we will take advantage of

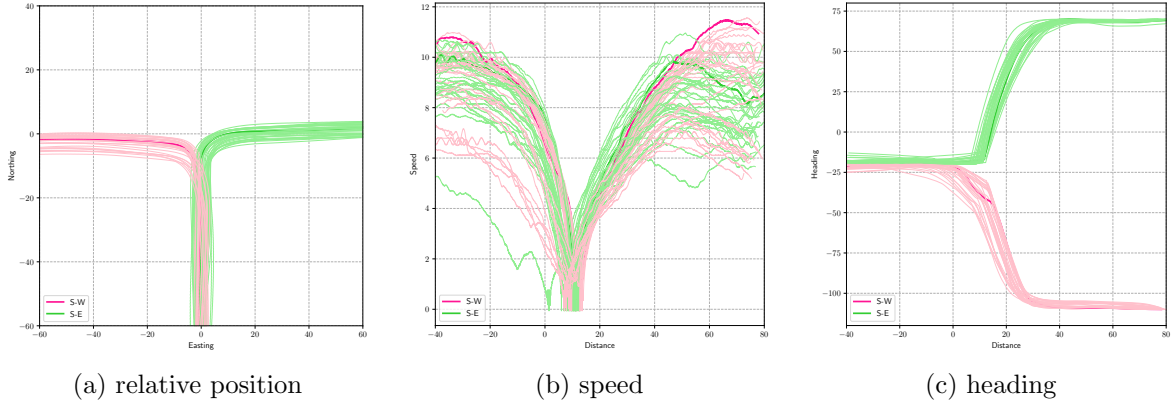


Figure 6.3: Relative position, speed, and heading of the data for all maneuvers originated from the south (i.e., south-west and south-east). The first sample of each category is shown in bold. Smoothed speed values are calculated for visualization.

the temporal convolutional network as they have proven [308] to have the capability to capture the temporal characteristics of the sequential series in other applications such as natural language processing or digit classification.

The primary purpose of the driver behavior prediction is to estimate the potential maneuver a driver is most likely to perform at the T-junction before the actual occurrence of the maneuver, given some previous driving and vehicle-related information with respect to time. We can then consider the consecutive sensor observations obtained from GPS or IMUs as time series. Therefore, these observations can be given to a model sequentially as a time series sequence with an arbitrary length, each of which represents a specific intention/maneuver. In our defined problem and according to the used dataset (nature of a T-junction), which is explained in Section 6.1.1, these maneuvers include keeping going straight, and turning either left and right, which we have labeled based on the origin and destination direction at T-junction (e.g., South-East with South being the origin of the maneuver). A short window of observations is fed to the introduced model to estimate the potential drivers intention before it actually happens. This short interval can be denoted as follows:

$$S_k = [v_{k1}, v_{k2}, \dots, v_{kn}] = \begin{Bmatrix} f_1[0] & f_2[0] & \dots & f_F[0] \\ f_1[1] & f_2[1] & \dots & f_F[1] \\ \vdots & \vdots & \ddots & \vdots \\ f_1[n] & f_2[n] & \dots & f_F[n] \end{Bmatrix} \quad (6.1)$$

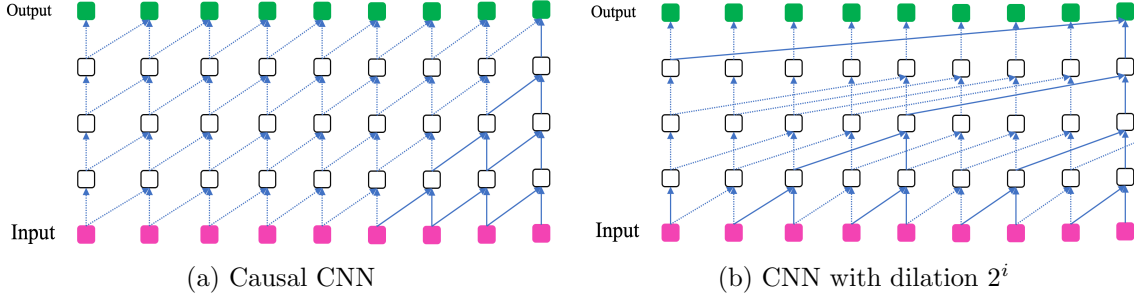


Figure 6.4: Difference between the architecture of causal CNNs with and without dilation.

We now explain the choice of our deep learning-based architecture. Since the advent of convolutional neural networks (CNNs), they have proven to be extremely powerful for a wide variety of applications, especially for image processing and classification [118]. The main characteristics of a CNN which differentiate it from other types of neural networks are the existing shared weights, local connections, and the sub-sampling feature with respect to time or space in its architecture. Since, in our application, the sensor data can be considered as time series, which are one-dimensional as opposed to two-dimensional images, we then focus on one-dimensional CNNs (causal CNN). The main feature of a causal CNN is that this model does not allow any information from the future observations to be used in the calculation at the current time. This feature is of significant importance for time series tasks such as forecasting and classification. Consider now a 1-D input $s = (s_t)_{t=0}^{N-1}$ with the length of N . Its corresponding feature map extracted from the first layer can be processed as:

$$(g_h^1 * s)(i) = \sum_{j=-\infty}^{\infty} g_h^1(j)s(i-j) \quad (6.2)$$

where $*$ is the convolution and g are the filters. For the next layer, however, the output is computed as:

$$(g_h^l * R^{l-1})(i) = \sum_{j=-\infty}^{\infty} \sum_{m=1}^{M_{l-1}} g_h^l(j, m)R^{l-1}(i-j, m) \quad (6.3)$$

where R are the fed feature maps to the current layer. One shortcoming of such architecture is its limited receptive field, which is later addressed using dilation concepts [288]. With dilation, we can expand the receptive field (i.e., the points from the input of the

CNN that have an impact on the output) of a CNN. The main difference between the architecture of a causal CNN with and without dilation is shown in Fig. 6.4. As indicated in Fig 6.4a, the local connections of a typical causal CNN are limited than that of the one with dilation (see Fig. 6.4b) with respect to time. Different participants for calculating the first output are shown in solid lines. For instance, if we consider the second and third hidden layer, in causal CNN, the first and second (from left) inputs in the second layer can have an impact on the first output in the third hidden layer. On the contrary, in a CNN with a dilation of $q = 4$, the filter is applied on the first and the fifth, allowing the model to capture long-term correlations effectively. Consequently, by taking advantage of dilation, we can capture longer temporal data dependency. The feature map of a CNN with dilation is computed as follows:

$$(g_h^l *_q R^{l-1})(i) = \sum_{j=-\infty}^{\infty} \sum_{m=1}^{M_{l-1}} g_h^l(j, m) R^{l-1}(i - q \cdot j, m) \quad (6.4)$$

where q and M_l are dilation factor and the number of channels, respectively. A factor of $q = 4$ means that the filter will be applied only on every four element from the input. This factor is increased every layer exponentially (2^l) to go back more in time. As demonstrated in Fig. 6.4b, the first layer has $q = 1$ whereas $q = 4$ for the third hidden layer in the shown sample network. As a result, the filter is applied to every four elements.

Therefore, we will use the dilated type of CNNs in our proposed prediction approach. We implemented three layers of stacked CNNs with dilation with ReLU activation to analyze drivers intentions. Also, to avoid over-fitting, we used spatial dropout between the layers.

To implement the proposed model, we used Python 3 with Tensorflow. Our model is validated using cross validation while training for about 14 minutes on average. A summary of parameter settings that are obtained using a grid search and used in this section is presented in Table 6.2.

6.1.3 Driver Intention Evaluation Results

We assessed the performance of our intent classification proposed approach using 5-fold cross-validation. We consider the accuracy of the classification as the evaluation metrics and the fact that how early the model can achieve 100% with respect to the distance from 20 meters before the T-junction center. Furthermore, we also compared

Table 6.2: Parameters settings for the introduced TCN-based model used for experiments

Parameters	Parameters settings
Learning rate	0.002
Number of stacks	3
Decay rates	(0.9, 0.999)
Dilation	{1,2,4,8,16}
Kernel size	3
Normalization	weight
Epochs	50

the obtained cross-validation intent prediction results with those of another benchmark methodology, namely QDA [278]. We will describe the main principles of this model in brief. QDA model is a Gaussian discriminant-family probabilistic classifier that works based on Bays theorem. In this family models, the posterior probabilities are:

$$Pr(y = I | v) \propto \pi_I N(v | \mu_I, \Sigma_I) \quad (6.5)$$

where the π values are the priors. Considering a classification problem with I number of labels (i.e., number of intentions/maneuvers in our case), QDA considers a different covariance matrix for each existing class which can be calculated as follows [278]:

$$\hat{\Sigma}_I = \frac{1}{N_I - 1} \sum_{y_i=I} (v_i - \hat{\mu}_I)(v_i - \hat{\mu}_I)^T \quad (6.6)$$

where $\hat{\mu}$ are the means of the Gaussians and can be obtained as:

$$\hat{\mu}_I = \frac{1}{N_I} \sum_{y_i=I} v_i \quad (6.7)$$

Figure 6.5 represents average accuracy results of 5 fold cross-validation for the scenarios where the vehicles are moving toward the T-junction from the south direction. The distance values are relative to 20 meters from the intersection center. In this scenario, the drivers reaching the intersection should perform a right or left turning maneuver. As a result, drivers need to reduce their speed or even stop to give the right of way, which makes it more challenging to estimate the future intent of the driver (see Fig. 6.3b). As

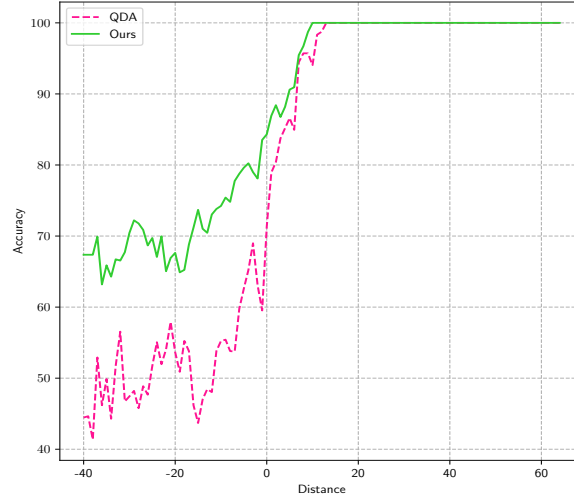


Figure 6.5: Intent prediction results for the scenarios where vehicles are moving toward the T-junction from south.

shown in Fig. 6.5, the proposed methodology outperforms the QDA model. The improvements are more considerable for distances farther from the T-junction center. For instance, considering approximately 20 meters before the vehicle enters the T-junction, our model achieves an accuracy of about 66% while the QDA model has an accuracy of almost 54%. Moreover, our model reaches 100% accuracy earlier in time, from a distance of 9 meters after the drivers enter the T-junction. This distance is 12 meters for the QDA model.

Figure 6.6 illustrates cross-validation average accuracy results for the east-origin scenarios where the vehicles are reaching the T-junction from the east direction. A driver in this scenario is allowed to turn left to the south or continue forward to the west of the T-junction. Unlike the previous scenario where the driver may need to come to a complete stop at the intersection to turn left, the driver does not need to yield the right of way in this scenario. This is due to the nature of the T-junction used in the analysis. The T-junction used in the experimental evaluation is located in Australia, where the vehicles are driven on the left side. As can be seen, our model reaches 100% accuracy approximately 5 meters after the vehicle has entered the intersection. This distance is 7 meters for the QDA model. Furthermore, compared to the result for the south-origin scenario, both models demonstrated higher average accuracy results, indicating more distinguishable characteristics in the speed of the vehicles for both maneuvers. For instance,

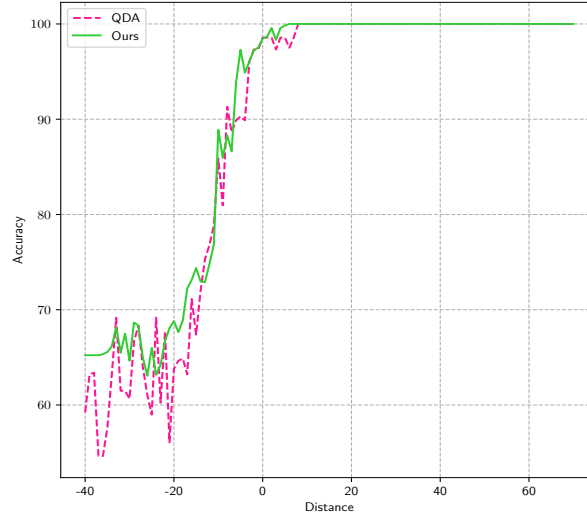


Figure 6.6: Prediction cross-validation results for the scenarios where vehicles are reaching the T-junction from east.

considering the distance -10 meters, our model achieves an accuracy of approximately 85% which is noticeably better than 75% obtained for the south-origin scenario.

Evaluation results for the vehicles approaching from the west of the T-junction are demonstrated in Fig. 6.7. In this scenario, a vehicle reaching the intersection can either continue its path straight or commit a right-turn maneuver. Similar to the east-origin scenario, the driver of the vehicle may not hit the brake if they are not going to turn. However, since the dataset is collected in Sydney, the driver may have to come to a complete stop to yield the right of way to the vehicles passing from the opposite direction. Consequently, such changes are reflected in the speed characteristics of the vehicles as illustrated in Fig. 6.7b, making the two driver intentions in this group (i.e., west-south and west-east) more distinguished. As indicated in Fig. 6.7a, our model outperforms the QDA. For instance, considering a distance of -20 meters, our model average accuracy reaches almost 82%, whereas the QDA model achieves an accuracy of 75%. Both models manage to predict the driver intention with 100% as early as about 5 meters after the vehicle enters the T-junction.

From the obtained results for all groups, it is evident that the speed of the vehicles plays an important role (i.e., more than the headings and the positions) in the ability of the classifiers to distinguish different maneuvers from one another. This fact can be seen when comparing the speed profiles of all drivers while performing the south-origin

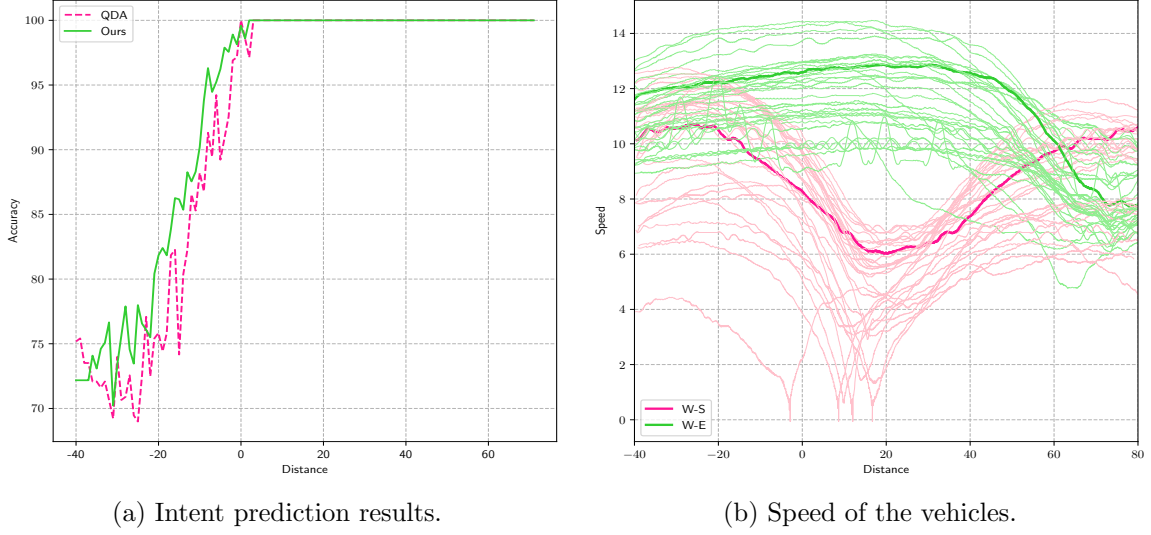


Figure 6.7: The scenarios where vehicles are approaching the T-junction from west. Top: the prediction accuracy result. Bottom: Speed of the vehicles for both maneuvers. One sample of each category is shown in bold. Smoothed speed values are calculated for visualization.

maneuvers (see Fig. 6.3a) and performing the west-origin maneuvers (see Fig. 6.7b). As indicated, for south-origin maneuvers from the distance d to the T-junction center (i.e., from 0 to 20 on the graph), it is needed to gradually reduce the velocity of the vehicles, or even stop (0 speed) in order to execute the desired turning maneuver for both directions. On the other hand, the speed profiles of the vehicles going south from the west of the T-junction are more distinguishable from the speed profiles of the vehicles that are heading east from the same origin.

Moreover, evaluation results revealed the capability of our method in maneuver classification, especially in distinguishing the maneuvers which are more challenging to predict, such as turning left or right at an unsignalized T-junction while vehicles are heading toward the north. Comparison between south-origin results for our model and the benchmark QDA model highlighted this capability. This is due to its ability to capture the long-term temporal characteristics of the trajectory of the vehicles.

6.2 Interaction-aware Driver Behavior Prediction

In the preceding section, our focus was on driver intention prediction, which involves anticipating the intended maneuver of a target vehicle. In this and the forthcoming sections, our attention shifts to behavior prediction. While foreseeing driver intentions offers insights into immediate actions, behavior prediction surpasses this by envisioning the potential future behavior of vehicles, delivering crucial data for navigation, collision avoidance, and overall traffic management. Nonetheless, a strong correlation exists between the behavior of a target vehicle and that of its surrounding vehicles, leading to a mutual influence on one another's behavior. Consequently, merely observing the past trajectories of the target vehicle falls short. To illustrate, consider the scenario of a target vehicle (TV) at an unsignalized intersection, as depicted in Fig. 6.8. The behavior of its surrounding vehicles (SV), particularly SV3, can impact its future behavior. Farther vehicles are shown as non-effective vehicles (NVs).

Observing and perceiving the behavior of all surrounding vehicles of a target vehicle and considering the interactions among them to implement an efficient behavior prediction model is still a challenging task for the ego vehicle, especially at less structured but highly dynamic scenarios such as unsignalized intersections. One limitation is that the driving environment is not fully observable by the sensors mounted on the ego vehicle due to occlusion and the limited range. Fig. 6.8 demonstrates an illustrative example of such a problem. To fully model the interactions between TV and its SVs, The ego vehicle needs to collect data from all nearby vehicles, including SV1, SV2, and SV3. However, SV2 and SV3 are within the occluded area of ego vehicle. While several previous studies have addressed inherent challenges, including behavior interdependency or multimodality nature of vehicle behavior for motion prediction of TV in structured areas like freeways and signalized intersections [11], there has been less focus on observability limitation of ego vehicle, particularly for complex scenarios like unsignalized roundabouts.

In this section, we address the above-mentioned challenge by introducing a novel interaction-aware and IoV-based methodology designed to improve the accuracy of vehicle behavior prediction. We can benefit from the Connected Automated vehicles (CAVs) and vehicular communications [309–312] to share information with the ego vehicle and analyze the SV and TV interactions using GCN, making the scene understanding process more informative. In particular, we take advantage of Spatio-temporal GCNs to better model the spatial and temporal interdependency between a TV and all of its SVs. The future behaviors are then anticipated based on temporal convolutions. As illustrated in

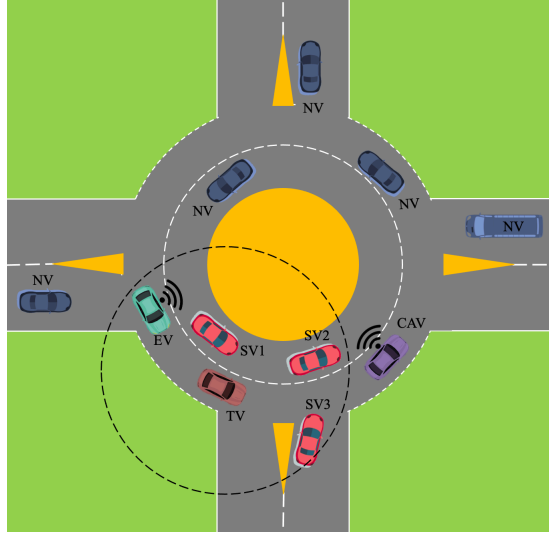


Figure 6.8: The interaction-aware prediction problem for CAVs i.e., the occlusion challenge.

Fig. 6.8, the ego vehicle can utilize the information about SV2 and SV3 obtained by sensors on the CAV and reduce the occlusion problem in the vehicles behavior anticipation context.

6.2.1 Problem Statement

This part will formulate TV behavior prediction problem for CAVs. The primary objective in a vehicle behavior prediction problem is to anticipate the upcoming behavior of a target vehicle according to its trajectory history that is observable before time t . Consequently, the problem under study can be considered a sequence modeling problem.

Vehicle behavior history, denoted as $P = [p_1, p_2, \dots, p_K]$, will be the input of the system, where K is total number of vehicles. There exist 14867 vehicle trajectories in our case (unsignalized intersections scenario). Each of these past trajectories from target vehicles consists of various informative features I for several time steps. If we consider M as the size of the given trajectories, accordingly, each one can be shown as:

$$p_k = [\vec{I}_1, \vec{I}_2, \dots, \vec{I}_j] = \begin{pmatrix} i_1[0] & i_2[0] & \dots & i_j[0] \\ i_1[1] & i_2[1] & \dots & i_j[1] \\ \vdots & \vdots & \ddots & \vdots \\ i_1[S] & i_2[S] & \dots & i_j[S] \end{pmatrix} \quad (6.8)$$

In the current section, we will take advantage of five different attributes related to the TVs and their surrounding ones, including lateral and longitudinal positions ($pos = (x, y)$), lateral and longitudinal velocity (v^x and v^y), and yaw angle (γ), which leads to $j = 5$. Therefore, the vehicles behavior history specific to our case of study can be shown as the following:

$$p_k = \begin{pmatrix} x[0] & y[0] & v^x[0] & v^y[0] & \gamma[0] \\ x[1] & y[1] & v^x[1] & v^y[1] & \gamma[1] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x[M] & y[M] & v^x[M] & v^y[M] & \gamma[M] \end{pmatrix} \quad (6.9)$$

Also, to be able to perform the prediction process, we need to split the current trajectories into different *observed*, and *predicted* intervals for all target vehicles. To do so, we consider an observation window of obs and the anticipation horizon of prd . Given a short interval of observation O , the ultimate objective of vehicle behavior anticipation is to find the upcoming positions (POS) of the TV.

$$O = [p_k^{t-obs+1}, p_k^{t-obs+2}, p_k^{t-obs+3}, \dots, p_k^t] \quad (6.10)$$

$$p_k^t = [x_k^t, y_k^t, vx_k^t, vy_k^t, r_k^t] \quad (6.11)$$

$$POS = [pos_k^{t+1}, pos_k^{t+2}, pos_k^{t+3}, \dots, pos_k^{t+prd}] \quad (6.12)$$

where $pos_k = (x_k, y_k)$ is the anticipated positions of the TV relative to the world space.

As discussed, the prime goal is to predict a TV's future behavior while considering its correlation with all of its SVs, even those within the occluded area of the ego vehicle. To do so, we assume that: (1) all CAVs are equipped with a GPS; (2) they can communicate using V2V wireless communications based on IEEE 802.11p; (3) they can observe the surrounding unoccluded environment through their mounted sensors. Consequently, the problem is how to correlate and fuse all this information together to gain knowledge about the future potential moves of the TV.

6.2.2 Methods

The broad view of the proposed methodology for anticipating the TV's future behavior is demonstrated in Fig. 6.9. There are two primary components within the introduced system: a spatial-temporal GCNs module to encode the features within the data and a temporal convolution module as a decoder to generate future trajectories. First, we

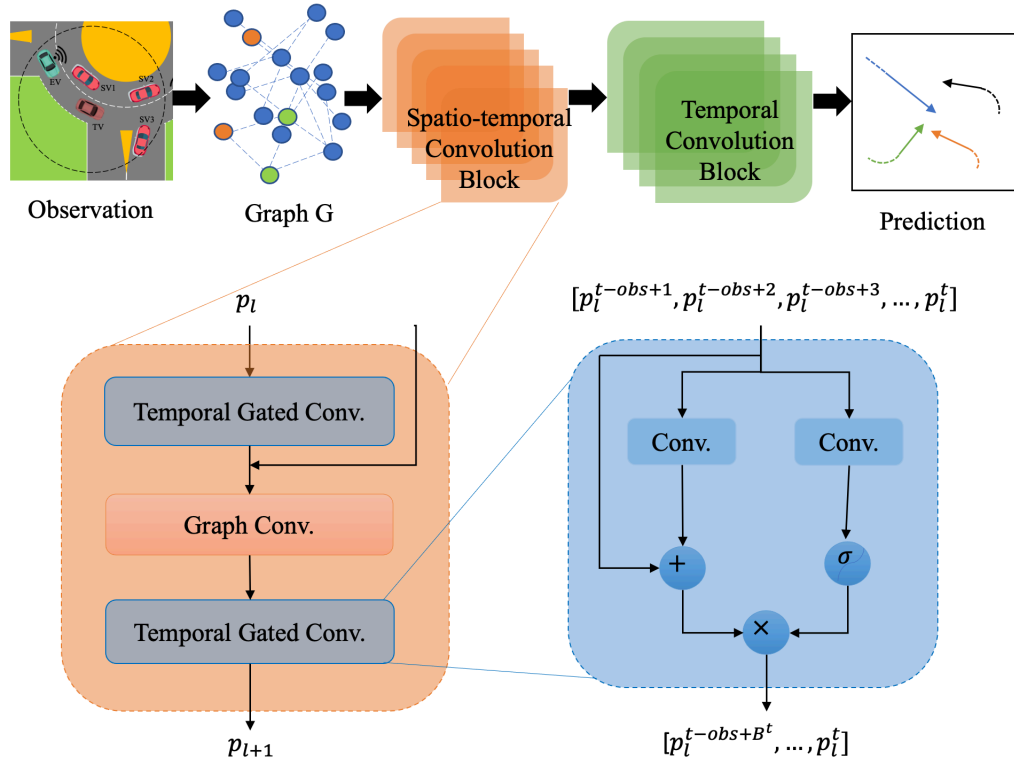


Figure 6.9: A broad view of the architecture of our vehicle behavior prediction system.

analyze the trajectories history to create an interaction graph G that models all vehicles' interactions within a driving scene. Information on vehicles within the occluded area can be transformed to the ego vehicle through V2V communications. Moreover, graph representation can handle the variable number of TVs in different scenarios. Next, we take advantage of spatial-temporal GCNs modules to extract the spatial correlations of the TV and all SVs as well as the temporal dependency within the data. Last, these extracted features by the GCN can be used to anticipate the future behavior of the TV with blocks of temporal convolutions. We further discuss the architecture modules in the following.

Spatio-Temporal Convolution Module

This section will discuss the components of the proposed Spatio-temporal convolution module in detail.

Graph Construction

Graph Neural Networks are introduced to capture the hidden features within the graph-structured data and explore the interdependency between the nodes [129,313]. As mentioned earlier, the future behavior of a TV can be affected by its SVs. For instance, consider a TV in a car following scenario. Here, the sudden brake of the leading vehicle may cause the TV to reduce its speed or change lanes. Consequently, we utilize graph structures to present the interdependency between TVs and their SVs in this chapter. This way, we can better capture the underlying behavioral patterns for a set of interactive vehicles. Given the past trajectories $[p_k^{t-obs+1}, p_k^{t-obs+2}, p_k^{t-obs+3}, \dots, p_k^t]$, we create an undirected graph $G^t = (V^t, E^t)$ at time step t , where V^t is the set of nodes representing the vehicles within a scene and $E^t = \{a_{ij}^t | i, j = 1, \dots, f\}$ is a set of edges that indicates vehicles' connectivity based on a weighted adjacency matrix A . In other words, we build a graph where the edges represent the strength of the vehicle influence on each other based on the distances between the vehicles in G (see the left side of Fig. 6.10). Therefore, the elements in A can be shown as:

$$a_{ij}^t = \begin{cases} 1/||p_i^t - p_j^t||_2, & i \neq j \\ 0, & \text{otherwise} \end{cases} \quad (6.13)$$

Graph Convolutions

Evidently, regular convolution operation is not applicable to the graph-structured data and some modifications are needed. We denote the graph convolution (GC) operation for a given graph signal s as $*_g$ which is calculated using Eq. 6.14:

$$x = \mathcal{K} *_g s = \mathcal{K}(U\Lambda U^T)s = U\mathcal{K}(\Lambda)U^T s \quad (6.14)$$

where $\mathcal{K}$ is the kernel, and Fourier basis $U \in \mathbb{R}^{F \times F}$ is the characteristic vector matrix of the Laplacian L , having the identity matrix I_F :

$$L = I_F - D^{-1/2}ED^{-1/2} = U\Lambda U^T \quad (6.15)$$

where the diagonal degree matrix D will be characterized as $D_{ii} = \sum_j E_{ij}$, and Λ is the diagonal matrix where $\Lambda_{ii} = \lambda_i$. Using these definitions, we can then filter a graph signal s through the multiplication of kernel $\mathcal{K}$ and Fourier Transform. An illustrative example is demonstrated in Fig. 6.10 (right) in which the GC is operated on the purple node.

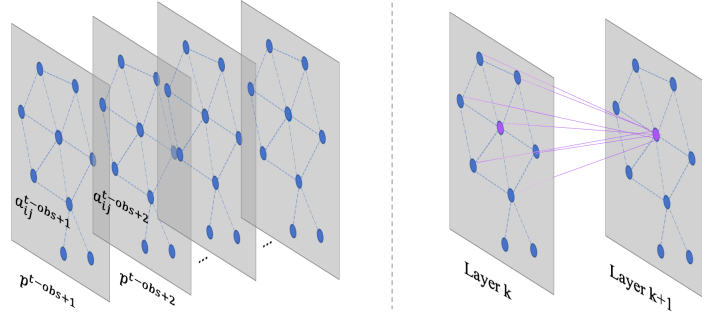


Figure 6.10: Graph-based modeling of TV and SVs interactions (left), and graph-based convolution process (right).

The GC operation mentioned above can be extended with multiple channels. Given a graph signal $S \in \mathbb{R}^{F \times C_i}$, the extended GC can be computed as:

$$X_l = \sum_{i=1}^{C_i} \mathcal{K}_{i,l}(U \Lambda U^T) s_i, \quad i \leq l \leq C_{out} \quad (6.16)$$

Gated Convolutions

From the temporal aspects, the future behavior of a TV is highly dependent on its trajectory history. For example, consider a TV in the middle of performing a maneuver such as a U-turn. This TV is most likely to complete its turning maneuver in the future. Moreover, the impact of SVs on the future behavior of a TV is time-varying which makes the temporal analysis important. We propose to analyze the temporal behavioral patterns of the trajectory data using Temporal Gated Convolutions (TGC). The main benefits of CNN-based design compared to those based on recurrent units are simplicity of the structure and less computational complexity. As illustrated in Fig. 6.9, the TGC layers contain one-dimensional causal convolutions to capture the temporal dependencies. For each vertex in G , TGC explores the $\mathcal{B}^t$ elements. This process does not involve padding, meaning the sequence size will be reduced $\mathcal{B}^t - 1$ each time. If we consider an input $X \in \mathbb{R}^{L \times C_i}$ for each vertex in G where L is the size of the sequence, the kernel ψ , at this module, will map the X to the output of the TGC. More formally, we can calculate TGC operation denoted as $*_{\mathcal{T}}$ by Eq. 6.17:

$$\psi *_{\mathcal{T}} X = Q \odot \sigma(R) \in \mathbb{R}^{(L-\mathcal{B}^t+1) \times C_{out}} \quad (6.17)$$

where Q and R are the inputs of the last gate.

Spatio-Temporal Block

As demonstrated in Fig. 6.9, spatio-temporal block directly analyzes the graph-structured data. It captures the underlying vertex attributes according to the past trajectories and spatial interaction among the vehicles. Each block consists of two temporal layers with a spatial layer in between as described in graph and gated convolutions sections. Given the Spatio-temporal block l and the input p_l the output will be calculated as:

$$p_{l+1} = \psi_l^1 *_{\mathcal{T}} \mathcal{F}(\mathcal{K}_l *_{\mathcal{g}} (\psi_l^0 *_{\mathcal{T}} p_l)) \quad (6.18)$$

where ψ_l^i , $\mathcal{K}_l$, and $F(\cdot)$ are the kernels for GC and TGC modules of l^{th} block and activation function, respectively. Furthermore, as for the training loss, we consider maximizing the log-likelihood of future trajectories and minimizing the distance between the anticipated and true values using Eq. 6.19.

$$\begin{aligned} Loss = & - \sum_t \log(\mathbb{P}(pos_g^t | \hat{\mu}^t, \sigma^t)) + \\ & 1/prd \sum_t ||pos_g^t - pos^t||^2 \end{aligned} \quad (6.19)$$

where pos_g is the ground truth positions of the TV.

Temporal Convolution Module

As mentioned earlier, we have now learned the spatial interdependencies among TVs and SVs through GC operation over the inputs and captured their temporal dependencies by TGC operation. In the next step, we are required to analyze these features to generate future trajectories. We will perform this task using our temporal convolution module. This module consists of stacked layers of temporal CNNs (one-dimensional) directly applied to the features extracted by the previous module. These stacked temporal CNNs are connected using residual connections.

6.2.3 Evaluation Results

In this part, we assess the introduced approach for vehicle behavior prediction and report the comparison results with state-of-the-art baselines with respect to two different metrics. The method is implemented using Pytorch library and Adam optimizer. The batch size is 64, and the learning rate starts with the amount of $1e^{-3}$, which is increased every 40 epochs. The observation window is 10 frames. The anticipation horizon is up

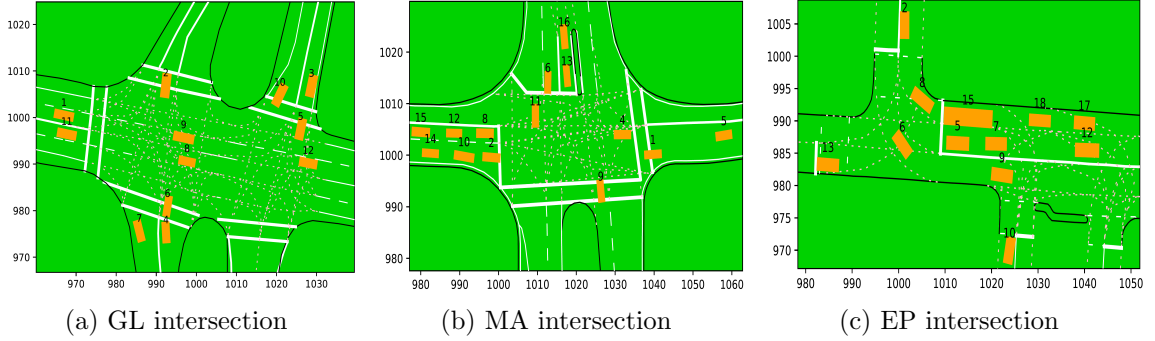


Figure 6.11: Example cases from the unsignalized intersections. The positions of the TVs are relative to the world space.

to 30 frames, which indicates a trajectory history of 1s and the future behaviors of 3s according to the data available within the used dataset.

Dataset

We evaluate the proposed framework over a large real-world dataset [82]. The full dataset contains several driving and environment scenarios, and the sampling rate is 10 Hz. We truncate the dataset to select the less structured but highly interactive scenes such as unsignalized intersections. The total number of agents within the used scenes is more than 14000 for each of which 4s of data is available. The High-Definition Maps (HD-Maps) are also provided. Figure 6.11 demonstrates three illustrative examples of complex and crowded driving scenarios from the dataset.

Comparison models and metrics

We draw a comparison between the prediction results of our method with those of four other models. Each model is discussed in brief.

- *Social-LSTM* [233]: This model contains an LSTM module for each vehicle. The interactions among a TV and its SVs are modeled as a connection between these LSTM modules by a social pooling approach.
- *Social-GAN* [134]: This model consists of a generative adversarial-based encoder-decoder architecture. It combines the sequence modeling capabilities of recurrent neural networks with GANs.

Table 6.3: Comparison results for different approaches.

Models	ADE			FDE		
	1s	2s	3s	1s	2s	3s
Social-LSTM [233]	0.12	0.24	0.43	0.18	0.43	0.86
Social-GAN [134]	0.13	0.22	0.39	0.15	0.39	0.79
D-LSTM [314]	0.14	0.27	0.39	0.18	0.42	0.82
ML-LSTM [39]	0.16	0.27	0.48	0.20	0.52	0.88
Proposed approach	0.09	0.19	0.34	0.14	0.30	0.71

- *D-LSTM* [314]: The trajectory histories are analyzed using an LSTM module and six FC layers.
- *ML-LSTM* [39]: This model is a recurrent network that consists of three layers of hyper LSTM units. We adopt the model to extract a unimodal prediction (the one with the highest probability).

The results are evaluated using two metrics defined as follows:

1. *Average Displacement Error (ADE)* measures the MSE error for all anticipated positions and their true values at each time step.
2. *Final Displacement Error (FDE)* measures the MSE between the anticipated and true value at the final anticipated time step.

Quantitative and Qualitative Analysis

Table 6.3 summarizes the quantitative comparison results. We report both ADE and FDE for our approach and all other state-of-the-art models. As illustrated, the introduced approach outperforms all these benchmarks for all prediction horizons. Note that the model obtains the lowest error value, representing the potential capability of our approach to extract both spatial and temporal interdependencies between the vehicles. More specifically, our model improves the behavior prediction performance by 25%, 13.6%, and 12.8% on ADE for horizons 1, 2, and 3s, respectively, compared with the previous

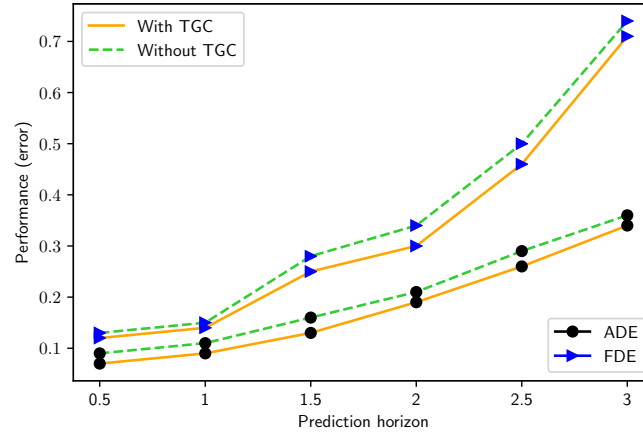


Figure 6.12: Ablation analysis results for the proposed method.

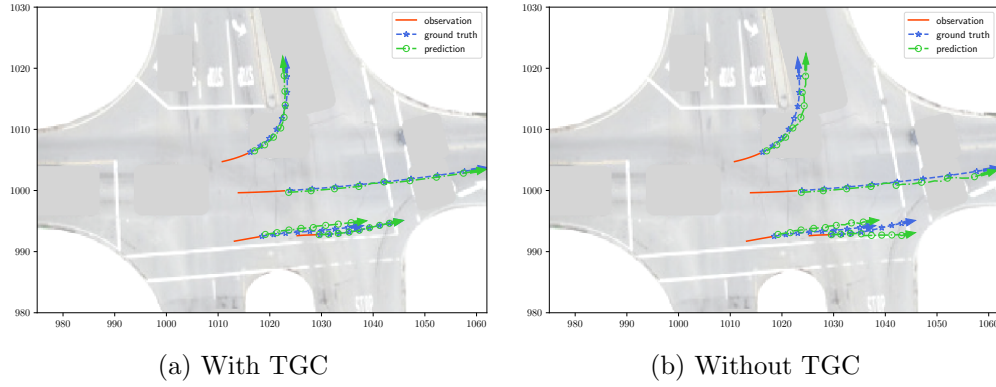


Figure 6.13: Qualitative results over the MA intersection.

baseline. The model also demonstrates an average improvement of about 13% on FDE. Social-LSTM benefits from a pooling technique to model the TV and SV interactions. Social-GAN improves the results by using a variety loss. Unlike the Social-LSTM, the pooling technique is not performed at every time step. The ML-LSTM model achieves the highest error values since this model does not consider the interactions among a TV and its SV while predicting the future trajectories. We further perform ablation analysis for the proposed model. Figure 6.12 illustrates how considering both spectral and temporal aspects can improve the performance.

We also qualitatively evaluate the performance of the model by visualizing the produced future trajectories in Fig. 6.13. Based on the results, we can observe that the model can accurately anticipate the future behavior of the TV and SVs within a scene. The

model can handle different scenarios within an unsignalized intersection, such as turning and going straight. As shown, the model is also able to perform satisfactorily when the behavior direction changes significantly (e.g., the vehicle turning left). Also, the comparison between Figures 6.13a and 6.13b demonstrates that the anticipated trajectories of the model with TGC are smoother and more probable.

6.3 Multimodal Vehicle Behavior Prediction

Efficient deployment and safe operation of an automated vehicle require high-level and efficient perception and motion planning. This task involves both understanding the current states of other surrounding road users and the continuous prediction of their potential behavior. There are two aspects regarding this problem. The first aspect is to detect pedestrians and anticipate their future trajectories [315–318] which can be extremely beneficial to collision avoidance and risk assessment of automated vehicles. The other aspect is anticipating the future behavior of the surrounding target vehicles by the ego vehicle (i.e., the automated vehicle). Taking advantage of V2V communications [302, 319–324] can be a potential solution to anticipate a vehicle behavior and transfer it to the ego vehicle. Nonetheless, this solution requires the widespread deployment of vehicular communication and connected vehicles technologies to be fully viable. The alternative for an ego vehicle is to collect data by its perception systems [39, 325] and anticipate the behavior of nearby target vehicles. While a majority of previous studies have focused on behavior prediction of target vehicles in highly structured environments such as highways or signalized intersections that have multiple lanes [11], less attention is devoted to less structured yet highly dynamic areas such as unsignalized intersections.

This section addresses vehicle behavior prediction with uncertainty based on perception for the target vehicles at unsignalized intersections. Taking advantage of multimodal output, we model the inherited uncertainties of different possible behaviors as shown in the provided sample in Fig. 6.14. This is crucial for automated vehicles to make safe and informed decisions. In fact, generating one solution may result in an invalid behavior since the model tries to find the average of various possible behaviors, which may not be a realistic/possible behavior for the target vehicle at the roundabout. For instance, as demonstrated in Fig. 6.14, the proposed model has generated two modes (shown in blue) of going straight and turning right for the target vehicle predicted at 5 meters after it enters the intersection. The probabilities are shown as a heatmap indicating which behavior the vehicle is most likely to follow. Noticeably, the average of different obtained

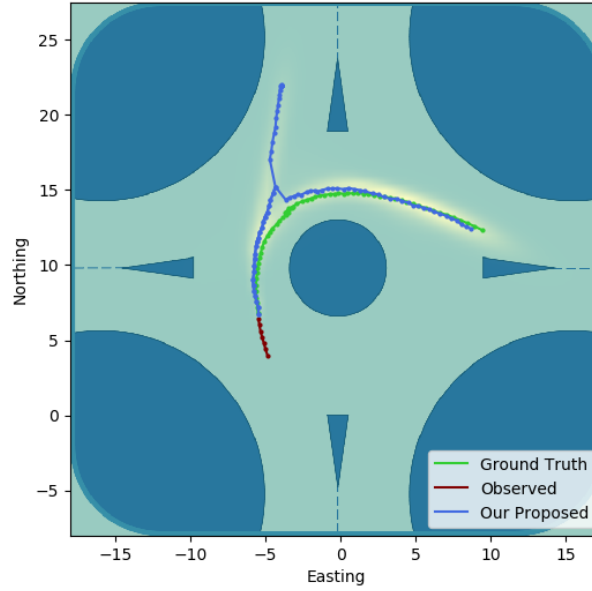


Figure 6.14: Multimodal output of the vehicle behavior prediction with the horizon of 4.8 seconds. Different modes and probabilities are shown in blue and heatmaps, respectively.

behaviors is not a feasible one for the vehicle under study. Our model works based on dilated convolutions networks to consider the temporal aspects of driving data in combination with a mixture density network (MDN). We conduct a quantitative comparison with several baselines and benchmarks. We also perform a qualitative analysis for the proposed behavior prediction approach.

6.3.1 Problem Statement

In this part, we first formulate the vehicle behavior prediction problem and then provide insights about the used dataset.

Problem Formulation

We consider the vehicle behavior prediction problem as a sequence modeling problem. The primary aim is to anticipate the potential behaviors of a target vehicle based on its observed past trajectory before time t . The input is the vehicle trajectories $Tr = [tr_1, tr_2, \dots, tr_N]$ from the dataset where N is the number of all vehicle behaviors, which is 23854 in our case. Each trajectory is made up of different information F about the

target vehicles for many time steps. Consider L as the length of the trajectories. Each trajectory can be denoted as follows:

$$tr_i = [\vec{F}_1, \vec{F}_2, \dots, \vec{F}_k] = \begin{pmatrix} f_1[0] & f_2[0] & \dots & f_k[0] \\ f_1[1] & f_2[1] & \dots & f_k[1] \\ \vdots & \vdots & \ddots & \vdots \\ f_1[L] & f_2[L] & \dots & f_k[L] \end{pmatrix} \quad (6.20)$$

We will use the information about lateral and longitudinal positions ($p = (x, y)$) as well as the target vehicle speed (v) and heading (h), leading to $k = 4$. Consequently, the vehicles trajectories in our case will be equal to:

$$tr_i = \begin{pmatrix} x[0] & y[0] & v[0] & h[0] \\ x[1] & y[1] & v[1] & h[1] \\ \vdots & \vdots & \vdots & \vdots \\ x[L] & y[L] & v[L] & h[L] \end{pmatrix} \quad (6.21)$$

To predict the future behavior of the target vehicles, we need to divide vehicle trajectories into all possible intervals of observation and prediction windows. We denote observation period length as obs and prediction horizon as prd . The primary goal in this prediction problem is to find the output distribution over the anticipated vehicles positions $p_i = (x_i, y_i)$ given a short observed interval (I):

$$I = [tr_i^{t-obs+1}, tr_i^{t-obs+2}, tr_i^{t-obs+3}, \dots, tr_i^t] \\ tr_i^t = [x_i^t, y_i^t, v_i^t, h_i^t], O = [p_i^{t+1}, p_i^{t+2}, p_i^{t+3}, \dots, p_i^{t+prd}]$$

Dataset

In this part, the used evaluation dataset is discussed in brief. As explained earlier, unsignalized intersections such as roundabouts contain widely variable motions, which are more complicated than maneuvers occurring in structured environments. We carry out analysis over a large public and naturalistic dataset, namely ACFR dataset [81]. ACFR contains more than 60 hours of data on vehicles passing five different unsignalized roundabouts located in Sydney, Australia. The common feature among all selected roundabouts is that they are all single-lane intersections with no existing traffic lights

Table 6.4: Number of employed maneuvers according to the relative destination of the vehicle motion.

Intersections	Maneuvers			
	Straight	Right	Left	Total
Roundabout 1	1356	1237	2577	5170
Roundabout 2	1825	217	1716	3758
Roundabout 3	5110	155	466	5731
Roundabout 4	3000	69	183	3252
Roundabout 5	5347	222	374	5943
Total	16638	1900	5316	23854

indicating who has the right of way. The data is collected using a vehicle equipped with LiDAR (range: 200 m and frequency:25 Hz) and tracking systems. ACFR consists of more than 23800 trajectories from different vehicles, including bikes, cars, and trucks. This dataset also includes information on the speed, heading, positions, and physical features such as the type of the vehicle or its size from the vehicles performing different types of maneuvers. Table 6.4 represents a summary of maneuvers used in this chapter grouped by different roundabouts and the final destination. There are also a limited number of u-turn maneuvers (a total of 58 in all roundabouts under study) in the initial dataset, which are not used in this study since the u-turn samples are not enough for train, validation, and test split. For all other maneuvers, the data from four first roundabouts are selected for training and validation. Since roundabout number 5 has the highest number of samples (i.e., 5943) among all roundabouts, we use its samples to test the proposed approach.

Furthermore, a sample of about 1000 vehicle trajectories for a single roundabout (first roundabout in particular) is demonstrated in Fig. 6.15. For each labeled maneuver, the samples are taken from different origin-destination pairs. As an example, considering straight maneuvers shown at Fig. 6.15a, upper trajectories are vehicles entering the roundabout from the west and exiting it from the east. On the contrary, those shown on the bottom of the intersection center are from vehicles entering from the east. Note that the dataset is collected in Sydney, where vehicles move clockwise after entering the roundabout. Furthermore, Fig. 6.16 indicates the average speed and heading for different

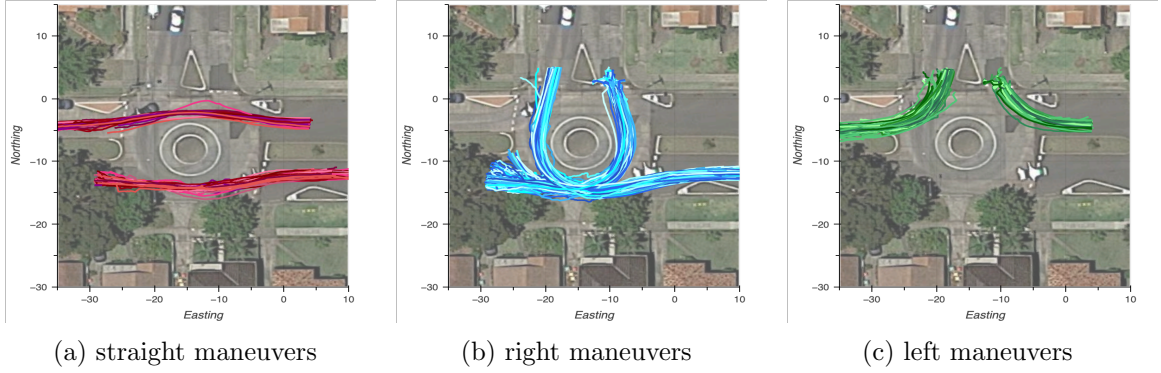


Figure 6.15: Different maneuvers occurred at one of the roundabouts in the dataset. The samples are taken from different origin-destination pairs for each maneuver. The used dataset is collected in Sydney, where the vehicles move clockwise.

maneuvers across all used trajectories. Bold lines indicate the mean value, and dashed lines represent standard deviations. Both average speed and headings are plotted as a function of distance relative to the entrance of the roundabout. Negative values mean the vehicle is approaching the roundabout entrance. As illustrated in Fig. 6.16, different maneuvers demonstrate unique profiles w.r.t average speed and heading. This reflects the importance of evaluating a vehicle behavior prediction model over different types of maneuvers to better analyze its capabilities.

6.3.2 Methodology

This part explains the data processing steps and the principal methodologies of the proposed model. We also discuss the introduced encoder-decoder architecture in detail.

Data Processing

In the processing step, we shift the vehicle trajectories into the same reference frame so that prediction models can work properly. To do so, we use a reference frame as shown in Fig. 6.14. Vehicle trajectories are normalized in a way that all vehicles move upward to the reference roundabout. The coordinate origin is selected at the center of the upward road where the vehicles enter the intersection or, in other words, where the vehicles start to perform their intended maneuvers. Negative values indicate that the target vehicles are still approaching the roundabout. For instance, the trajectory shown in Fig. 6.14

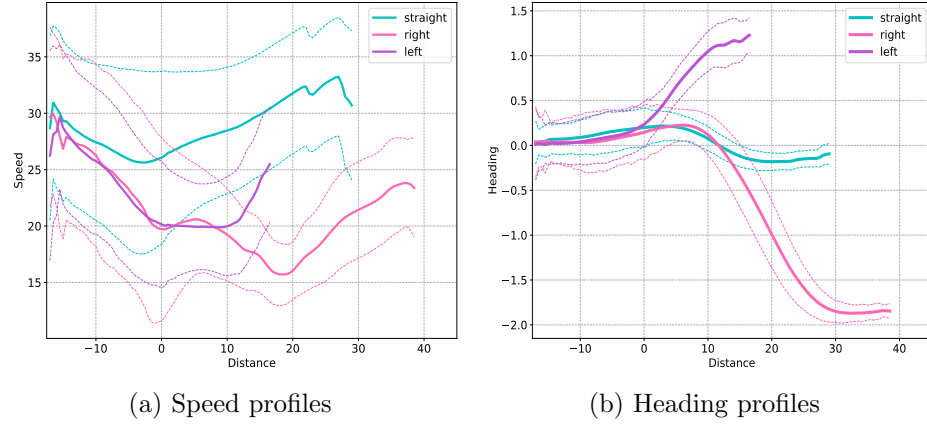


Figure 6.16: Average speed and heading over all used trajectories.

represents a vehicle performing the right maneuver, and the observation period starts from about 5 meters after the vehicle enters the roundabout.

Second, vehicle trajectories include data from when the target vehicle is at a far distance from the roundabout, traversing the intersection, and after the vehicle completes its maneuver fully. This point should be taken into account when we tend to extract the observation and prediction intervals. Suppose the selected observation intervals are far away from the intersection. In that case, all vehicles are going toward the intersection, and the drivers may not have made the decision to where to go next, making the prediction almost impossible. On the other side of the continuum, if the selected interval is after the target vehicle leaves the roundabout, their intended direction is already given, making the prediction process extremely trivial. To balance this, we select observation intervals from 5 meters before the roundabout entrance to the entrance and predict the vehicle behavior while they it is going around the roundabout during models test time.

Finally, as our primary goal is to predict the vehicle behavior at unsignalized intersections such as a roundabout, the amount of time the vehicles are traversing the intersection can vary based on the type of maneuver they are executing. For instance, vehicles turning right (clockwise driving) spend much more time at the roundabout than those turning left. Vehicle behavior prediction methodologies should take account of this point for the training process to avoid the possibility of the model generating the most probable result based on the time step. For instance, the model may predict the vehicle is moving to the left when the prediction time step is shorter than a value v and moving straight at the time step larger than v for all trajectories. This is clearly not a desirable

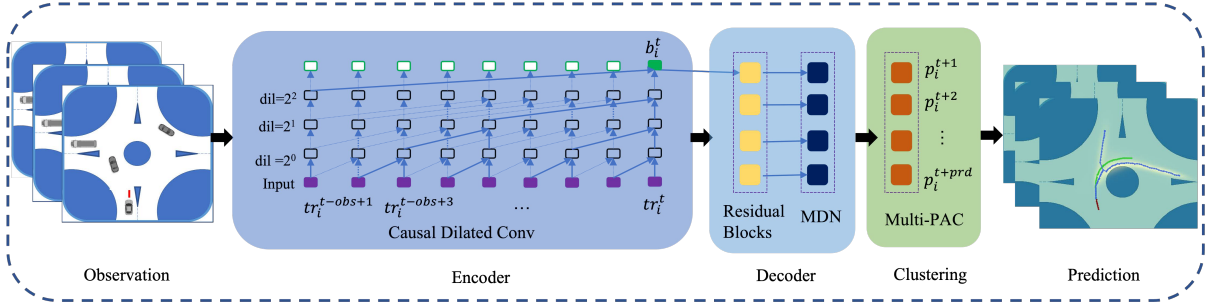


Figure 6.17: High-level architecture of the proposed vehicle behavior prediction model.

outcome. To address this issue, we apply the padding technique for each trajectory (sequence). Shorter tracks are padded to the output length. In other words, each track is at least as long as the sum of observation and prediction horizons.

Prediction Model

As explained earlier in problem formulation, we consider the vehicle behavior prediction task as a sequence modeling problem. Our primary aim is to infer a probability distribution $P(p^{t+1:t+prd}|tr^{1:t})$ of the future trajectories $\{p_i^{t+1}, p_i^{t+2}, p_i^{t+3}, \dots, p_i^{t+prd}\}$, given a set of previous observation $\{tr_i^{t-obs+1}, tr_i^{t-obs+2}, tr_i^{t-obs+3}, \dots, tr_i^t\}$. Clearly, the vehicle motion cannot be predicted based on future information of the target vehicle (i.e., after time t). To this aim, we propose to train a CNN-based encoder-decoder to model target vehicles' sequential behavior. A high-level structure of the introduced model is presented in Fig. 6.17. The encoder will analyze the observation information from time 1 to t , and its responsibility is to model the stochastic and temporal process of the previous behavior of the vehicle at the observation time and produce the intermediate output denoted as b_i^t , which are considered the latent representations. Then, the decoder module will generate the vehicle's future behavior at the prediction time. The decoder responsibility is to analyze the latent intermediate outputs generated by the encoder and produce the probabilistic predictions of the future vehicle trajectories. These probability distributions will then be grouped into modes. We provide a detailed explanation of each component in the following.

Encoder Module

Figure 6.18 illustrates a more detailed architecture of the encoder and decoder modules of the network. As indicated in Fig. 6.18a, the core component of the introduced encoder architecture is causal convolutions. By taking advantage of causal convolutions, we ensure that the prediction model cannot ignore the ordering in which we intend to model the trajectories' time steps. In other words, the prediction $\{p^{t+1}|tr^1, \dots, tr^t\}$ generated by the casual convolution at t should not be based on the subsequent time steps $tr^{t+1}, tr^{t+2}, \dots, tr^{t+prd}$. Rather, it can only be gained from the time steps that are no later than t . This feature of the causal convolution matches the primary characteristic of the problem we are addressing (i.e., only previous observations should be used to predict the vehicle behavior).

Nevertheless, one drawback of the causal convolutions is their limited receptive field that is the set of points from the original inputs of the network that can affect a specific point in the output. This means that to analyze longer temporal dependencies, we require either increasing the depth of the network (larger numbers of layers) or a larger filter size, leading to higher computational complexity. To overcome this problem, we benefit from dilated causal convolutions inspired by [326]. These dilated convolutions work based on the dilation concept, which allows the filter of the CNN to be applied over a wider area, increasing the receptive size. We can define dilation dil as the number of steps the filter skips the input elements to produce the output, which can be exponentially increased as illustrated in Fig. 6.17. We can calculate the feature map f as follows:

$$f(t) = (I *_{dil} M)(t) \sum_{s=0}^{S-1} M(s)I(t - dil.s) \quad (6.22)$$

where I is the input, M represents the filter, and S is the filter size. When stacked, dilated convolutions allow the encoder to expand its receptive size and capture long temporal dependencies better. An illustrative example is demonstrated on the left part of Fig. 6.17, where the dilation factor increases exponentially (i.e., 1, 2, 4, and 8 for the output layer of the encoder). Consider $dil = 2^1$, as demonstrated, the filter at the second hidden layer will skip two input elements (from the previous layer).

The introduced encoder module consists of two distinct dilated convolutions at each layer, as shown in Fig. 6.18a. The filter size S and dilation factor dil are the same for the two convolutions of the same encoder layer. As illustrated, we employ skip-connections (i.e., the input will skip one or more layers and is concatenated with the output) in each block as they are shown [289] to help network stability and training

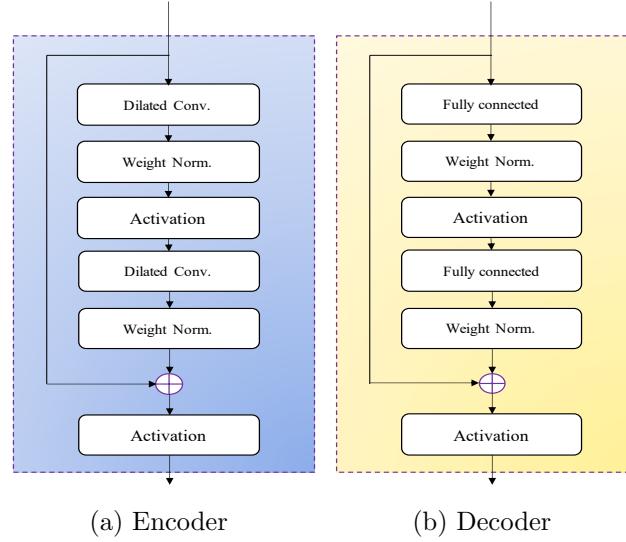


Figure 6.18: Structure of the residual blocks.

process, especially if the time steps are of a high number. There are two stacked dilated convolution layers in each residual block, followed by weight normalization and ReLU activation. Skip-connections take place after the second weight normalization layer. The optimized hyper-parameters used for the encoder are further discussed in Section 6.3.3.

Decoder Module

As mentioned earlier, the responsibility of the decoder module is to process the intermediate network outputs generated by the encoder and generate the output distribution for the future vehicle trajectories. The introduced decoder consists of two primary components: (1) the residual blocks and (2) the MDN dense layer. The latter is a mixture density layer where the network predicts the probability estimates of a vehicle's possible future behaviors, which we will explain in the subsequent section. Figure 6.18b demonstrates the detailed architecture of the first component that is residual blocks of the proposed decoder. Similar to the encoder module, skip-connections are used to make the training process go smoothly. There exist two fully connected layers, followed by weight normalization and the ReLU activation function. The input of the residual block is the latent representation b_i^t generated using the encoder at time t . The residual blocks are designed to capture the information gained from the historical sequential data and posterior distribution. The latent outputs of the residual blocks are then fed to an MDN where we address the multimodality characteristic of the behavior prediction.

MDN Module

As mentioned earlier, unimodal models output a single solution that appears to be the most probable one. However, future behaviors of the target vehicles follow an inherent multimodality characteristic in which different solutions with different probabilities are possible and their average is not necessarily a valid behavior for the vehicle. To model this inherent behavior, we employ a mixture density layer as an output layer in our model. This will allow the encoder-decoder network to produce multiple solutions and measure their degree of uncertainty. We can calculate the probability density as follows:

$$P(p^t|I) = P(p^t|p^{t-1}) = \sum_{n=1}^l \pi_n^{t-1} \mathcal{N}(p^t|\mu_n^{t-1}, \sigma_n^{t-1}, \rho_n^{t-1})$$

where,

$$\mathcal{N} = (p|\mu, \sigma, \rho) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-Z/2(1-\rho^2)\right)$$

$$Z = \frac{(x_g - \mu_1)^2}{\sigma_1^2} + \frac{(y_g - \mu_2)^2}{\sigma_2^2} - \frac{2\rho(x_g - \mu_1)(y_g - \mu_2)}{\sigma_1\sigma_2}$$

where (x_g, y_g) is the actual future positions, σ and μ are the standard deviation and the mean (i.e., a 2-D vector), and π and ρ represent density weights and correlation coefficient of Gaussian distribution, respectively, which are generated and processed from MDN parameters $(\hat{\mu}, \hat{\sigma}, \hat{\rho}, \hat{\pi})$ based on the following equations.

$$\begin{aligned} \mu_n^t &= \hat{\mu}_n^t, \{\mu_n^t \in \mathbb{R}\} \\ \sigma_n^t &= \exp(\hat{\sigma}_n^t), \{\sigma_n^t > 0\} \\ \rho_n^t &= \tanh(\hat{\rho}_n^t), \{\rho_n^t \in (-1, 1)\} \\ \pi_n^t &= \frac{\exp(\hat{\pi}_n^t)}{\sum_{n'=1}^l \exp(\hat{\pi}_{n'}^t)}, \{\pi_n^t \in (0, 1) : \sum_{n=1}^l \pi_n^t = 1\} \end{aligned}$$

Moreover, the loss function for the MDN is calculated as follows.

$$\mathcal{L}(x) = -\log \sum_{n=1}^l \pi_n \mathcal{N}(p|\mu_n, \sigma_n, \rho_n) \quad (6.23)$$

which is the probability of (x_g, y_g) being within the output distribution. In other words, while training the network, this loss function guarantees that the model is maximizing the target feature output probability.

The Output

In the previous part, we model the multimodality nature of the vehicle's possible future behaviors and extract the probability distribution, which is the heatmaps shown in Fig. 6.14. These represent a set of 2-D Gaussians, each of which has a corresponding weight and time step. These multimodal probability distributions should be grouped into modes in order to demonstrate a comprehensive and meaningful solution. To this aim, we use the Multi-PAC clustering algorithm [39]; however, any other clustering algorithms could also be adapted instead as long as the same algorithm is also applied to the output distribution of all other multimodal comparison models. This will guarantee a fair comparison between the output distribution of different models. We discuss the processing procedures of Multi-PAC in the following.

The primary responsibility of Multi-PAC [39] is to group output distributions into possible behaviors. First, a filter is applied to the outputs to remove the weak Gaussians and reduce noise. Then, the remaining Gaussians are grouped together at each time step. There are now some groups/clusters at each time step, which can be ordered in time by making a tree. We can assign a parent to each tree node according to its nearest node (according to the Euclidean distance) from the previous time step. A new mode is generated if there is more than one child for a parent node. This way, we can obtain all the possible modes or vehicle behaviors by going from parents to the leaves of the tree. During the evaluation time and to compare the performance of multimodal approaches, we use the most probable solutions, i.e., the behaviors with the highest probability for all comparison models.

6.3.3 Driver Behavior Prediction Experimental Evaluation

This section presents how we evaluate the proposed vehicle behavior prediction approach and compare its performance with several baselines and benchmarks. We first explain the details about the evaluation process, followed by evaluation metrics and comparison behavior prediction models. We then report the quantitative evaluation and qualitative analysis results.

All models are compared over the same dataset explained in Section 6.3.1. To avoid model biasing toward more common moves, we use the oversampling technique only at the training time. This will guarantee the proper operation of the models when coming across imbalanced data. Furthermore, the models are implemented using Python and the TensorFlow library. Hyper-parameters of all deep learning-based models are optimized

Table 6.5: Optimized hyper-parameters settings

Parameters	Parameters settings
Learning rate	0.002
Decay rate	0.96
Dilation	{1,2,4,8}
Batch size	100
Mixtures	6
Threshold	0.5
Epochs	250

based on a grid search technique at the time of training and validation and according to the cross-validation splits. Then, we employ the obtained optimized hyper-parameters during the test. The optimized hyper-parameters for the proposed approach are presented in Table 6.5. Observation length and prediction horizons are 0.56 and 4.8 seconds, respectively, following the standard evaluation [210, 327].

Furthermore, we use the data from the first four roundabouts, shown in Table 6.4, to create the train and validation sets. The split ratio is 4:1 for the cross-validation. The number of trajectories is high enough to have a considerable number of samples for the validation sets (more than 14000 trajectories for training and 3000 for validation at each fold). Since the last roundabout has the highest number of samples (i.e., 5943) among all roundabouts, we use its samples to assess the performance of the introduced behavior prediction approach as well as comparison models. Note that the trajectories from this roundabout are not employed at the time of training or validation. Instead, they are used during test time only. In this way, we can evaluate the generalization ability of prediction models over the trajectories from new roundabouts.

Evaluation Metrics

We now explain the metrics utilized to evaluate and compare the performance of behavior prediction methodologies. In our experiments, we employ the following three distance metrics based on which we compare the behavior prediction results.

- **Euclidean** distance (Euclid) between the ground truth (i.e., vehicle chosen behavior) and the predicted vehicle behavior.

- **Modified Hausdorff Distance** (MHD) [328] which is calculated based on the nearest point on one trajectory to all the points on the second trajectory.
- **Horizon** distance (HRZN) which is calculated as the Euclid distance between various points that are recorded at a particular time in seconds.

We report the Euclid results in three formats, including the RMSE, the worst 5%, and 1%. The RMSE for the predicted results and ground truth is calculated as follows:

$$RMSE = \sqrt{\frac{1}{prd} \sum_{i=t+1}^{t+prd} (x_g^i - x_p^i)^2 + (y_g^i - y_p^i)^2} \quad (6.24)$$

where prd represents the size of the prediction window or the number of time steps, (x_g, y_g) and (x_p, y_p) refer to the ground truth and predicted tracks, respectively. The mean, worst 5%, and 1% are also reported for the other two metrics.

Comparison Models

We compare the vehicle behavior prediction results of our introduced model with several baselines and benchmark models. The following is the list of comparison models along with a brief description of each one.

- *D-LSTM* [314]: A deep LSTM model is trained, which consists of one layer of LSTM followed by six layers of fully connected layers. The tuned hyper-parameters for this model are: (1) optimizer: Adam, (2) batch size: 100, and (3) learning rate: 0.0001.
- *ML-LSTM* [39]: Three layers of stacked hyper LSTM recurrent units combined with a mixture density network to obtain probabilistic results are trained in this model. The tuned hyper-parameters for this model consists of: (1) optimizer: Adam, (2) batch size: 100, (3) learning rate: 0.0005, (4) threshold: 0.5, and (5): mixture numbers: 6, which are all tuned by a grid search.
- *LSTM-FL*: A basic encoder-decoder based on multiple layers (three layers) of LSTM and first loss training approach is trained. The tuned hyper-parameters include (1) optimizer: Adam, (2) batch size: 100, and (3) learning rate: 0.0001.
- *Gaussian process*: We train a Gaussian Process Regression (GPR) model over the used dataset.

- *Three baselines:* Three different kinematic baseline models are used for the evaluation comparison. We employ constant velocity (CV), constant turn rate and velocity (CTRV), and constant turn rate and acceleration (CTRA) as the baselines.

Quantitative Evaluation Results

We assess the performance of our vehicle behavior prediction model and compare it with that of several other benchmark and baseline approaches mentioned in Section 6.3.3 and present the results based on three different evaluation metrics. Tables 6.6, 6.7, and 6.8 demonstrate the behavior prediction results when the target vehicle is performing one of the straight, right, and left maneuvers while traversing the intersection, respectively. The obtained results indicate the deviation of the predicted positions from the ground truth for all comparison models based on three different distance metrics. The worst scores of each metric are also recorded to emphasize better how all prediction methodologies may fail the task and provide a more detailed analysis of the performance of comparison methods. Examining different evaluation metrics indicates that the proposed model outperforms all baselines and benchmarks. For instance, as shown in Table 6.9, our model achieves 1.064 ± 0.0247 , 3.548 ± 0.0366 , and 0.531 ± 0.0153 for Euclid, HRZN, and MHD mean, respectively, with a confidence level of 95%. This consolidates the capability of dilated convolutions over recurrent units such as LSTM. Moreover, the performance order of the models is almost the same for all maneuvers: our proposed approach, followed by ML-LSTM, D-LSTM, and GPR. The poor performance of the LSTM-FL model can be traced back to the fact that only one time step is used for the LSTM encoder. Furthermore, considering the computational complexity, the inference time of our proposed scheme (0.04 seconds) is less than that of D-LSTM (1.46 seconds) and ML-LSTM (0.28 seconds). Unlike these two approaches, we employ dilated convolutions resulting in a significant speed increase.

Comparing the results obtained for all models when vehicles are carrying out different maneuvers, we can observe that predicting the behavior of vehicles performing the right maneuver is more challenging, especially for baseline models. The reason is that it is harder to distinguish the behaviors of the vehicles moving straight and turning right than those of the vehicles going straight and turning left at the roundabout, which is because the vehicles turning left spend less time traversing the roundabout. Note that the dataset is collected in Australia, where vehicles move clockwise after entering the roundabout. For instance, considering CV baseline and MHD metric results, the

error increases sharply from 2.38 ± 0.0191 , 3.61, and 4.79 for left-turn trajectories to 7.18 ± 0.0585 , 10.67, and 11.36 for right-turn vehicle tracks (with a confidence level of 95%). Deep learning-based models demonstrate better performance than baseline approaches. Considering our model and Euclid metric results, the behavior prediction error increases from 1.71 ± 0.0220 , 3.23, and 3.89 for vehicles performing left maneuver to 1.94 ± 0.0275 , 4.03, and 4.42 for vehicles carrying out right maneuver. Our multimodal behavior prediction model performs better than others even in the challenging cases of right-turn maneuvers, achieving an average improvement of about 15.3%, 4.2%, and 12.2% for Euclid, HRZN, and MHD metrics, respectively, compared to the best previous benchmark.

The behavior prediction results when the target vehicle is executing a left maneuver, as shown in Table 6.6, demonstrate the better performance of the GPR model in a few cases (e.g., 1.505 ± 0.0190 RMSE), especially w.r.t. the Euclid metric. However, it should be reminded that unlike our proposed and other compared deep learning approaches, GPR outputs a unimodal solution that can be the average of two or three trajectories. The provided solution will not always be a feasible behavior based on the direction. Nevertheless, GPR will receive a low distance for this solution since it is closer to the actual vehicle behavior (i.e., ground truth). Our behavior prediction model outperforms other baselines and comparison benchmarks. Also, considering the HRZN worst scores, our approach performs better than other multimodal approaches. It also achieves very close results to those of GPR, representing that a proper behavior exists in our set of predictions.

Furthermore, Table 6.9 demonstrates model performance for all maneuvers existing in the data trajectories. Our prediction model performs better than all other baselines and benchmarks, demonstrating its capability in handling maneuver imbalance when all trajectories are considered. It indicates an average improvement of about 14.5%, 10.8%, and 19.3% w.r.t. Euclid, HRZN, and MHD metrics, respectively, in comparison with the best previous benchmark, namely ML-LSTM (1.291 ± 0.031 , 4.002 ± 0.0877 , and 0.726 ± 0.0242). In addition, comparing the results for different types of LSTM-based models, we can observe that LSTM-FL exhibits a poor performance, matching results to those of CTRV and CTRA. This highlights the failure of first loss training in recurrent-based models (i.e., loss over the first time step) for the behavior prediction task since fewer temporal patterns are used. Another interesting point is the lower performance of D-LSTM compared to ML-LSTM, which demonstrates that higher depths of the network will not necessarily yield improvements.

Table 6.6: Comparison results of all models over left maneuvers.

Metric	Models						
	CTRA	CTRV	CV	Gaussian Process	D-LSTM	ML-LSTM	LSTM-FL
Euclid RMSE	1.848	1.792	2.492	1.505	1.894	1.764	3.764
Euclid worst 5%	4.540	3.941	3.862	3.052	3.478	3.402	4.435
Euclid worst 1%	6.120	5.609	4.834	4.073	4.656	3.976	4.968
HRZN mean	2.253	2.184	3.053	1.847	1.754	1.635	4.795
HRZN worst 5%	4.004	3.724	4.487	4.098	5.650	5.296	7.140
HRZN worst 1%	9.021	7.688	5.523	5.252	6.301	6.127	7.841
MHD mean	1.710	1.672	2.385	1.439	1.118	1.112	3.670
MHD worst 5%	3.652	3.730	3.612	2.861	3.734	2.465	4.253
MHD worst 1%	5.986	5.279	4.793	3.818	3.988	3.278	4.728

Table 6.7: Evaluation results for all compared models over straight maneuvers.

Metric	Models							
	CTRA	CTRV	CV	Gaussian Process	D-LSTM	ML-LSTM	LSTM-FL	Our proposed
Euclid RMSE	2.752	2.729	2.021	1.290	2.374	1.291	7.594	1.273
Euclid worst 5%	6.989	6.104	3.756	3.410	3.304	3.153	8.788	3.022
Euclid worst 1%	11.080	10.883	5.629	5.440	6.755	6.495	9.272	4.860
HRZN mean	2.060	2.082	1.498	1.031	1.189	1.172	6.342	1.301
HRZN worst 5%	4.936	4.812	2.815	2.914	2.880	2.793	7.213	2.405
HRZN worst 1%	11.812	11.559	4.275	4.203	5.902	4.934	10.556	3.901
MHD mean	2.528	2.485	1.785	0.945	0.953	0.692	6.854	0.675
MHD worst 5%	5.930	5.504	2.882	2.623	2.334	1.903	8.096	1.672
MHD worst 1%	10.235	10.210	4.509	4.818	3.435	2.556	8.531	2.562

Table 6.8: Comparison results of all models over right maneuver trajectories.

Metric	Models						
	CTRA	CTRV	CV	Gaussian Process	D-LSTM	ML-LSTM	LSTM-FL
Euclid RMSE	9.046	9.842	8.463	5.010	3.854	2.210	7.183
Euclid worst 5%	14.873	15.843	12.616	7.238	4.872	4.938	8.034
Euclid worst 1%	20.507	17.921	13.356	8.354	5.599	5.232	9.132
HRZN mean	9.764	10.591	8.796	5.541	2.450	2.124	7.756
HRZN worst 5%	16.542	18.874	13.859	8.391	6.014	5.450	10.923
HRZN worst 1%	21.830	21.912	16.067	9.965	6.623	6.894	11.238
MHD mean	7.338	7.999	7.178	4.519	1.437	1.492	6.632
MHD worst 5%	11.816	11.576	10.671	6.516	3.903	2.768	7.542
MHD worst 1%	18.273	13.550	11.362	7.209	4.002	3.523	7.972

Table 6.9: Evaluation results for all models over all existing maneuvers in the trajectories.

Metric	Models							
	CTRA	CTRV	CV	Gaussian Process	D-LSTM	ML-LSTM	LSTM-FL	Our proposed
Euclid RMSE	2.939	2.947	2.303	1.536	1.780	1.291	7.243	1.064
Euclid worst 5%	9.132	9.451	5.014	4.337	3.567	2.807	8.761	2.806
Euclid worst 1%	11.581	11.994	10.440	6.346	5.928	5.341	9.260	4.732
HRZN mean	8.031	7.978	6.343	4.932	3.756	4.002	13.442	3.548
HRZN worst 5%	17.176	15.789	11.231	9.104	8.239	7.784	17.289	7.193
HRZN worst 1%	21.638	20.588	14.245	12.377	11.791	11.937	18.111	10.303
MHD mean	2.663	2.649	2.035	1.117	0.934	0.726	6.63	0.531
MHD worst 5%	7.281	8.245	4.008	3.659	1.982	1.760	8.056	1.406
MHD worst 1%	10.360	10.437	9.066	5.611	4.098	2.747	8.567	2.450

Qualitative Analysis Results

We also qualitatively analyze the results of our behavior prediction model to better highlight the multimodality of the potential solutions. We demonstrate an illustration of the predicted behaviors for twelve sample vehicles while executing various types of maneuvers in Fig. 6.19. The first row indicates predicted trajectories for the target vehicles moving straight. As can be seen, the model considers a slight possibility for the target vehicle to travel left at the early stage of the prediction (Fig. 6.19a1 and 6.19a3), as well as a slight possibility of right-turn near the center of the roundabout (Fig. 6.19a2 and 6.19a4), as this may be true in a real-world situation. Nevertheless, the most probable solution is moving straight, which is close to the ground truth, as indicated by the heatmap. Unimodal baselines predictions are also shown to emphasize that the average of several solutions may not be a feasible behavior for the target vehicle. The second and third rows represent the results while target vehicles are executing right and left turns, respectively. The same multimodal behavior can be observed for these maneuvers as well. As shown in Fig. 6.19b3 or 6.19b4, the model manages to predict the behavior even for right-turns effectively. At the entrance of the roundabout, although a left turn may be possible, the vehicles move faster to be considered for this direction, which makes the model put more weight on the right maneuver.

Figure 6.20 illustrates a sample of predicted behaviors for two particular vehicles moving toward the right of the roundabout. A complete sequence for each vehicle and at different distances from the roundabout entrance is shown in each row to demonstrate how the model output converges from different possibilities to the best one. At the early stage, when the target vehicle is still moving upward to the reference frame and at a farther distance from the roundabout (Fig. 6.20a1 and 6.20b1), no particular maneuver or direction is taken by the driver. Since all directions are feasible in these scenarios, the model has produced a multimodal output with three different possibilities. Note that the right turn is still within the behavior possibility set, but its probability is very low. As the vehicle approaches the intersection, the speed and heading profile of the vehicles appear to diverge (see also Fig. 6.16). Therefore, the model performance improves at the distance of 0 meter, which is when the vehicle enters the roundabout. Figure 6.20a2 represents a more challenging scenario where the target vehicle probably stops or reduces speed while entering the roundabout. The model predicts their behaviors with more confidence as the target vehicles pass through the roundabout.

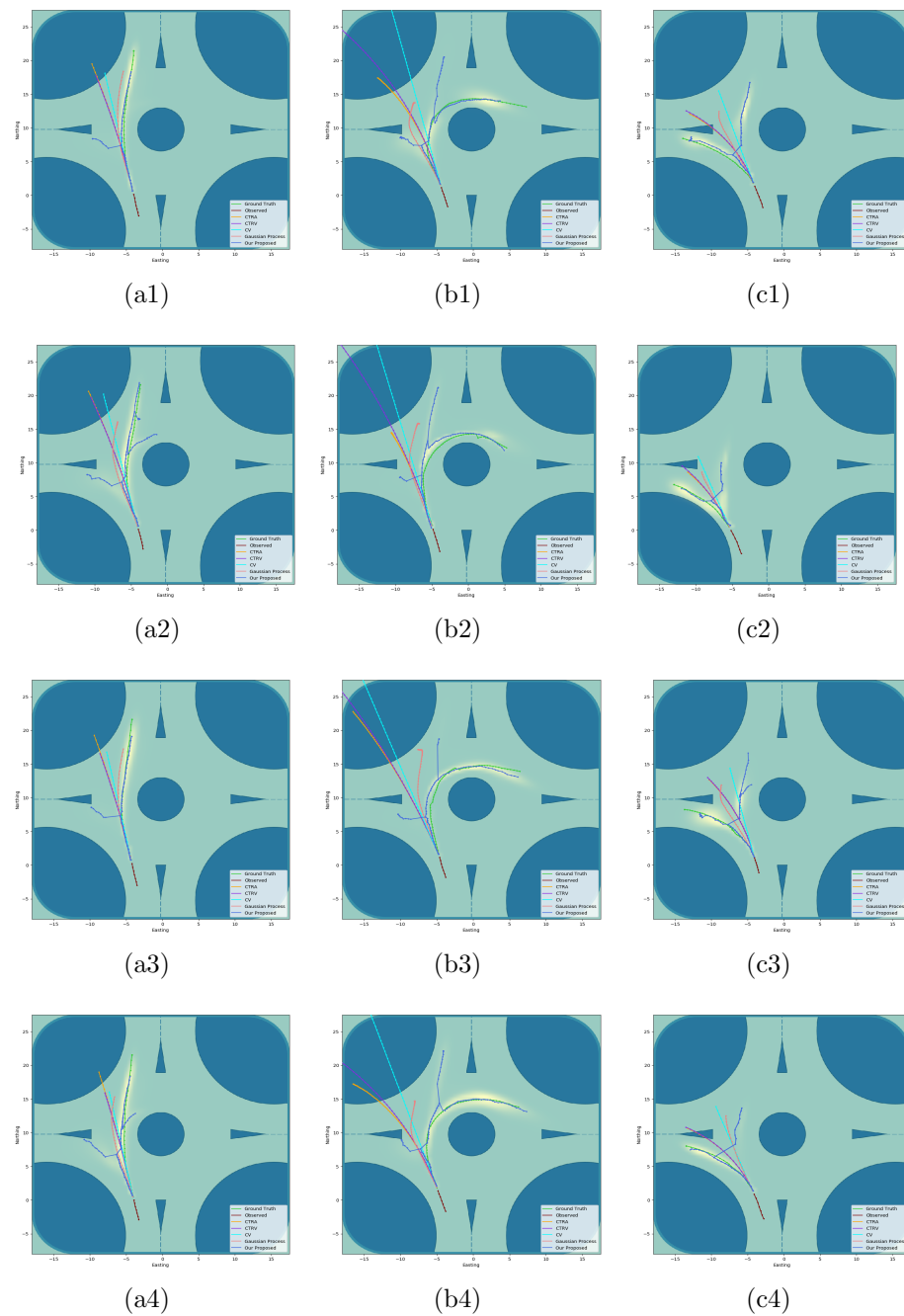


Figure 6.19: Qualitative analysis of behavior prediction results for our proposed model. Trajectories are shown from twelve different vehicles when carrying out various maneuvers: straight (a1-a4), right (b1-b4), and left (c1-c4). Different mode and probability outputs are shown in blue and heatmaps, respectively. Unimodal baseline results are demonstrated for comparison.

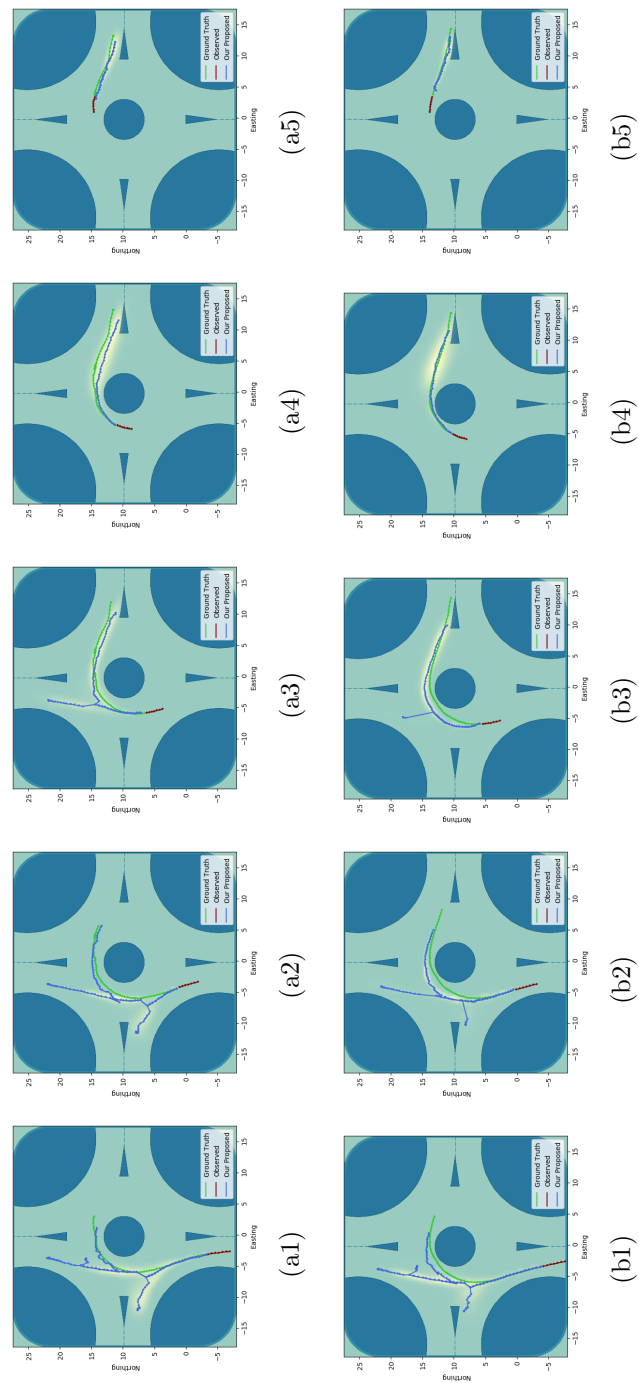


Figure 6.20: Predicted trajectories for two sample vehicles while carrying out a right-turn maneuver. Each row represents a complete sequence of predictions at various distances (i.e., -5, 0, 5, 10, and 20 meters) from the entrance of the roundabout. Negative distance indicates the target vehicle has not entered the roundabout yet.

6.4 Concluding Remarks

This chapter has embarked on a comprehensive exploration of driver intention and vehicle behavior prediction in the context of complex driving scenarios like unsignalized intersections. Overall, we can summarize the following conclusion points according to the obtained results in this chapter:

- We addressed the task of anticipating drivers intentions. We propose to analyze a short sequence of trajectory data to predict the possible maneuver of the vehicle at the T-junction using a deep learning-based approach. Cross-validation experimental results for driver intention at a T-junction in Sydney indicate the high performance of the introduced methodology. Our model achieved an accuracy of 100% at an average distance of 7 meters after the reference distance from the center of the T-junction. Based on the obtained driver intention results, it can be concluded that the velocity profile of the vehicles moving towards the T-junction has a substantial impact on how early the models can reach 100% prediction.
- We proposed an interaction-aware framework to analyze the past behaviors of the TV and its SVs as well as the interdependencies between them using spatial-temporal GCNs. Evaluation results have demonstrated, first, how considering the interaction among the TV and its SVs could enhance the behavior prediction performance. Second, the proposed approach outperformed several state-of-the-art models.
- We addressed the multimodal behavior prediction problem at unsignalized intersections and presents a deep learning-based methodology to anticipate the potential behaviors of the target vehicles passing the intersection. Our introduced model can successfully predict possible future behaviors of the target vehicles and generate a multimodal solution. Quantitative evaluation results over a dataset containing more than 23800 trajectories have revealed that the proposed system outperforms several baselines and state-of-the-art models based on three different metrics. Second, predicting the behaviors for the vehicles executing right maneuvers in clockwise driving scenarios is more challenging since it is harder to distinguish the headings of the vehicles moving straight and turning right than those of the vehicles going straight and turning left. This trend is observed for all the comparison models.

Chapter 7

STAG: Spatio-Temporal Attention Graphs for Interaction-Aware Agent Behavior Prediction

As discussed in Chapter 2, more studies have considered and modeled social interactions between vehicles and their SAs in recent years. Focusing on the temporal interactions, a large number of studies adopt recurrent neural networks [329] that are powerful in modeling temporal interactions but lack spatial ones. Later methodologies [134, 233] adopt social pooling as a mechanism to allow the recurrent units of SAs to share their features with one another. These methodologies, however, fail to correctly model the complex spatio-temporal correlations among the traffic participants. To alleviate these issues, more recent techniques [127, 130, 235] take advantage of simultaneous modeling of spatial and temporal interactions among agents utilizing graph neural networks and distance-based weighting mechanisms and achieve state-of-the-art results in vehicle behavior prediction. However, these approaches consider social interactions between all neighboring agents symmetric and model them as a dense undirected graph where it is assumed that the strength of agents' influence on one another is the same, resulting in superfluous and fixed interactions as illustrated in Fig 7.1 (see symmetric social interactions).

In this chapter, we introduce Spatio-Temporal Attention Graphs (STAG) to address the aforementioned challenge for the vehicle behavior prediction, where we create directed spatio-temporal graphs to model the spatial and temporal social correlations between the TA and those of its SAs that can effectively impact its behavior. We adopt attention-

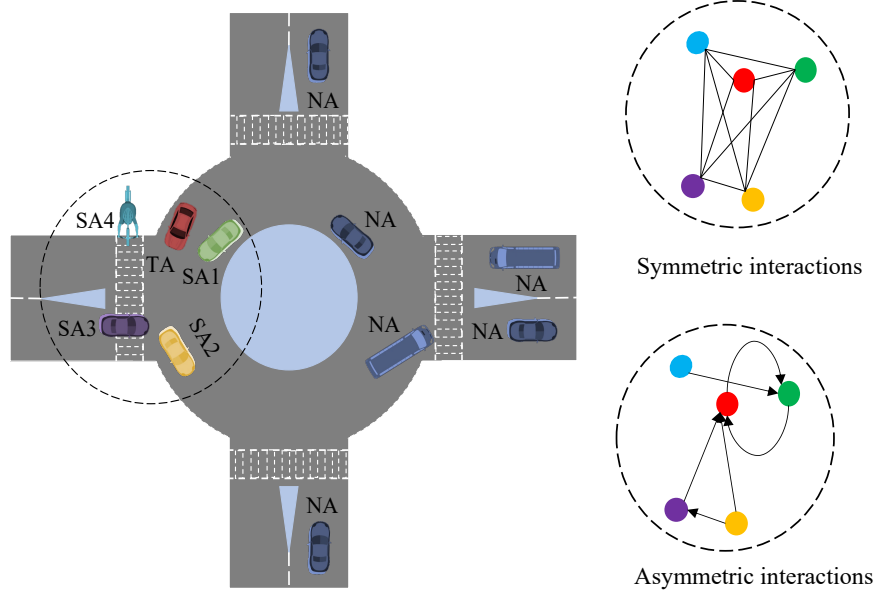


Figure 7.1: Graph-based interaction-aware behavior prediction while considering symmetric and asymmetric social interactions between the target (TA) and surrounding agents (SA)s.

based mask blocks to learn asymmetric interactions among multi-type agents¹. We then propose to extract the behavior representations using graph convolutional networks from the generated graphs. We generate the vehicles' future behavior using temporal convolutional networks. Therefore, the model takes spatial inter-agent correlations and motion tendency into account at the same time as well as managing the influence strength of interactions. For instance, consider the scenario shown in Fig. 7.1 where there are four heterogeneous surrounding agents (SA) for a target vehicle within the scene. SA3 represents aggressive behavior while entering the roundabout w.r.t. the target agent (TA), resulting in only the target vehicle needing to stop or detour its path. Also, the behaviors of the target vehicle and the bicycle do not have any effects on each other. Evidently, the construction of dense graphs based on all social interactions will lead to extra and not necessary interactions and cannot reflect the true interactions in this scenario. Directed graphs, on the other hand can represent these sorts of social interactions for the vehicle behavior prediction, as illustrated in Fig 7.1. We can summarize the main contributions

¹Throughout this chapter, the general term "agent" refers to various road participants, including drivers, cyclists, and pedestrians.

of this chapter as follows. We develop STAG, a novel encoder-decoder vehicle behavior prediction system with asymmetric social interaction reasoning which uses graph convolutions to extract the representations and generates the vehicles' future behaviors with TCN. Also, We model the social interactions between the TA and its heterogeneous SAs using a directed graph representation. As a result, the model not only considers spatial inter-agent correlations and motion tendency simultaneously but also avoids symmetric interactions, making it possible to take the strength of SAs' influence into account.

The rest of this chapter is organized as follows. Section 7.1 provides insights on the relevant deep learning-based preliminaries used in our proposed behavior prediction method. Section 7.2 formulates the behavior prediction problem. Section 7.3 introduces the primary interaction-aware behavior prediction methodology and the GNN-based method used to model inter-agent interactions. Section 7.4 discusses the experimental settings and summarizes evaluation results. Eventually, Section 7.5 provides final remarks, conclusion, and improvement directions.

7.1 Background

This section describes the preliminaries used in our proposed approach and summarizes a high-level description of GNNs and attention mechanism.

7.1.1 GNNs

Graph Neural Networks, as powerful deep learning tools, are proposed to capture the underlying characteristics as well as relational reasoning within the graph structured data [129]. GNNs have been investigated and proved to be efficient in several research domains such as traffic prediction [330], few-shot learning [331], or action recognition [332]. Here, a graph structure can be formulated as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ correspond to node and edge groups, respectively. Consider $v_i \in \mathcal{V}$ as a node, r_v as total number of nodes in $\mathcal{G}$, and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ as an edge; an adjacency matrix $\mathcal{A}$ can be defined with the size of $r_v \times r_v$. Also, the set of neighbors for node v_i are denoted as $N(v_i) = \{u \in \mathcal{V} | (u, v_i) \in \mathcal{E}\}$. A graph definition can also contain node and edge attributes denoted as $\mathcal{B}_v \in \mathbb{R}^{r_v \times d_v}$ and $\mathcal{B}_e \in \mathbb{R}^{r_e \times d_e}$, where d_v and d_e are dimensions, and r_e is the total number of edges. Let $\mathcal{G}$ be the input graph of a GNN. The output graph can be the result of different levels of updates operated on $\mathcal{G}$ including node, edge, or graph-level updates [129], which can be denoted as shown in Eq 7.1, Eq 7.2, and Eq 7.3:

- Edge-level:

$$b_e^{i'} = \mathcal{F}_e(b_e^i, v_i, v_j, b_u), \quad \bar{b}_e^{i'} = \mathcal{L}^{e \rightarrow v}(\mathcal{B}_e^{i'}) \quad (7.1)$$

- Node-level:

$$b_v^{i'} = \mathcal{F}_v(\bar{b}_e^{i'}, b_v^i, b_u), \quad \bar{b}_e^{i'} = \mathcal{L}^{e \rightarrow u}(\mathcal{B}_e^{i'}) \quad (7.2)$$

- Graph-level:

$$b_u^{i'} = \mathcal{F}_u(\bar{b}_e^{i'}, \bar{b}_v^{i'}, b_u), \quad \bar{b}_v^{i'} = \mathcal{L}^{v \rightarrow u}(\mathcal{B}_v^{i'}) \quad (7.3)$$

where $\mathcal{F}$, $\mathcal{L}$, and b_u are neural networks, aggregation functions, and graph attributes, respectively.

STAG will model the social interactions between the target vehicles and their SAs based on spatial and temporal graphs in which the node attributes will be updated over time. There will be no graph-level attributes.

7.1.2 Attention Mechanism

Transformer architectures are introduced to replace the complex recurrent architectures with much simpler and parallelizable networks and have produced state-of-the-art results in NLP tasks such as machine translation [333]. These networks work merely on the basis of attention mechanisms, obviating the need for recurrent units. The core concept of a transformer is self-attention, whose purpose is to learn a latent representation of data. Let $\{h^t\}_{t=1}^T$ be the embeddings, $\mathcal{Q} = \mathcal{F}_{\mathcal{Q}}(\{h^t\}_{t=1}^T)$ be the queries, $\mathcal{K} = \mathcal{F}_{\mathcal{K}}(\{h^t\}_{t=1}^T)$ be the keys, and $\mathcal{M} = \mathcal{F}_{\mathcal{M}}(\{h^t\}_{t=1}^T)$ be the values. The output scores can be calculated by Eq 7.4 [333]:

$$Att(\mathcal{Q}, \mathcal{K}, \mathcal{M}) = softmax\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{d_k}}\right)\mathcal{M} \quad (7.4)$$

where the numerator is scaled by using the dimension d_k of each key. Compared with RNNs, this approach has the capability of learning longer and more complex temporal correlations within the data.

The overall concept of the MHA mechanism is to divide the attention components into several parts with different dimensions. The attention computation is then performed on each part individually and the results are then concatenated. This approach allows the system to pay attention to various latent representations sub-spaces information. Consider l as the number of attention heads. MHA can be performed by Eq 7.5:

$$MHA(\mathcal{Q}, \mathcal{K}, \mathcal{M}) = \mathcal{F}_{concat}(\{head_i\}_{i=1}^l) \quad (7.5)$$

$$\text{where } head_i = Attention(\mathcal{Q}, \mathcal{K}, \mathcal{M})_i \quad (7.6)$$

where $\mathcal{F}_{concat}$ responsibility is to merge the output of all attention heads from $i = 1$ to l . STAG will adopt MHA to learn the temporal correlations within the motion history of target vehicles and their SAs.

7.2 Problem Definition

This part formulates the task of behavior prediction in which the primary goal is to anticipate the future behavior of TAs knowing their motion history and those of their heterogeneous SAs at observation horizon t .

We consider the motion history of the TAs and their SAs as the input of the STAG. We denote such history for an agent k as $X_k = [P_k^1, P_k^2, \dots, P_k^T]$ where we have the lateral and longitudinal positions as $P^t = [(x, y)_1^t, (x, y)_2^t, \dots, (x, y)_k^t, \dots, (x, y)_K^t]$ and K is the number of agents within the system which is variable since the proposed approach is evaluated over three different benchmark datasets/scenarios. The input X represents the states of all agents in a dataset, and the positions are relative to the world space.

The output of STAG will be a set of parameters representing a probability distribution of the anticipated vehicles' positions. Considering H as the prediction horizon, we denote the output of the system by Eq 7.7:

$$Y_k = [P_k^{t+1}, P_k^{t+2}, \dots, P_k^{t+H}] \quad (7.7)$$

$$P^{t+h} = [(x, y)_1^{t+h}, (x, y)_2^{t+h}, \dots, (x, y)_k^{t+h}, \dots, (x, y)_K^{t+h}] \quad (7.8)$$

The principal aim is to infer the parameters of Bivariate Gaussian distribution $\mathcal{N} = (\mu, \sigma, \rho)$ over the anticipated positions Y (see Eq 7.9) where μ is the mean, σ is the standard deviation, and ρ is the correlation coefficient:

$$Y_k^t \sim \mathcal{N}(\mu, \sigma, \rho)_k^t \quad (7.9)$$

This way, the anticipation of vehicles' future behavior is defined as obtaining the distribution of future coordinates of multiple target vehicles within a driving scene, given their own and their SAs' motion history. The number of surrounding agents is flexible in various scenarios.

7.3 Methodology

This part presents an overall summary of the primary components and the architecture design of our introduced behavior prediction method, namely STAG. We then further discuss the details of each key module within the system.

7.3.1 STAG Overview

Figure 7.2 highlights the architecture and overview of the major components of the STAG system. At a high level, STAG has an end-to-end encoder-decoder design to extract the latent representations of behaviors history and generate the probability distributions of future behaviors for multiple target vehicles. As illustrated, STAG consists of three key steps, including feature extraction, asymmetric interaction modeling and encoding, and decoding. At first, the dynamic and relative features of target vehicles and their SAs are embedded into high-dimensional vectors using MLP networks. These extracted feature embeddings are then fed to the encoder module. The encoder module consists of two key consecutive sub-modules that are used to represent the asymmetric spatial and temporal correlations among the TAs and their SAs, respectively. In the encoder, dense spatio-temporal graphs are generated employing the embeddings extracted from the previous steps. Attention mechanism and convolution network (see Section 7.3.3) are used to obtain an asymmetric mask which is further employed to learn the spatial correlation scores in the spatial attention layer. The same process is repeated from the temporal aspect after the output of the spatial layer is reshaped. The number of target vehicles and involved SAs may vary depending on the scenarios. Relational reasoning is then performed by adopting blocks of graph convolutions to extract the latent representations of motion history. These representations are then fed to the decoder module. Eventually, the probability distribution for agents' future behaviors are produced by the decoder module using layers of temporal convolutions.

7.3.2 Feature Extraction

STAG feature extraction step has two aspects: first, the dynamic features of target vehicles and their SAs, and second, the relative features between the agents. The key responsibility of this step is to encode different types of features into a high-dimensional space before encoding using an MLP, which can be performed at each t to obtain dynamic (E^D) or relative (E^R) embeddings. Regarding the former, MLP_D embeds the dynamic

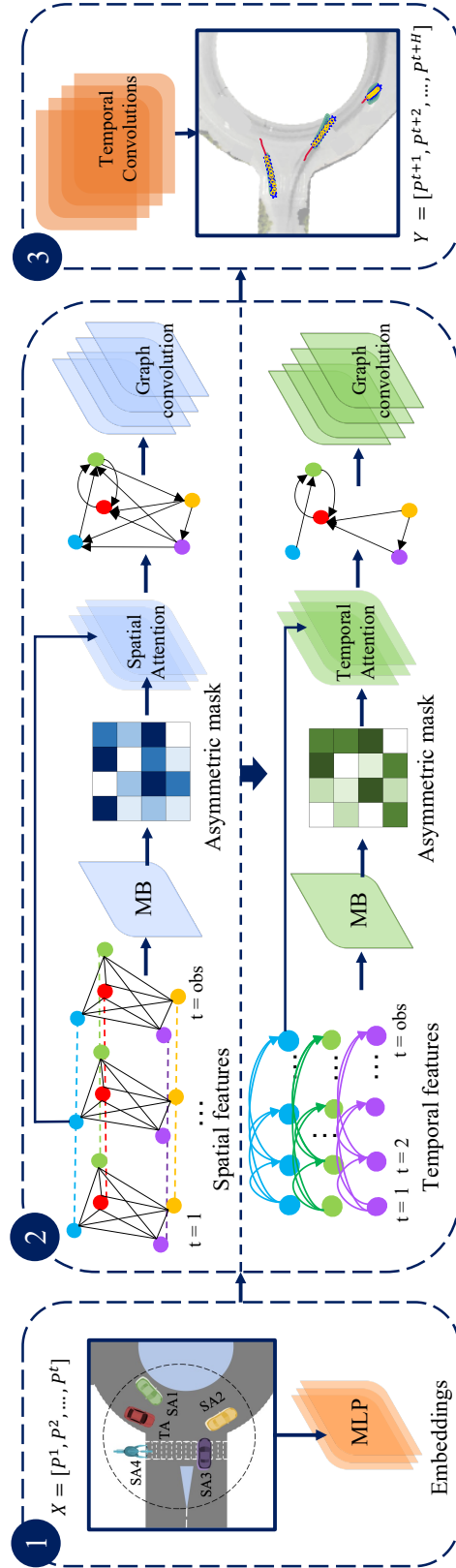


Figure 7.2: An overview of the architecture of STAG including three main steps: (1) The feature extraction that is responsible for extracting the agents states and relations from the agents motion history using MLP; (2) The encoder module that consists of graphs construction, learning asymmetric masks through a mask block (MB), spatial and temporal attention layers to model spatio-temporal interactions, and graph convolutions for obtaining trajectory representations; (3) The decoder module including blocks of TCNs to generate future behavior distribution.

states of agents into a high-dimensional state vector. Although we use agents' positions, note that we utilize the general term dynamic states/features to refer to the features available in various datasets such as agents' positions, velocity, or headings. The SAs of a target vehicle are heterogeneous. Consequently, this process can be conducted for all existing types of agents (if any) in the datasets such as surrounding pedestrians or bicycles. These dynamic embeddings for an agent k are computed by Eq 7.10:

$$E_k^D(t) = MLP_D(\{P_k^t | t = 1, 2, \dots, T\}, W_D) \quad (7.10)$$

where W_D is the weight parameter.

In addition to dynamic features, we embed the relative features between agents pairs into a high-dimensional relation vector using MLP_R . Let k and k' be two agents, ΔP be the relative features between these two agents at timestep t , and W_R be the encoding weight. Then, the relative embeddings for agents k and k' are calculated by Eq 7.11:

$$E_{kk'}^R(t) = MLP_R(\Delta P_{kk'}^t, W_R) \quad (7.11)$$

7.3.3 Encoder Module

Once the feature embeddings are extracted, we generate spatio-temporal graphs $\mathcal{G}^t$ to represent the dynamic and relative information of involved vehicles and SAs. Let K be the number of agents at t . We represent the graphs as $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ where $\mathcal{V} = \{v_k \in \mathbb{R}^{d^D}\}_{k=1}^K$, $\mathcal{E} = \{e_{kk'} \in \mathbb{R}^{d^R} | k, k' \in \{1, \dots, K\}\}$, and the node and edge attributes are dynamic and relative embeddings with dimension d^D and d^R , respectively. The STAG encoder module, as shown in Fig. 7.2, is made of different sub-modules to better capture the spatio-temporal correlations using the attention mechanism, from which it can generate asymmetric interactions. Each sub-module is explained in the following.

Mask Learning

To avoid superfluous interactions with fixed strength of agents' influence, we need to calculate asymmetric masks from the dense interactions among the vehicles and their SAs. Figure 7.3 represents the overall proposed mask learning process. Here, we first compute the interaction scores using MHA mechanism [333] based on the obtained embeddings:

$$\mathcal{Q}_i = \mathcal{F}(E, W_i^{\mathcal{Q}}), \quad (7.12)$$

$$\mathcal{K}_i = \mathcal{F}(E, W_i^{\mathcal{K}}), \quad (7.13)$$

$$\mathcal{S} = \mathcal{F}_{concat}(\{head_i\}_{i=1}^l) \quad (7.14)$$

$$\text{where } head_i = softmax(\frac{\mathcal{Q}_i \mathcal{K}_i^T}{\sqrt{d^{\mathcal{K}}}}) \quad (7.15)$$

where $\mathcal{Q}_i \in \mathbb{R}^{T \times K \times d^{\mathcal{Q}}}$ is the query of MHA, $\mathcal{K}_i \in \mathbb{R}^{T \times K \times d^{\mathcal{K}}}$ is the key of MHA. E is the embedding, $\mathcal{F}(\cdot)$ and W represent the linear transformations and linear transformation learnable weights, and $\frac{1}{\sqrt{d^{\mathcal{K}}}}$ is a scaling factor.

The result of this part is a score matrix $\mathcal{S} \in \mathbb{R}^{l \times T \times K \times K}$ which is computed by concatenation of MHA scores of all heads from 1 to l . In other words, one slice of matrix $\mathcal{S}$ represents the dense interaction value of an agent/vehicle k on another agent/vehicle k' over the l^{th} head at time t . MHA values are obtained from several subspaces [333] which can be combined using a convolution network from the head dimension aspect and then mapped to values between 0 and 1 using the Sigmoid function denoted as σ . Let $\mathcal{L}$ be the kernel of 1×1 convolution kernel; then we will have Eq 7.16:

$$F = \sigma(Conv(\mathcal{S}, \mathcal{L})) \quad (7.16)$$

where $F \in \mathbb{R}^{T \times 1 \times K \times K}$ represents the asymmetric correlations. We then generate the correlation mask based on which the updated adjacency matrix for spatio-temporal graphs can be constructed in the future steps. To do so, a threshold-based mask is defined over the obtained asymmetric correlations. We can consider this threshold ε as another learning hyper-parameter of STAG. The *Mask* is then calculated by Eq 7.17:

$$Mask = \mathcal{I}[F \geq \varepsilon] \quad (7.17)$$

where $\mathcal{I}$ is the characteristic function that outputs 1 if the provided condition holds. The obtained mask will be further employed in the subsequent steps to update and obtain directed graphs with asymmetric interactions strength.

Spatial Correlations

Capturing the spatial social interactions between the vehicles and their SAs plays a pivotal role in the behavior prediction problem. We adopt spatial attention [334] to learn

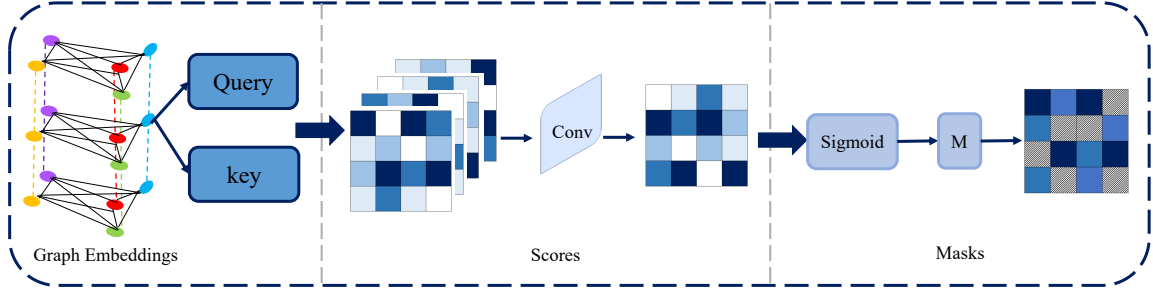


Figure 7.3: An illustration of the mask block of STAG with attention mechanism to compute the asymmetric masks.

the dense spatial correlations between the target vehicles and surrounding agents and apply our generated mask to the scores. This sub-module receives the spatio-temporal graphs and the mask generated in the previous sub-module and output the updated graphs from spatial aspects where $\mathcal{V}^t = \{v'_k \in \mathbb{R}^{d'}\}_{k=1}^K$ at time t . The aim is to recognize the group of vehicles involved in a scene in the graphs. Consider $N(k) = \{k' \in \mathcal{V}^t | (k, k') \in \mathcal{E}^t\}$ as the set of surrounding neighbors of vehicle k in $\mathcal{G}$. Then, the initial attention scores A are computed by Eq 7.18:

$$a_{kk'}^t = \frac{\exp(\alpha \|v_k^t - v_{k'}^t\|^2 + \beta \|e_{kk'}^t\|^2)}{\sum_{k'' \in N(k)} \exp(\alpha \|v_k^t - v_{k''}^t\|^2 + \beta \|e_{kk''}^t\|^2)} \quad (7.18)$$

where α and β are weighting parameters to control the balance between the node and edge attributes. The motivation is that the SAs having more similar attributes to the target vehicle or having less distance from it demonstrate stronger correlations, which requires more attention. The generated attention scores are then fused with the mask by Eq 7.19 to gain the attention matrix.

$$A' = (Mask + I) \odot A \quad (7.19)$$

where $\odot$ represents element-wise multiplication and I denotes an identity matrix to make sure the node v_k can preserve its own information i.e., node v_k is correlated to itself. We compute the final attention coefficients $\mathcal{C}$ by normalizing A' using Eq 7.20:

$$c_{kk'} = \text{normalize}(A') = \frac{(\exp(a'_{kk'}) - 1)^2}{\sum_{k'' \in N(k)} (\exp(a'_{kk''}) - 1)^2 + \epsilon} \quad (7.20)$$

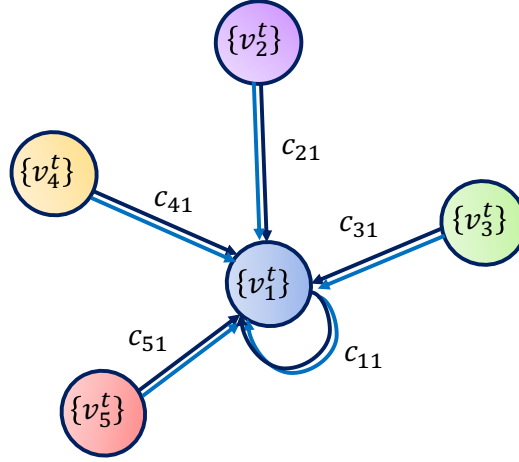


Figure 7.4: An illustration of spatial attention module to model the spatial interactions for the target vehicle v_1 .

where ϵ in the denominator is a small value used for function stability. Zero-softmax function [224] is used, as opposed to the softmax function, to guarantee that zero values remain zero.

Once the spatial attention coefficients are calculated, we can perform graph aggregation using Eq 7.21:

$$v_k'^t = \delta \left(\sum_{k' \in N(k)} c_{kk'}^t W v_{k'}^t \right) \quad (7.21)$$

where $\delta(\cdot)$ and W are activation function and weight parameter, respectively. Figure 7.4 demonstrates our message passing process where we benefit from MHA for information aggregation. In other words, we update the node attributes by Eq 7.22:

$$v_k'^t = \left\| \right\|_{i=1}^l \delta \left(\sum_{k' \in N(k)} c_{kk'}^{ti} W^i v_{k'}^t \right) \quad (7.22)$$

where there are i attention heads and W^i is the weight parameter of i^{th} head.

The output of this sub-module will serve as the input of the temporal attention layer to take into account the motion tendency.

Temporal Correlations

Following learning spatial correlations using attention and asymmetric mask, we obtain the temporal correlations with a similar approach to the one for spatial correlations as illustrated in Fig. 7.2. As can be seen, the output of the previous sub-module is fed to the temporal layer after being reshaped to its proper form to gain the corresponding intermediate representations, i.e., the final output of the encoder module. To be exact, the inputs of this part are the set of $\mathcal{V}^t = \{v'_k \in \mathbb{R}^{d'}\}_{k=1}^K$ and the outputs are the spatio-temporal behavior representations that are further employed to generate the behavior distribution. This sub-module creates the asymmetric mask using our proposed mask learning block explained in Section 7.3.3 and fuses it with initial attention scores in a similar way to the previous sub-module with two differences. First, the initial temporal attention scores A_{tp} are calculated by Eq 7.23:

$$a_k^t = \frac{\exp(\delta(v_k'^t W))}{\sum_{t'=1}^{T_{obs}} \exp(\delta(v_k'^{t'} W))} \quad (7.23)$$

where $\delta(\cdot)$ is an activation function and W is a learnable parameter. Second, the message passing process is calculated using Eq 7.24:

$$v_k'' = \left\|_{i=1}^l \delta \left(\sum_{t=1}^{T_{obs}} c_k^{ti} W^i v_k'^t \right) \right. \quad (7.24)$$

7.3.4 Decoder Module

As discussed in Section 7.2, the principal aim of STAG is to anticipate the parameters of Bivariate Gaussian distribution $\mathcal{N} = (\mu, \sigma, \rho)$ over the future positions of vehicles. The decoder task is to process the behavior representations extracted by the previous module and produce the final distribution of the future vehicles' behaviors. To do so, in this work, we benefit from temporal convolutions. Compared to RNN-based sequence modeling architectures, TCNs have proven to obtain better results while reducing the computational complexity [308] in several research areas such as machine translation. We directly apply stacked layers of dilated convolutions to the features extracted by the encoder. Furthermore, STAG is trained end-to-end using our training loss function. STAG training loss consists of two sections, including a likelihood $\mathcal{L}_l$ and a distance $\mathcal{L}_d$ loss denoted as Eq 7.25:

$$\mathcal{L} = \mathcal{L}_l + \mathcal{L}_d \quad (7.25)$$

where:

$$\mathcal{L}_l = - \sum_{t=t-H+1}^{t+H} \log P((x, y)_k^t \mid (\mu, \sigma, \rho)_k^t) \quad (7.26)$$

$$\mathcal{L}_d = \frac{1}{H} \sum_{t=t+1}^{t+H} ((\hat{x}, \hat{y})_k^t - (x, y)_k^t)^2 \quad (7.27)$$

where H indicates the horizon. The former term aims at maximizing the likelihood of future behaviors, and the latter term aims at minimizing the L_2 distance loss between the true and anticipated behaviors of the vehicles.

7.4 Experimental Results

This section discusses how we evaluate the performance of the proposed behavior prediction method, namely STAG, and make a comparison with those of various benchmarks over three different scenarios, including structured highways, unsignalized intersections, and roundabouts. We first provide insights on the evaluation datasets followed by the employed metrics. Comparison methods are then discussed in brief. We then summarize the quantitative and qualitative results.

7.4.1 Evaluation Datasets

To validate the performance of STAG and thoroughly explore its capacity in modeling social interactions among agents in diverse road scenarios, we carefully consider three public datasets each of which is briefly introduced in the following. Each dataset is specifically chosen for its inclusion of comprehensive information on surrounding agents. This criterion is essential as it allows us to extract and analyze interactions among agents, providing a robust evaluation of STAG’s effectiveness across a range of realistic environmental conditions.

- **NGSIM.** For a highly structured scenario, we select NGSIM dataset (US 101 [88] and I-80 [89]) that contains naturalistic vehicle trajectories captured from overhead cameras over freeways. The initial data is collected at 10Hz and includes

more than 40 minutes of driving scenes divided into three different parts based on traffic conditions, from mild to heavy traffic. This dataset contains data on vehicles only, which allows us to test the STAG for homogeneous SAs for a specific target vehicle. To have an objective comparison, we have conducted the exact preprocessing steps as the ones done by state-of-the-art approaches [130, 133, 231, 235] on this dataset: (1) we downsample both trajectories by a factor of two to decrease the computational cost, (2) the dataset is divided into windows of 8s, having 3s as the observation window and up to 5s for prediction horizon (i.e., more than 13000 segments), and (3) the training and test split ratio is 3:1.

- **InD.** Anticipating the future behavior of vehicles in more dynamic and complex conditions is more challenging. One example of such scenarios is unsignalized interactions where we do not have any traffic lights to control the flow, or multiple lanes in each direction to have prior clues on the future intention of the involved agents. For this scenario, we use inD [78] dataset that consists of more than 11000 trajectories and 10 hours of drone recordings from four German unsignalized intersections, all of which we select for STAG evaluation. We use all intersections each of which contains data collected at 25 Hz from various types of traffic participants, including vehicles, cyclists, and pedestrians. Following existing works [139, 226, 227] on this dataset, we downsample data by a factor of ten and choose observation and horizon windows of 8 (3.2s) and 12 (4.8s) steps, respectively. Also, We follow the same train and validation split.
- **INTERACTION.** Prediction at roundabouts as another example of complex scenarios with no explicit traffic rules is also a challenging task. INTERACTION [82] dataset is a large collection of naturalistic driving behaviors with top-view recordings (with a frequency of 10 Hz) from several environments, including roundabouts. Furthermore, the dataset contains diverse driving behavior such as aggressive driving. We select four of the existing roundabouts, namely CHN-LN (roundabout 1), USA-SR (roundabout 2), DEU-OF (roundabout3), and USA-EP (roundabout 4), which contain more than 155 minutes of data from about 2980 agents for each of which 4s of data is available. We downsample data by a factor of two, and out of the available steps, we observe 8 steps of motion history and predict 10 and 12 steps of future behaviors.

7.4.2 Evaluation Metrics

Following existing works on the used datasets, we adopt three well-established error metrics that are frequently used to assess the performance of behavior prediction approaches.

- **RMSE.** To be able to compare with benchmark approaches, We utilize Root Mean Square Error (RMSE) values for experiments conducted on NGSIM dataset. The values are computed as Eq 7.28:

$$RMSE = \sqrt{\frac{1}{H} \sum_{t+1}^{t+H} (\hat{p}^t - p^t)^2} \quad (7.28)$$

- **ADE.** We calculate Average Displacement Error (ADE) by taking the mean distance between obtained results and the real behaviors for each TA over prediction horizon. The formal definition can be summarized as Eq 7.29:

$$ADE = \frac{\sum_{k=1}^K \sum_{t+1}^{t+H} \|\hat{P}_k^t - P_k^t\|_2}{K.H} \quad (7.29)$$

- **FDE.** We also report Final Displacement Error (FDE) representing the mean distance between the obtained results and real trajectories only at the last predicted step. We denote FDE as Eq 7.30:

$$FDE = \frac{\sum_{k=1}^K \|\hat{P}_k^t - P_k^t\|_2}{K}, t = t + H \quad (7.30)$$

The last two metrics are used in experiments conducted on inD and INTERACTION datasets. For all three metrics, a smaller value demonstrates higher performance.

7.4.3 Experimental Settings

This part summarizes the implementation details used in this work. We have trained STAG with a batch size of 128 for 150 epochs using Adam optimizer. The learning rate is 0.001 with a decay factor of 0.1 every 50 epochs. The MLP in the feature extraction step has three layers with 128 units. The dimension of graph and attention embeddings is 64, and there are 4 attention heads. The threshold at the mask learning process, i.e., ε , is empirically obtained as 0.5. Moreover, LeakyReLU is employed as nonlinear activation in the model. A grid search is employed to optimize the hype-parameters of all modules of

the STAG. At test time, several potential behaviors (N) are sampled from the Gaussian distribution, and the closest one to the true values is used to obtain evaluation metrics (ADE/FDE). Following prior studies, we use $N = 20$ for the evaluation unless specified otherwise.

7.4.4 Baseline Models

We conduct performance evaluation on several baselines and benchmarks based on the evaluations scenarios so that we can compare the performance of STAG with benchmark approaches on each used dataset. We explain the chosen models in the following.

- *Highways scenario.* In this scenario, we report the results from 10 baselines and benchmarks and compare them with those obtained by our proposed approach, STAG. We use constant velocity (CV) as the baseline. *M-LSTM* [221] uses an LSTM-based encoder to encode the relative positions and motion history of the vehicles into context vectors which are later combined with maneuver encodings. We also compare the STAG results with those obtained by three social pooling-based approaches including *S-LSTM* [233], *S-GAN* [134], and *CS-LSTM* [231]. In addition, our comparison includes the most recent approaches such as *MATF-GAN* [243], *PJTSM* [209], and *MHA-LSTM* [133] which is an attention-based approach that generates the future behavior by using LSTM-based decoder, and two undirected graph-based models including *GRIP* [230] and *HSTA* [130].
- *Unsignalized intersection scenario.* In this scenario, in addition to two pooling-based baseline approaches *S-LSTM* [233] and *S-GAN* [134], we compare the STAG results on inD dataset with those of four of the most recent prediction approaches including *AMENet* [227], *DCENet* [139], *SCOUT* [226], and *Grip++* [237]. The first two approaches work based on Conditional Variational Auto-Encoder (CVAE) generative models. *SCOUT* is a socially-consistent approach that constructs a dense spatio-temporal graph with three different aggregation schemes. *Grip++* benefits from graph-based interaction reasoning as well. It constructs a dense graph in which the edges represent interaction existence without considering their strength.
- *Roundabout scenario.* In this scenario, we assess the ability of the STAG in terms of generalization over unseen driving environments. We evaluate its performance over unseen scenarios, i.e., roundabouts. More specifically, the training data related

to four selected roundabouts from the INTERACTION dataset are not used. We compare the obtained results over validation data for prediction steps 10 and 12 with those of the *S-GAN* [134] model.

7.4.5 Quantitative Evaluation

This part summarizes and discusses the prediction results for all scenarios:

Structured Highways

We assess the performance of our proposed approach, namely STAG, over the NGSIM dataset and compare it with that of models mentioned in Section 7.4.4. Table 7.1 represents the prediction performance of the STAG and the benchmark approaches in terms of RMSE, where the observation window is 3s and the prediction horizon varies from 1s to 5s. All methods consider social interactions among the TV and all of its homogeneous SAs (NGSIM includes data related to vehicles only) with their specific approach except for the baseline model CV. Taking advantage of social pooling as a tool to incorporate interactions among vehicles, *S-LSTM* and *CS-LSTM* improve the baseline results. *S-GAN* enhances their results by benefiting from generative modeling. *MHA-LSTM* moves the improvement one step forward by taking into account the attention-based weighted social interactions among the vehicles. Comparing the results of *MHA-LSTM* with those of *HSTA* indicates the advantages of spatio-temporal modeling for long-term predictions (horizons 4 and 5). As demonstrated, STAG outperforms all compared models, especially at long-term predictions, by explicitly considering sparse and asymmetric social interactions among vehicles and constructing spatio-temporal directed graphs. For instance, the results over horizon 3s indicate that the error value of 1.27 is reduced by 12.4% and 27% compared to the two best previous results.

Unsignalized Intersections

The STAG results over the inD dataset are summarized and compared with benchmark approaches in Table 7.2 where we report ADE and FDE values for all intersections while using 8 and 12 as observation and prediction steps, respectively. Overall, our introduced method achieves the lowest prediction error. For instance, STAG reduces ADE by 23% and FDE by 39% compared to the following best approach over Bendplatz unsignalized intersection. Comparing the results of generative models such as *AMENet* and *DCENet* with that of the *S-LSTM* indicates the capability of CVAE architectures

Table 7.1: Evaluation results of structured highways scenario over NGSIM dataset for various state-of-the-art approaches. RMSE values in meters are reported for 3s observation. STAG outperforms all baselines.

Horizon(s)	CV	S-LSTM	M-LSTM	CS-LSTM	S-GAN	MATF GAN	MHA-LSTM	GRIP	HSTA	PJTSM	STAG
1	0.73	0.65	0.58	0.61	0.57	0.66	0.41	0.37	0.52	0.52	0.37
2	1.78	1.31	1.26	1.27	1.32	1.34	1.01	0.86	1.12	1.33	0.83
3	3.13	2.16	2.12	2.09	2.22	2.08	1.74	1.45	1.82	2.28	1.27
4	4.78	3.25	3.24	3.10	3.26	2.97	2.67	2.21	2.61	2.88	2.04
5	6.68	4.55	4.65	4.37	4.40	4.13	3.83	3.16	3.49	3.49	2.74

Table 7.2: Comparison results (ADE / FDE) over inD dataset for obs=8 and H=12 steps.

Intersections	S-LSTM	S-GAN	Grip++	SCOUT	AMENet	DCENet	STAG
Strasse	2.29 / 5.33	3.02 / 5.30	1.65 / 3.65	0.67 / 1.55	1.07 / 2.22	0.96 / 2.12	0.51 / 1.30
Bendplatz	1.28 / 3.19	1.55 / 3.23	0.94 / 2.06	0.48 / 1.08	0.65 / 1.46	0.64 / 1.41	0.37 / 0.66
Frankenburg	1.78 / 4.24	2.22 / 4.45	0.59 / 1.41	0.30 / 0.69	0.83 / 1.87	0.86 / 1.93	0.49 / 1.06
Heckstrasse	2.17 / 5.11	2.71 / 5.64	1.94 / 4.46	0.40 / 0.83	0.37 / 0.80	0.28 / 0.62	0.30 / 0.48
Average	1.88 / 4.47	2.38 / 4.66	1.28 / 2.88	0.46 / 1.03	0.73 / 1.59	0.69 / 1.52	0.42 / 0.88

Table 7.3: ADE / FDE results for different types of agents involved within a scene over inD dataset.

Intersections	SCOUT				STAG			
	Vehicles	Pedestrians	Bicycles	All	Vehicles	Pedestrians	Bicycles	All
Strasse	0.67 / 1.54	0.50 / 1.12	0.29 / 0.65	0.67 / 1.55	0.49 / 1.29	0.38 / 0.66	2.65 / 5.30	0.51 / 1.30
Bendplatz	0.41 / 0.91	0.54 / 1.21	0.46 / 1.01	0.48 / 1.08	0.35 / 0.66	0.43 / 0.77	1.28 / 2.77	0.37 / 0.66
Frankenburg	0.18 / 0.41	0.52 / 1.15	0.51 / 1.14	0.30 / 0.69	0.27 / 0.41	0.39 / 0.68	1.07 / 2.11	0.49 / 1.06
Heckstrasse	0.39 / 0.80	0.41 / 0.84	0.23 / 0.48	0.40 / 0.83	0.28 / 0.38	0.28 / 0.41	0.97 / 1.98	0.30 / 0.48

in capturing the interactions among agents as well as trajectory embeddings. It also demonstrates that considering multiple behaviors yields better results. *SCOUT* improves the results by incorporating graph-based spatial modeling. Furthermore, the results indicate that STAG can effectively model the spatio-temporal correlations among a TA and its heterogeneous SAs and accurately predict multiple agents' future behaviors.

Moreover, in this scenario, we perform an analysis of the impact of different types of agents involved within a scene. Table 7.3 represents the obtained prediction errors for STAG and *SCOUT* models while considering different types of agents each of which has a different mobility pattern. STAG outperforms *SCOUT* by a significant margin concerning three intersections for vehicles and pedestrians. Comparing the results of STAG and *SCOUT* shows how considering asymmetric social interactions can improve the performance of spatio-temporal interaction reasoning.

Roundabouts

As discussed in Section 7.4.4, to check the generalization ability of the STAG, we assess its performance in a totally unseen and complex scenario such as roundabouts. For this experiment, we train STAG over two unsignalized intersections from the inD dataset and assess its generalization ability over validation data from selected roundabouts of the INTERACTION dataset (see Section 7.4.1 for more details). The exact process is done for *S-GAN* approach, and the errors are summarized in Table 7.4. As summarized, compared to *S-GAN*, STAG demonstrates lower error values, i.e., better performance, representing its higher ability to generalize over unseen data not only from the same type of environments (e.g., two previous scenarios) but also from a totally new driving context that is roundabouts in our case. STAG reduces ADE by 33% and FDE by 22% while predicting 12 steps.

Effect of Asymmetric Mask

As shown in Table 7.5, we assess several versions of STAG, by which we analyze the impact of the masking threshold in the mask learning process on the overall performance of STAG. We compare four different STAG variants using the same experimental settings as the roundabout scenario. The average results are reported for 8 and 12 predictions steps. Version 1 (STAG-v1) refers to the situation where we set $\varepsilon = 1$, meaning that there exist no social correlations among each pair of agents. In the second version (STAG-v2), we perform experiments while considering $\varepsilon = 0.75$, which indicates only a very small

Table 7.4: Experimental results of validation on unseen scenario i.e., roundabouts (R1-R4) from INTERACTION dataset. ADE / FDE in meters are reported for obs=8 steps.

Scenario	S-GAN		STAG	
	10	12	10	12
Roundabout 1	0.78 / 1.35	1.22 / 1.79	0.48 / 0.92	0.69 / 1.20
Roundabout 2	0.47 / 0.83	0.72 / 1.28	0.30 / 0.60	0.49 / 0.95
Roundabout 3	0.74 / 1.39	1.21 / 2.06	0.60 / 1.01	0.87 / 1.62
Roundabout 4	0.44 / 0.71	0.69 / 1.03	0.33 / 0.70	0.53 / 1.03
Average	0.61 / 1.07	0.98 / 1.54	0.43 / 0.81	0.65 / 1.20

Table 7.5: Ablation analysis of mask threshold ε for prediction steps of 10 and 12.

Our model	10	12
STAG-v1 ($\varepsilon = 1$)	0.60 / 1.11	0.79 / 1.40
STAG-v2 ($\varepsilon = 0.75$)	0.47 / 0.84	0.68 / 1.28
STAG-v3 ($\varepsilon = 0$)	0.51 / 0.86	0.76 / 1.34
STAG	0.43 / 0.81	0.65 / 1.20

number of correlations are considered. The third version (STAG-v3) represents $\varepsilon = 0$, meaning all social interactions among TAs and SAs are considered with the same effect on each other. The last row represents the results of our full model where $\varepsilon = 0.5$. The obtained prediction errors are summarized in Table 7.5. As illustrated, avoiding superfluous interactions among the agents to a certain point can enhance the performance of spatio-temporal interaction modeling. Comparing the results of STAG-v3 and STAG shows the higher performance of STAG, demonstrating that using an asymmetric mask can improve performance. For instance, the error values are reduced by 14.4% and 10.5% for ADE and FDE for prediction steps 10 and 12, respectively. Also, the higher ADE/FDE error values for STAG-v1 highlight the significance of interaction reasoning.

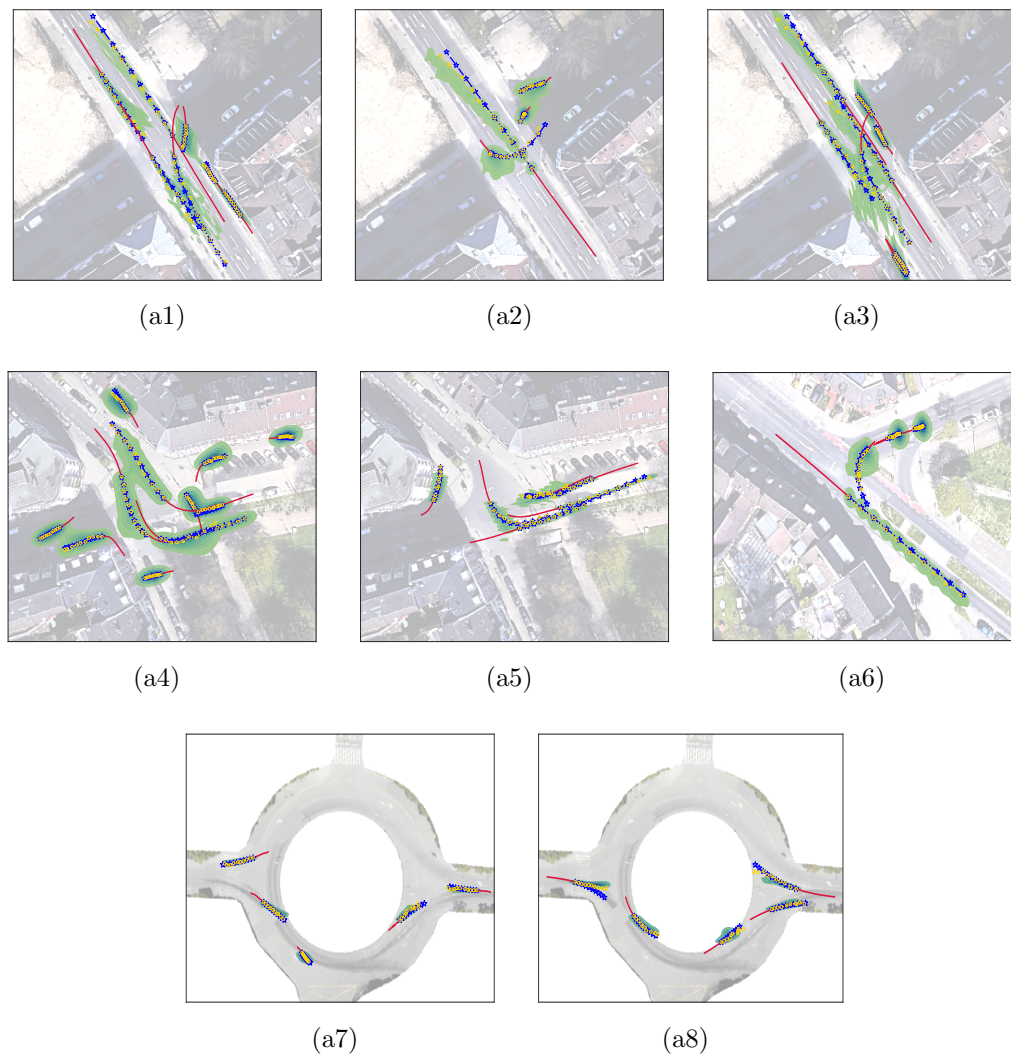


Figure 7.5: Qualitative analysis of predicted trajectories for the proposed STAG model over different scenarios: inD/Bendplatz (a1-a3), inD/Frankenburg (a4-a5), inD/Heckstrasse (a6), and INTERACTION/Roundabout 1 (a7-a8). The red and blue/star lines indicate the observation and true behaviors, respectively. The predicted distribution and the best sample are shown with green masks and yellow/circle lines, respectively.

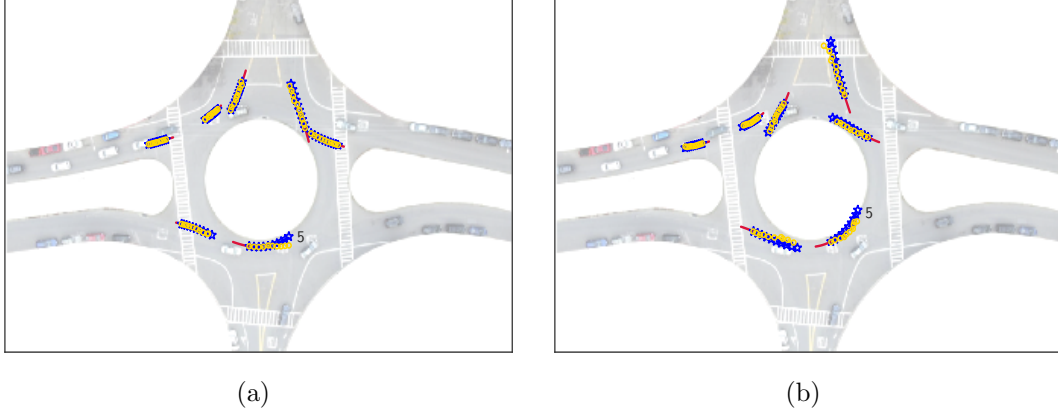


Figure 7.6: Qualitative analysis of the effect of vehicles mobility over INTERACTION dataset (Roundabout 2).

7.4.6 Qualitative Evaluation

The results of qualitative analysis are demonstrated in Fig 7.5 taken from several driving environments. Note that the qualitative study is conducted while setting the prediction horizon for 10 steps. As shown, our model, STAG, precisely predicts the potential future behaviors of multiple agents within a scene in several complex scenarios. Consider Fig. 7.5a1; STAG accurately anticipates several fast-driving vehicles moving toward each other or performing a turn maneuver while considering the possible interactions between the cyclist and two pedestrians. The green shadows represent the prediction distribution. Solid lines indicate the observation, and blue/star and yellow/circle lines represent the true behaviors and the best samples, respectively. STAG also correctly anticipates the behavior of TAs whose future direction may cause conflict, such as those shown in Fig 7.5a4, where the vehicle on the right side stops to give the right of way to a person walking out of the park. The results also indicate the situations where STAG had some difficulty in prediction, such as the vehicle entering the roundabout on the left side of Fig 7.5a8. Overall, the results show the capability of STAG in handling various driving environments, different maneuvers from going straight to turning maneuvers and yielding, and heterogeneous SAs.

We also carry out a qualitative analysis of the effect of vehicles mobility on the performance of STAG. Figure 7.6 illustrates a sample scenario (from the INTERACTION dataset) for seven variant vehicles each of which is going around the roundabout. Two different sequences of observations and predictions are shown for each vehicle to illustrate

how STAG manages to converge to the best possible behavior. Consider the vehicle labeled number 5. At the early step (Fig 7.6a), when the TA has just entered the roundabout, STAG has difficulty predicting its future behavior, and the results deviate from the ground truth. However, as the vehicle moves further toward the roundabout's center and starts to perform its turning maneuver, the model's performance enhances. Although with a different angle and speed, the model anticipates the turning maneuver.

7.5 Concluding Remarks

In this chapter, a spatio-temporal behavior prediction system architecture is introduced that takes advantage of social interaction reasoning in general and asymmetric modeling in particular. Our model, namely STAG, works based on capturing spatial and temporal correlations using graph convolutions and attention mechanism and anticipating the future behaviors with temporal convolutions. Unlike several prior works, STAG avoids superfluous and fixed interactions among agents by applying an attention-based and threshold-based mask over the dense social interactions. We assess the prediction performance of STAG over three variant scenarios, including highways, unsignalized intersections, and roundabouts. The following points can be concluded according to the results obtained. Quantitative results reveal that STAG is able to precisely model the spatio-temporal interactions among both heterogeneous and homogeneous SAs, and demonstrates better performance compared with several benchmarks in terms of prediction error. It can also successfully generalize over unseen environmental scenarios. Qualitative results indicate STAG capability in handling different driving behaviors, e.g., turning or yielding, and heterogeneous SAs. An important aspect to acknowledge in our study is that the performance of STAG has not been specifically tested on a real-world autonomous vehicle. While our experiments have yielded valuable insights and assessments within the utilized settings, it is crucial to recognize that the evaluation has primarily taken place using naturalistic datasets.

Chapter 8

CAPHA: Context-Aware Behavior Prediction of Heterogeneous Agents

Throughout the preceding chapters, it becomes evident that the capability of AVs to navigate safely and successfully through complex and dense driving scenes hinges on their ability to develop situational awareness. One significant challenge involves estimating the distribution of future motions of surrounding agents while accounting for uncertainty in human behavior, with the aim of generating paths that are both realistic and scene compliant (see Fig. 8.1). Therefore, effective prediction methods must accurately represent such multimodality in the output space matching the inherent distribution. Different methods have been explored to address the diversity of behavior patterns in behavior prediction. Regression-based approaches [250, 251] consider various possible behaviors, but research has shown [134, 231] that they intend to average various possible behavior modes, which may not be a feasible and scene-compliant behavior for an SA. Some recent studies [265, 266, 335, 336] have used latent variables to capture different behavior modes. Although promising, these proposed approaches still suffer from inherent limitations such as their interpretability which makes it difficult to include expert knowledge within the model and real-world suitability for stochastic sampling at execution time. Alternatively, mixture models [39, 253, 254] assign behavior modes to different mixture components, creating a distribution of potential behaviors. Yet, these methods face challenges in mapping complex agent features to corresponding behaviors, making optimization difficult and prone to overfitting.

In this chapter, we propose a novel Context-Aware behavior Prediction system for Heterogeneous Agents (CAPHA) to alleviate these shortcomings. We argue that condi-

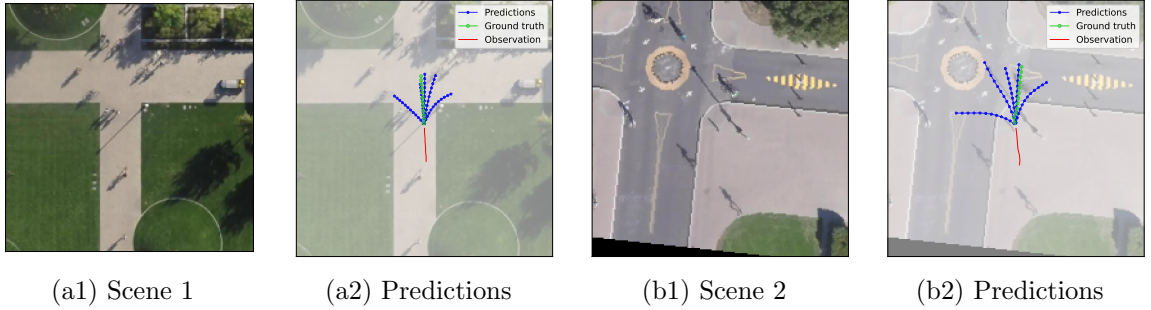


Figure 8.1: Multiple modes of the future behavior. The predicted behaviors are scene-compliant.

tioning future behaviors of agents¹ on grid-based plans derived from the traffic scenes can help the prediction model to capture the underlying correlations between the scene and SAs' future behaviors. This will allow CAPHA to not only produce more scene-compliant behaviors for both vehicles and pedestrians but also better generalize to unknown driving environments. We propose to incorporate both kinematic and scene features in an Inverse Reinforcement Learning (IRL) framework, where an SA is modeled as a Markov Decision Process (MDP), and the state rewards are gained using Maximum Entropy IRL (ME-IRL). In short, our main contributions are as follows: We design CAPHA, a novel planning-based behavior prediction framework for heterogeneous agents with context awareness which uses rasterized Bird's-Eye-View (BEV) images of the surrounding environment to forecast future behaviors conditioned on plans learned by IRL-based policies on a two-dimensional grid. Hence, the model can produce behaviors that conform to the environmental circumstances and are plausible for practical deployment. We compute policy maps benefiting from Value Iteration Network (VIN) and perform sampling via Gumbel Softmax technique, leading to more robust plan sampling. We propose a novel behavior generation architecture, where we adopt stacked transformers to integrate the agents' motion and context information and multi-layer perceptron to produce future behaviors conditioned on plans sampled from policy maps. Thus, the model can better address the multimodality and interpretability of the prediction problem.

The remainder of this chapter is categorized into four sections: Section 8.1 highlights the relevant IRL preliminaries used in our introduced methodology. Section 8.2 describes how we define the prediction problem and model it as an MDP. Section 8.3 discusses

¹In the context of agent behavior prediction, the term 'agent' refers to different road participants including drivers, cyclists, and pedestrians.

the critical aspects of our proposed model, CAPHA, in detail. Section 8.4 explains the evaluation settings and datasets and reports the experimental results. Finally, Section 8.5 concludes the key points observed in this chapter.

8.1 Background

This part highlights the relevant markov decision process and inverse reinforcement learning primary concepts and definitions utilized in the CAPHA backbone.

8.1.1 Markov Decision Process

Sequential decisions and Reinforcement Learning (RL) problems can be mathematically modeled as an MDP where there exists a set of concepts, including (1) an agent defined as the entity being trained to learn to draw the right decisions within an RL system, (2) a finite set of states which refer to the circumstances of the agent at any given moment, (3) an environment which is the surrounding context of an agent with which it has interactions but cannot change its circumstances, and (4) a set of actions that represent a set of options available to the agent to choose from at any given moment. The set of actions that an agent can carry out is predefined. However, it is the agent that should learn which actions to carry out at any given state and moment by interacting with the environment. In general, an MDP problem is formulated using a set of $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ represents the underlying state space, $\mathcal{A}$ demonstrates the available actions, $\mathcal{T}$ indicates the probability matrix representing the probability of the transition of an agent to state s_2 performing a given action in state s_1 , and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ shows the cost or reward function. In other words, an MDP is an extension of Markov Chains. The ultimate purpose will be the minimization or maximization of the accumulated costs or rewards, respectively. Furthermore, a solution can be obtained w.r.t a policy π representing the probability distribution that maps $\mathcal{S}$ to the actions. The higher the probability of the action, the more rewarding the action. These policies will establish how an agent behaves.

8.1.2 Inverse Reinforcement Learning

In the inverse scenario, i.e., IRL, the purpose is to estimate the rewards $\mathcal{R}$ of a learner, given optimal policies π or expert knowledge [140]. Here, $\mathcal{R}$ of the MDP is not known

and the learner is to go by expert behaviors $\mathcal{E} = \{e_1, e_2, \dots, e_K\}$ sampled from an optimal policy π . Each behavior includes pairs of state-actions $e_k = \{(s_0, a_0), (s_1, a_1), \dots, (s_i, a_i)\}$ and the ultimate purpose is to learn the rewards from these behaviors. One approach [337] is to formulate the IRL problem as ME-IRL, where behaviors showing higher rewards have exponentially more probability:

$$P(e|\mathcal{R}) \propto \exp \sum_{s,a \in e} \mathcal{R}_{s,a} \quad (8.1)$$

Learning from the expert behavior is equivalent to obtaining a reward map that maximizes the likelihood of the demonstrated behavior under maximum entropy distribution:

$$\underset{\theta}{\operatorname{argmax}} \mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{e_k \in \mathcal{E}} \log P(e_k|\mathcal{R}_\theta) \quad (8.2)$$

where θ denotes reward weights and

$$P(e_k|\mathcal{R}_\theta) = \frac{1}{Z(\theta)} e^{\sum_{s_i \in e_k} \theta^T \mathcal{R}(s_i)} \quad (8.3)$$

Our proposed model, CAPHA, benefits from MDP to model the prediction task and IRL to encode the scene information and increase the generalization ability of the prediction model since the rewards are intrinsically more transferable than the policies, i.e., a subtle change in the surrounding context does not influence the rewards transferability.

8.2 Problem Formulation

In this section, we formulate the motion forecasting task whose objective is to predict the SAs' future behaviors given their history and some knowledge of the surrounding context at horizon t . More specifically, considering an observation σ for an agent (e.g., a vehicle or pedestrian), the primary purpose is to anticipate the probability distribution $P(\mathcal{Y}|\sigma)$ of its upcoming behaviors represented as $\mathcal{Y} = \{\mathcal{Y}^T \in \mathbb{R}^2 | T = t, \dots, t + H\}$, where H indicates the forecasting window. In our scenario, the observation σ for an SA includes BEV images of its surrounding environment centered at its current location as well as its behavior history which is represented as $\mathcal{X} = \{\mathcal{X}^T \in \mathbb{R}^2 | T = 1, \dots, t - 1\}$.

We mathematically formulate the forecasting task as a deterministic MDP problem in which the SAs' future behaviors are conditioned on grid-based state sequences sampled from the optimal expert behavior. We express this decision process mathematical model as a set of $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ where:

- $\mathcal{S}$ indicates possible states each of which is made up of cells on a two-dimensional grid considered over the surrounding context. An agent behavior can be considered as a series of states $(s_0, s_1, \dots, s_I)$, where s indicates the agent's location.
- $\mathcal{A}$ is the set of actions (i.e., discretized possible motions) for the agents moving over the grid, including four directions north, south, east, west, and a final action indicating the agent has reached the end state of MDP.
- $\mathcal{T}$ is the deterministic probability matrix representing the probability of the transition of an SA to state s_j performing an action in state s_i .
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents the reward functions, i.e., one for the SAs' goals and one for the ways reaching them.

Under this formulation, the agent employs a non-stationary (i.e., dependent on the MDP steps) policy π_θ^I to identify which actions to choose at a particular MDP step and estimate the state sequence, essentially predicting its future behaviors.

8.3 CAPHA for Behavior Prediction

We model the behavior prediction task as an MDP problem and solve it through a Maximum Entropy IRL pipeline to benefit from its transferability. Figure 8.2 presents the architecture of CAPHA, our proposed method, and summarizes its key components. As illustrated, CAPHA is made up of three principal modules, including context encoder, policy network, and behavior decoder. Context encoder is responsible for approximating $\mathcal{R}$ function given the BEV images of the scene and agents behavior history. At each cell over the defined grid, this module maps the context and kinematic features to rewards. Using these obtained rewards, the policy network module generates the optimal policy π_θ^I adopting a value iteration network. This policy is then sampled using the Gumbel Softmax technique to consider a variety of plausible goals for the SAs. The sample plans are later fed to the behavior decoder whose responsibility is to generate the behavior distribution. The decoder benefits from stacked transformers to encode the behavior history and sampled plans and MLP to produce behaviors conditioned on the plans. More details of each module are elaborated below.

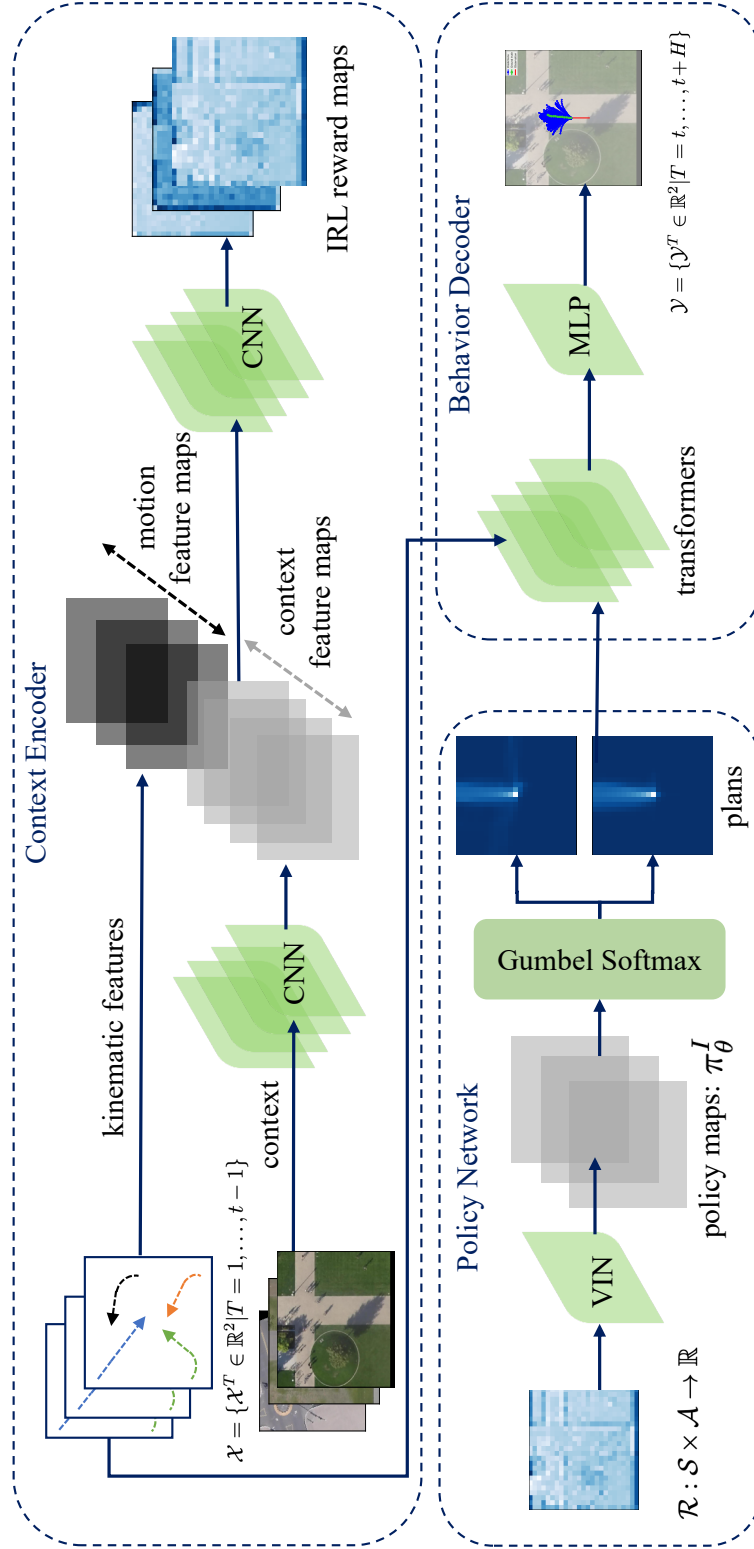


Figure 8.2: An overview of CAPHA with three primary modules: (1) The context encoder to embed the motion and context features and learn the IRL rewards; (2) The policy network to compute the policy maps through the value iteration network; (3) The behavior decoder module, including stacked transformers and MLP to forecast the distribution of SAs' future motions.

8.3.1 Context Encoder

The first component of CAPHA is a context encoder in which we encode the context features from the bird’s-eye-view images (B) of the SAs’ surroundings and concatenate them with motion features from SAs’ behavior history to obtain the reward maps. First, we adopt a few blocks of CNN to embed the context feature maps from B centered at SAs’ locations:

$$\beta = CNN(B) \quad (8.4)$$

where the dimensions of feature map β equal those of MDP grid representing state space S , i.e, 25×25 . Moreover, to consider kinematic features in the IRL rewards, we employ the behavior history of the SAs. This is necessary since the rewards are contingent upon not only the SA surrounding but also its previous motions. To do so, we leverage the SA’s velocity and grid cell positions obtained from the agent behavior history. The positions are extracted from the SA-centered frame coordinates in which the SA’s current position is the origin. We denote the kinematic features as follows:

$$\alpha(x, y) = [v, x, y] \quad (8.5)$$

where v represents the SA velocity.

Once the kinematic and scene feature maps are concatenated, we map them into IRL behavior and goal rewards. A CNN architecture is adopted to compute the behavior $\mathcal{R}_{\theta b(s,a)}$ and goal $\mathcal{R}_{\theta g(s,a)}$ rewards at each step leveraging the feature maps.

$$\mathcal{R}_{\theta}(s, a) = CNN(\alpha, \beta) \quad (8.6)$$

While the behavior rewards $\mathcal{R}_{\theta b(s,a)}$ are used to choose an action a , the goal rewards $\mathcal{R}_{\theta g(s,a)}$ are employed to mark the actions ending. As explained in Section 8.1.2, our purpose is to compute reward maps that maximize the likelihood of the expert behavior, which is differentiable w.r.t. θ [236]. According to Eq. 8.2, we will have:

$$\frac{d\mathcal{L}_{\theta}}{d\theta} = \sum_{e \in \mathcal{E}} (\mu - \mathbb{E}) \frac{d\mathcal{R}_{\theta}}{d\theta} \quad (8.7)$$

where μ , $\mathbb{E}$, and θ are the State Visitation Frequencies (SVF) from expert behavior, the expected SVF from the rewards, and the reward weights, respectively.

8.3.2 Policy Network

The second module of CAPHA is a policy network where we intend to produce an optimal policy $\pi_\theta^I(s, a)$ leveraging the reward maps gained in the previous module, and then sample this policy via Gumbel Softmax technique to generate potential plans for the SAs future motion which can be terminated at various goals, addressing the multimodality of the prediction task. The inputs of the policy network are the obtained state rewards conditioned on the kinematic and context features for the SA, and the output will be the sampled plans from the learned policy.

Given the computed reward maps $\mathcal{R}_{\theta p(s,a)}$ and $\mathcal{R}_{\theta g(s,a)}$, we leverage a value iteration to generate a policy at each step. Algorithm 2 describes the policy learning process in detail, where we solve the IRL problem for the policy via adopting the VIN [236] in ME-IRL formulation to improve the performance. The policy π_θ indicates the probability of carrying out an action within a known state which can be either stationary or, in our case, dependent on the MDP time steps shown as $\pi_\theta^I(s, a)$. According to Algo. 2 (lines 4 and 5), we recursively update the log partition values $Q^i(s, a)$ and V_s^i , which can be explained as the approximation of non-stationary rewards based on a state or a state-action group. The V_s is initialized with a large negative number to guarantee that MDP terminates at any goal state in the finite number of stages (line 1). At every stage, the policy is computed as follows:

$$\pi_\theta^i(s, a) = \exp(Q^i(s, a) - V_s^i) \quad (8.8)$$

Once the policy $\pi_\theta^i(s, a)$ is generated, we employ Algo. 3 to compute expected SVFs $\mathbb{E}$. As illustrated in line 4, policy propagation includes iterative application of optimal policy $\pi_\theta^i(s, a)$ to generate SVFs beginning with the start state, i.e., the behavior state at the grid center. The calculated SVFs are further used while training the rewards maps. The overall process is illustrated in Algo. 4 by which we maximize the likelihood $\mathcal{L}_\theta$ shown in Eq. 8.2 taking advantage of gradients calculated by Eq. 8.7 $\mathcal{F}$, and BP represents the neural network used to obtain the reward maps and back-propagation process.

In our proposed model, CAPHA, we produce trajectories conditioned on samples from $\pi_\theta^i(s, a)$ roll-out, namely plans. Consequently, as a final step in the policy network module, we need to sample the policy to address context-aware prediction. While random sampling may seem a straightforward approach, it makes the loss back-propagation process unstable [338]. Therefore, instead of random sampling, we benefit from the Gumbel

Algorithm 2: Approximate value iteration network

Input: $\mathcal{R}_{\theta b}^I(s, a), \mathcal{R}_{\theta g}^I(s, a)$: reward functions**Output:** $\pi_\theta^I(s, a)$: optimal policy

```

1  $V_s^I = -\infty, \forall s \in \text{behavior states};$ 
2 for  $i \in \{I, \dots, 2, 1\}$  do
3    $V_s^i = \mathcal{R}_{\theta g}^I(s, a), \forall s \in \text{goal states};$ 
4    $Q^i(s, a) = \mathcal{R}_{\theta b}^I(s, a) + V_{s'}^i = \mathcal{T}(s, a), \forall a \in \mathcal{A},$ 
    $\forall s \in \text{behavior states};$ 
5    $V_s^{i-1} = \log \sum_{a \in \mathcal{A}} \exp(Q^i(s, a)), \forall s \in \text{beh. states};$ 
6    $\pi_\theta^i(s, a) = \exp(Q^i(s, a) - V_s^i)$ 
end
return:  $\pi_\theta^I(s, a)$ 
```

Algorithm 3: Policy propagation

Input: $\pi_\theta(s, a)$, and s_{st} : start state**Output:** $\mathbb{E}$: Expected state visitations

```

1  $\mathbb{E}_s^1 = 0, \forall s \in \text{states};$ 
2  $\mathbb{E}_{s_{st}}^1 = 1;$ 
3 for  $i \in \{1, 2, \dots, I\}$  do
4    $\mathbb{E}_s^{i+1} = \sum_{a, s'} \pi_\theta^i(s', a) \mathbb{E}_{s'}^i, \mathcal{T}(s', a) = s$ 
end
5  $\mathbb{E}_s = \sum_i \mathbb{E}_s^i;$ 
return:  $\mathbb{E}_s$ 
```

Softmax approach in the sampling process, leading to continuous values. We formulate generating N -dimensional samples as follows:

$$g_k = \frac{\exp((\log(a_k) + d_k)/p)}{\sum_{j \in k} \exp((\log(a_j) + d_j)/p)}, \text{ for } k = 1, \dots, N \quad (8.9)$$

where a_k is the action probability from π_θ , d_k is an i.i.d sample, and p is a scaling factor. Considering the optimal policy $\pi_\theta^i(s, a)$, we denote the k^{th} plan sampled from it as $s^k = \{s_1^k, s_2^k, \dots, s_I^k\}$ which is expected to terminate at various goal states and discover a variety of ways to reach these goals. Furthermore, the samples are expected to be scene-compliant and represent the SA's decision process.

Algorithm 4: Maximum entropy IRL

Input: $\mathcal{E}, \mathcal{S}, \mathcal{A}, \mathcal{T}$ **Output:** θ : optimal weights

```

1  $\theta^1$  = weight initialization;
2 for  $i \in \{1, 2, \dots, I\}$  do
3    $E_{batch} \subset E$ ;
4    $R_\theta^i(s, a) = \mathcal{F}(s, \theta^i), \forall s \in \text{states}$ ;
5    $\pi_\theta^i(s, a) = \text{Approximate-VIN}(R_\theta^i(s, a))$ ;
6    $\mathbb{E}^i = \text{Policy-propagation}(\pi_\theta^i(s, a))$ ;
7    $\frac{d\mathcal{L}_\theta}{dR_\theta} = \mu - \mathbb{E}^i$ ;
8    $\theta^{i+1} = BP(\mathcal{F}, \theta^i, \frac{d\mathcal{L}_\theta}{dR_\theta})$ 
end
return:  $\theta$ 

```

8.3.3 Behavior Decoder

The final module of CAPHA is a behavior decoder in which we take advantage of the sampled plans from the policy network as well as the behavior history of SA from the context encoder to generate the future behaviors.

We further benefit from stacked transformers [339] to encode the plan and motion features. A transformer is an attention-based approach which was introduced to simplify recurrent structures. Transformers have shown considerable improvements in various domains, such as machine translation [333], by employing self-attention mechanisms and learning high-quality latent representations. Consider the encodings as $\{m^t\}_{t=1}^T$, queries as $\mathcal{Q} = \mathcal{W}_{\mathcal{Q}}(\{m^t\}_{t=1}^T)$, keys as $\mathcal{K} = \mathcal{W}_{\mathcal{K}}(\{m^t\}_{t=1}^T)$, and values as $\mathcal{J} = \mathcal{W}_{\mathcal{J}}(\{m^t\}_{t=1}^T)$. Then we will have:

$$\text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{J}) = \text{softmax}\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{d_k}}\right)\mathcal{J} \quad (8.10)$$

where $\sqrt{d_k}$ is a scaling factor. Afterward, consider h as attention heads. The MHA is calculated as below:

$$\text{MHA}(\mathcal{Q}, \mathcal{K}, \mathcal{J}) = \mathcal{W}_{\text{concat}}(\{\text{head}_n\}_{n=1}^h) \quad (8.11)$$

$$\text{where } \text{head}_n = \text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{J})_n \quad (8.12)$$

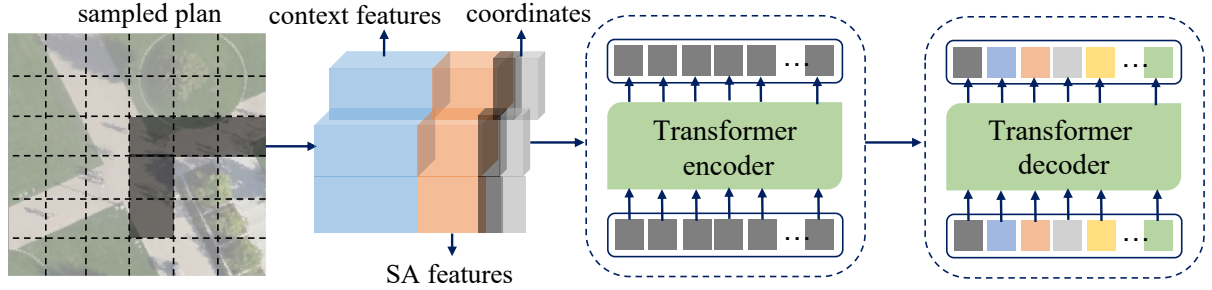


Figure 8.3: An illustration of the plan transformer. The local context, SA features at each state $s_i^k \in s^k$, and location coordinates (x, y) are encoded using the plan transformer.

Unlike approaches that employ transformer as interaction correlations extraction, we utilize a transformer-based structure to integrate and encode the plan and motion features.

Our behavior decoder is made up of two transformer units in a hierarchical structure: one for the motion features and another for the plan features. The latter is responsible for encoding the context features, the SA features, and the state coordinates from the plan. Given a sample plan S^k , we choose the local context features from the context maps which are the output β from the context encoder. As for the SAs, we add their velocity and acceleration to their corresponding grid cells. Finally, for each $s_i^k \in s^k$, we concatenate the context and SA features with state locations (x, y) as the input of the plan transformer. Figure 8.3 illustrates this process. The former transformer is responsible for encoding the behavior history of the agent $\mathcal{X}$. Its input is the observed trajectories of the agent as $\mathcal{X} = \{\mathcal{X}^T \in \mathbb{R}^2 | T = 1, \dots, t-1\}$, where $t-1$ represents the observation horizon.

Finally, to generate a behavior $\mathcal{Y}_{t:t+H}^k$ from the model, we feed the output of plan and motion transformer O to a 3-layer MLP structure. The MLP will produce $\mathcal{Y}_{t:t+H}^k$, i.e., the SA future positions over H prediction horizon:

$$\mathcal{Y}_{t:t+H}^k = MLP(O) \quad (8.13)$$

Previous studies have reported their results over K modes of behavior distribution. We also employ the same finite set of modes using K-means clustering for a fair comparison. The center of clusters are considered as the output of the model denoted as $[\mathcal{Y}_{t:t+H}^1, \mathcal{Y}_{t:t+H}^2, \dots, \mathcal{Y}_{t:t+H}^K]$. Based on these behaviors sampled from the obtained distribution, we propose to train the behavior decoder using the variety loss:

$$\mathcal{L}_v = \min_k \|\mathcal{Y}^{gt} - \mathcal{Y}^k\|_2, \quad k \in \{1, 2, \dots, k\} \quad (8.14)$$

where $\mathcal{Y} = \{y_t, y_{t+1}, \dots, y_{t+H}\}$ and $\mathcal{Y}^{gt}$ represent the ground truth behavior.

8.4 Evaluation Results

In this section, we explain how we evaluate the performance of our introduced approach, called CAPHA, and compare its performance with those of several recent benchmarks on two large motion forecasting datasets. We first discuss the experimental setting, including the implementation details, the employed datasets, metrics, and comparison models. Afterward, we report and analyze the obtained quantitative and qualitative results.

8.4.1 Setting

In this part, we highlight the implementation details of our proposed model. We then briefly discuss the datasets, metrics, and baseline approaches used for our performance evaluation.

Implementation Details

We augment the context images and all corresponding SA behaviors in an agent-centered world-frame, where the reference is its current location. The BEV images of the surrounding context, which are part of the input of the context encoder, are cropped to a 200×200 image around the reference. As for the context mapping, we use the first block of EfficientNet [340] followed by convolutions with a kernel size of 4. This will reduce the dimensions of context maps from 200 to 25 that is the size of the MDP grid. We use a grid of 25×25 over the scene. To obtain the rewards, we utilize convolutions with kernel size 1, log-sigmoid as activation, and Adam optimizer (10^{-4}) and the loss that is shown in Eq. 8.2 and 8.7. Regarding the behavior decoder, we follow the standard structure of the transformers [333] for the architecture of the plan and motion transformers encoder and decoder. There are three hidden layers and eight attention heads. We sample 100 plans and cluster them into $k = 20$ behaviors unless specified otherwise. Note that once the optimal policy is obtained, we can provide our model with indefinite sample behaviors, which can quickly become superfluous. Instead, to address multimodality and to provide

the behavior decoder module with a concise representation of the behavior distribution, we cluster the sampled plans obtained leveraging the IRL reward maps into k behaviors. Furthermore, all experiments are performed on an NVIDIA RTX A6000 GPU unless specified otherwise. We use grid search to optimize hyper-parameters.

Datasets

To assess CAPHA performance, we carry out experiments on two large-scale motion forecasting datasets specifically chosen to incorporate contextual information. Recognizing the importance of assessing the model’s performance in the presence of comprehensive context, these datasets include BEV images from the surrounding environment, enabling a robust evaluation of CAPHA’s capabilities in capturing and leveraging contextual cues for accurate motion forecasting.

The Intersection Drone Dataset (inD). This dataset [78] is made up of about 11 thousand trajectories captured with drones over four intersections in Germany. InD records ten hours of data for heterogeneous agents, including vehicles, pedestrians, and cyclists. InD provides the BEV images of the intersections as well as the scaling factor for pixel/meter conversions. It consists of both trajectory recordings and meta information (i.e., start frame numbers or types of agents), simplifying the preprocessing process. We follow the previous work on inD dataset [139, 226, 227] for obtaining validation split, and we reduce the frequency of trajectories by ten times. The original data is sampled by $fps = 25$. We also select 8 and 12 timesteps as the observation and prediction horizons, which indicate 3.2s and 4.8s of data, respectively.

The Stanford Drone Dataset (SDD). The SDD dataset [86] covers data for about 20 thousand distinct agents categorized into pedestrians, vehicles, and cyclists. The data is collected with a drone over 60 different environments on the Stanford campus, providing us with a large variety of scenarios, which is helpful to validate the generalization of CAPHA to different scenarios. The dataset offers the BEV images and the SA locations in pixel world reference. Similar to the inD dataset, we follow the previous work on SDD [134, 227, 266] to set the validation setting. We use 8 and 12 timesteps as the observation and prediction horizons, which represent 3.2 and 4.8 seconds using $fps = 2.5$.

Metric

Following previous studies on inD and SDD benchmarks, we use two sample-based metrics: (1) Average Displacement Error (ADE), and (2) Final Displacement Error (FDE) to

compare the prediction models' performance. These two metrics do not penalize feasible behaviors produced by prediction approaches that are not corresponding to the ground truth. The ADE calculates the average of l_2 norm between the computed behaviors by the models and the true behaviors for all agents:

$$ADE = \min_k \frac{1}{H} \sum_t^{t+H} \|\mathcal{Y}_t^{gt} - \mathcal{Y}_t^k\|_2 \quad (8.15)$$

where H is the prediction length/horizon, i.e., the number of timesteps into the future for which a prediction is made. Furthermore, the FDE metric computes the mean l_2 norm between the calculated behaviors and the true ones at a single time step $t + H$:

$$FDE = \min_k \|\mathcal{Y}_{t+h}^{gt} - \mathcal{Y}_{t+h}^k\|_2 \quad (8.16)$$

Additionally, to compare CAPHA with state-of-the-art methods on the Trajnet split of SDD dataset, we also report Off-road Rate (OR) metric which computes the number of predictions that are not scene-compliant following the definition provided in work [143].

Baseline Models

We compare CAPHA's performance with several benchmarks on inD and SDD datasets which are discussed earlier. In the following, we briefly discuss the methodology of each used model:

InD dataset. We select the following state-of-the-art methods as the baselines for various experiments on inD dataset:

- Social-GAN [134] implicitly models various behavior modes as latent variables where the emphasis is on understanding the intents of the SAs and produces socially acceptable behaviors by GANs.
- Social-LSTM [233] follows an encoder-decoder scheme to generate the future behaviors, taking advantage of social pooling between each LSTM layer.
- AMENet [227] models the SA behaviors and learns their interactions, taking advantage of latent variables and CVAE-based architecture.
- DCENet [139] builds upon the AMENet with the difference that it uses an MHA to learn social interactions, while the former uses two separate MHA modules for spatial modeling of interactions and temporal one.

- Grip++ [237] represents the SAs' social interactions using graphs and encodes features with blocks of GCN. It produces behaviors using an RNN-based encoder-decoder.
- SCOUT [226] employs graph attention to model the interaction between the SAs, where the vertices are the agents and the connections represent their interactions with the attention-based weights based on their distance.
- STAG uses graph attention and MHA to model the agents' interactions. It uses temporal convolutions to generate the future behavior of the agents.
- Y-net [240] employs rasterized BEV images to estimate the agents' goals represented by a heatmap which are further fed to the behavior decoder.
- Guo *et al.* [143] uses a CNN and an LSTM to encode the context and history features. These features and the predicted Occupancy Grid Maps (OGM)s are fed to an IRL-based network to learn the behaviors' distribution.
- MTP-GO [255] models the agents' interactions using graph GRU components. It address multimodality by learning an MDN.
- PECNet [138] implicitly models behavior modes by a CVAE. It also builds a social mask using social pooling to model interactions among agents.
- StarNet [260] utilizes temporal convolutions to extract trajectory history and models agents' interactions using graph attentions and MHA.

SDD dataset. We compare our model's performance with several benchmarks that have presented results on all or part (i.e., Trajnet split) of SDD.

- MATF [243] creates a tensor that uses two parallel structures to encode behavior history by an LSTM and context images by a CNN. It utilizes GANs to generate different modes of behaviors to address multimodality.
- SoPhie [264] also benefits from GANs to generate a variety of behavior modes. It encodes the context scene with a CNN and behavior history of the agents using an LSTM encoder. SoPhie also employs several MHA modules to give more credit to the scene parts/agents with more effects on the target agent.

- CF-VAE [245] computes the prior distribution by conditioning it on context images and SA’s behavior history encoded using LSTM and CNN structures.
- P2T [144] encodes the context using a CNN-based structure and learns the reward maps using IRL. The behavior history of the agents is encoded via a GRU-based encoder. Similar to our model, P2T addresses multimodality by conditioning the behaviors on the plans.
- SimAug [242] uses 3D augmented simulated images for better generalization during validation of unseen context. It generates behavior modes by beam search.
- SSAGCN [238] encodes the context using a CNN and attention mechanism which are further combined with agents representations extracted from graph attentions to construct a global graph. It uses temporal convolutions to generate future behaviors.
- Goal-SAR [239] is a goal-based approach that benefits from attention to estimate the goals conditioned on extracted features from rasterized BEV images.
- Multipath [241] encode context features with CNN and generates the future behaviors by estimating their probabilities on clustered behavior modes.
- RCPNet [259] takes advantage of a bidirectional DVAE network to estimate future behaviors distributions. Its encoder uses recurrent units.
- MID [252] utilizes temporal-social structure to encode the trajectory history and agents’ interactions. It produces the final trajectories using transformers.

8.4.2 Results

This section presents our obtained behavior prediction results using the experimental setting discussed in Section 8.4.1. We report and analyze the evaluation results on inD and SDD datasets which are also explained in Section 8.4.1.

Quantitative Analysis

InD Dataset. We compare the performance of our introduced model, CAPHA, with those of recent state-of-the-art on inD dataset according to ADE and FDE metrics.

The CAPHA prediction results are summarized using an 8-step observation and a 12-step prediction horizon for all 4 intersections in the inD dataset. As demonstrated in Table 8.1, CAPHA achieves state-of-the-art results for this dataset, i.e., the lowest ADE and FDE error values for all intersections. For instance, let us consider the third intersection, Frankenburg. The results indicate that CAPHA decrease the errors by 43% and 30% w.r.t. ADE and FDE metrics compared with the SCOUT [226] which reports the following best performance. Furthermore, we observe that the CVAE-based methods such as AMENet and DCENet achieve better error values than pooling-based approaches like Social-LSTM, showing not only the significance of addressing multimodality within the forecasting task but also the effectiveness of conditioning various behavior modes on latent variables. Scout improves their effectiveness by integrating the social interactions between the agents via graph attention. STAG improves the overall accuracy by merely considering asymmetric social interactions while making representing graphs. However, these two approaches still fail to generate scene-compliant trajectories. We address this problem by using ME-IRL and conditioning the results on the plans sampled from optimal policy, achieving the average ADE of 0.29 and FDE of 0.63, reducing the errors by 31% and 28% compared to the STAG.

Table 8.2 summarizes the calculated ADE and FDE error values over different scenarios in the inD dataset categorized by the type of the SAs so that we can analyze the impact of agent groups on the robustness of our model. This is important as different agent groups demonstrate unique behaviors. As can be observed, the results indicate that CAPHA achieve lower error values for cars and pedestrians groups. For instance, let us consider car agent group. SCOUT generates the average errors of 0.41 and 0.92 w.r.t. ADE and FDE metrics, respectively, while the same values for our model, CAPHA, are 0.25 and 0.61. This indicates an improvement of 39% and 34%. Moreover, we also report the error values on different agent groups over all scenarios on inD dataset. Table 8.3 summarizes ADE and FDE error values. As demonstrated, CAPHA produces better results for vehicles and pedestrians than several social-aware methods such as MTP-GO and STAG. For instance, let us consider pedestrians. As can be seen, our method, CAPHA, reduces errors by 29% and 9%, w.r.t. ADE and FDE metrics compared with the STAG, which reports the following best performance, demonstrating the effectiveness of including scene encodings as well as social interactions among nearby agents.

To explore long-term prediction, we conduct more experiments in which we follow the previous work [240] settings over inD dataset (i.e., the observation and prediction horizons of 5s and 30s, respectively). We demonstrate the results for $k = 20$ for all comparison

Table 8.1: Comparison results (ADE / FDE) over inD dataset for observation=8 and horizon=12.

Intersections	Social-LSTM [233]	Social-GAN [134]	Grip++ [237]	DCENet [139]	AMENet [227]	SCOUT [226]	STAG	CAPHA
A (Strasse)	2.29 / 5.33	3.02 / 5.30	1.65 / 3.65	0.96 / 2.12	1.07 / 2.22	0.67 / 1.55	0.51 / 1.30	0.56 / 1.11
B (Bendplatz)	1.28 / 3.19	1.55 / 3.23	0.94 / 2.06	0.64 / 1.41	0.65 / 1.46	0.48 / 1.08	0.37 / 0.66	0.25 / 0.54
C (Frankenburg)	1.78 / 4.24	2.22 / 4.45	0.59 / 1.41	0.86 / 1.93	0.83 / 1.87	0.30 / 0.69	0.49 / 1.06	0.17 / 0.48
D (Heckstrasse)	2.17 / 5.11	2.71 / 5.64	1.94 / 4.46	0.28 / 0.62	0.37 / 0.80	0.40 / 0.83	0.30 / 0.48	0.19 / 0.39
Average	1.88 / 4.47	2.38 / 4.66	1.28 / 2.88	0.69 / 1.52	0.73 / 1.59	0.46 / 1.03	0.42 / 0.88	0.29 / 0.63

Table 8.2: ADE / FDE results over inD dataset categorized by agents types and interactions.

Agents	SCOUT [226]				CAPHA			
	A	B	C	D	A	B	C	D
Cars	0.67 / 1.54	0.41 / 0.91	0.18 / 0.41	0.39 / 0.80	0.51 / 1.32	0.28 / 0.55	0.09 / 0.28	0.10 / 0.29
Pedestrians	0.50 / 1.12	0.54 / 1.21	0.52 / 1.15	0.41 / 0.84	0.44 / 0.63	0.32 / 0.61	0.16 / 0.34	0.11 / 0.31
Cyclists	0.29 / 0.65	0.46 / 1.01	0.51 / 1.14	0.23 / 0.48	2.71 / 5.02	1.05 / 2.20	0.74 / 1.56	0.53 / 1.88
All	0.67 / 1.55	0.48 / 1.08	0.30 / 0.69	0.40 / 0.83	0.56 / 1.11	0.25 / 0.54	0.17 / 0.48	0.19 / 0.39

Table 8.3: ADE / FDE over inD dataset per agent type.

Models	Cars	Pedestrians	Cyclists
MTP-GO [255]	1.07 / 3.32	0.38 / 1.03	0.90 / 2.65
SCOUT [226]	0.36 / 0.80	0.52 / 1.16	0.47 / 1.06
STAG	0.35 / 0.69	0.38 / 0.90	1.28 / 2.84
Guo <i>et al.</i> [143]	-	1.97 / 2.98	-
StarNet [260]	0.32 / 0.89	-	-
CAPHA	0.28 / 0.61	0.27 / 0.82	1.25 / 2.65

baselines in Table 8.4. Our results has revealed that CAPHA outperforms all other benchmarks, achieving the lowest error of 11.09% ADE and 18.78% FDE. Compared to baselines which also use rasterized BEV images [138, 143, 240], CAPHA improves the results by 12% for ADE and 3% for FDE.

SDD dataset. We evaluate the performance of CAPHA on the SDD dataset (all) and provide a comparison between its performance and that of eight benchmarks on this dataset as described in Section 8.4.1. We summarize the experimental results in Table 8.5, where the observation and prediction horizons are 3.2s and 4.8s, respectively. All of our comparison approaches consider various possible behavior modes and produce multimodal results. Table 8.5 reports the validation metrics for $k = 20$ for all models. CAPHA demonstrates a better performance w.r.t both metrics compared with state-of-the-art benchmarks on this dataset. MATF outperforms pooling-based models like Social-GAN by generating a tensor for spatial reasoning at several scales. SoPhie enhances the performance by using both social and spatial attention while encoding the scenes and the SA’s behavior history. P2T improves FDE by conditioning behaviors on sampled plans from IRL. Exploring simulated and augmented data, SimAug obtains lower errors. Finally, CAPHA outperforms the above models by explicitly conditioning the future behavior on SVFs from optimal policy and using transformers to incorporate scene and motion information. Our results reveal that compared with the best previous benchmark, CAPHA reduces the error values by 11% and 22% according to ADE and FDE metrics, respectively.

Table 8.6 summarizes the behavior prediction results on a subset of the SDD dataset (Trajnet split) which comprises mostly pedestrians. All of our comparison methods take

Table 8.4: Long-term behavior prediction over inD dataset (pedestrians agents and the split used in Y-net.)

Models	ADE	FDE
Social-GAN [134]	38.57	84.61
PECNet [138]	20.25	32.95
Guo <i>et al.</i> [143]	13.09	19.39
Y-net [240]	12.67	21.13
CAPHA	11.09	18.78

account of several possible behavior modes and address multimodality. CAPHA achieves lower errors compared with seven recent baselines w.r.t ADE and FDE and outperforms all methods in terms of OR metric. We observe that models considering both social interactions and scene encodings, such as Y-net, Goal-SAR, or CAPHA, produce lower errors compared to models which merely use social interactions, such as RCPNet. Similar to CAPHA, Guo *et al.* [143] conditions future behaviors on plans obtained from IRL and OGM prediction, suggesting the effectiveness of modeling the surrounding environment to produce scene-compliant behaviors. Compared to their model, CAPHA obtains competitive ADE and FDE error values and improves OR metric by 36%.

Ablation Analysis

Modules. As discussed in Section 8.3, our approach is made up of different modules to perform the prediction task. We carry out an ablation experiment with different components of CAPHA: First, we examine the effect of learning the reward maps on the performance of the model. Instead of taking advantage of ME-IRL to learn the reward values, we use behavior cloning, where we estimate a policy which implicitly maps the context and kinematic encodings into probabilities. We call this ablation variant CAPHA-v2. Second, we investigate the effect of our sampling component, where we employ Gumbel softmax technique to extract samples from a policy. In CAPHA-v3, we remove this process and directly sample the optimal policy. Finally, we examine the impact of considering social interactions among the SAs. In this version (CAPHA-v1), we only utilize the SA motion features in the plan transformer. We summarize the results

Table 8.5: ADE / FDE results over SDD dataset (all).

Models	ADE	FDE
Social-LSTM [233]	31.19	56.97
Multipath [241]	28.32	58.38
Social-GAN [134]	27.25	41.44
MATF [243]	22.59	33.53
SoPhie [264]	16.27	29.38
CF-VAE [245]	12.60	22.30
P2T [144]	10.97	18.40
SimAug [242]	10.27	19.71
CAPHA	9.13	14.34

Table 8.6: Results on Trajnet split of the SDD dataset.

Models	ADE	FDE	OR
SSAGCN [238]	10.36	11.80	-
PECNet [138]	9.96	15.88	0.071
P2T [144]	8.78	14.08	0.060
RCPNet [259]	8.18	13.83	-
Y-net [240]	7.85	11.85	0.048
Goal-SAR [239]	7.75	11.83	-
MID [252]	7.61	14.30	-
Guo <i>et al.</i> [143]	6.77	10.46	0.066
CAPHA	7.67	11.41	0.042

Table 8.7: Ablation analysis over SDD dataset.

Our Model	ADE	FDE
CAPHA-v1	11.28	17.22
CAPHA-v2	10.36	16.01
CAPHA-v3	9.41	14.54
CAPHA	9.13	14.34

Table 8.8: Effect of feature map size over SDD dataset.

CAPHA	ADE	FDE	Inference time
Grid size = 20	8.92	13.34	56 ms / sample
Grid size = 25	9.13	14.34	57 ms / sample
Grid size = 33	9.48	14.37	59 ms / sample

of ablation experiments in Table 8.7. We observe that the complete version of our model achieves the best errors, indicating how different components of our model contribute to its overall performance. Comparing the reward-based results with the behavior cloning-based ones reveals the capability of ME-IRL in learning a policy and the transferability of the reward maps. Moreover, considering the social interaction among the SAs can improve the overall performance of the prediction model by 19% and 16.8% w.r.t. ADE and FDE, respectively.

Context encoder. We assess the impact of feature map size, i.e., β in our context encoder module, on the overall performance of our proposed model. Note that as mentioned in Section 8.3.1, the dimensions of the feature map β equal those of the MDP grid representing state space. Table 8.8 summarizes the experimental results over the SDD dataset (all) with a prediction horizon of 4.8 seconds. We experiment with three kernel sizes, including 3, 4, and 5, leading to feature map sizes of 33, 25, and 20, respectively. We observe that changing the feature map size within the context encoder does not result in a substantial change in the performance of CAPHA. The larger the feature map size and the MDP grid size, the longer the inference time since it takes longer to learn the policy. Moreover, a smaller grid size of 20 reduces the error values by 2% and 7% according to ADE and FDE.

Table 8.9: Inference and training time comparison results. Training time is in s/epoch for all modules of the methods.

		Guo <i>et al.</i> [143]		CAPHA	
Parameters		854040		96735	
Inference time		69 ms / sample		57 ms / sample	
Training time	OGM	770 s	-	-	
	Policy	1973 s	Policy	178 s	
	Refinement	1111 s	Decoder	148 s	
	Fine-tun.	2129 s	Fine-tun.	595 s	

Computation Time Analysis

To analyze the performance of CAPHA, we compare its number of trainable parameters, inference time, and training time with those of a similar methodology [143] which benefits from IRL and trains a policy network to learn the distribution of the future behavior of pedestrians. This comparison is conducted on the SDD dataset using an NVIDIA H100 GPU. We summarize the results of computation time analysis in Table 8.9. Note that we report training time in seconds per epoch. The results demonstrated are the average from ten epochs for different modules of both models. As shown, CAPHA is more efficient, having a lower number of parameters and computational cost. For instance, it reduces inference time by 17% per sample.

Qualitative Analysis

We carry out visualization analysis on different modules of the CAPHA and demonstrate several results examples in Fig. 8.4 chosen from a variety of context information. We present the input images for each SA (Fig. 8.4a1- 8.4e1), the heatmaps of goal SVFs (Fig. 8.4a2- 8.4e2), and behavior SVFs (Fig. 8.4a3- 8.4e3) from the learned IRL policy. Note that in the heatmaps, the lighter the color, the higher the values. As observed, the analysis reveals that the optimal policies explore feasible behaviors over the described grid (see Section 8.3.1) for the SA within different context scenarios, which leads to avoiding obstacles on its way and conforming to the surrounding scene. This suggests how CAPHA successfully generates an optimal policy $\pi_{\theta}^I(s, a)$ to be sampled later in order to address context-aware prediction. Next columns in Fig. 8.4 represent the predicted distribution

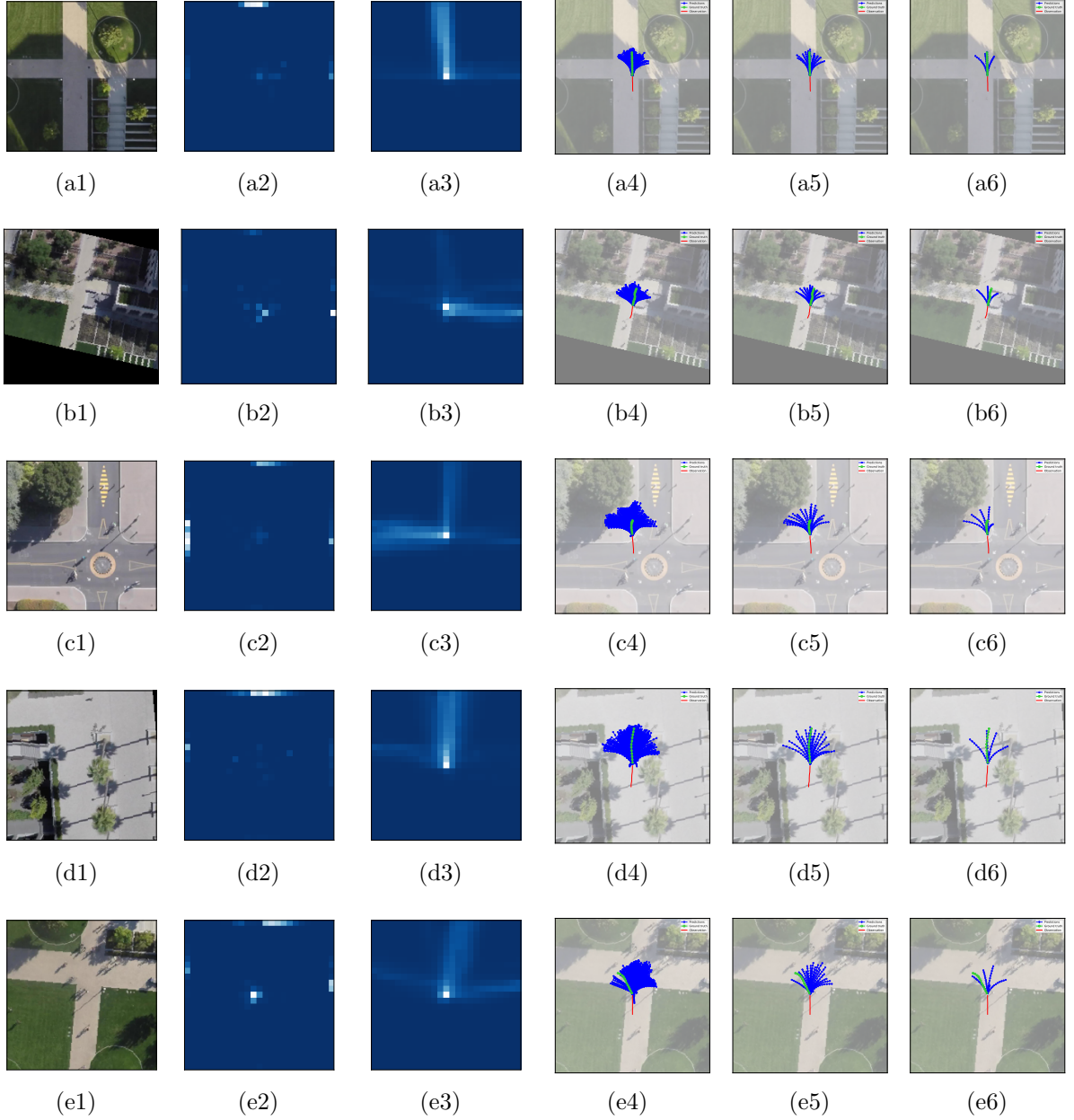


Figure 8.4: Qualitative analysis of the forecast behaviors for the introduced CAPHA model over several trajectory scenes: the agent-centered BEV images (a1-e1), the goals (a2-e2), the behavior SVFs (a3-e3), the predicted distribution (a4-e4), the multimodal outputs with $k=20$ (a5-e5), and the multimodal outputs with $k=5$ (a6-e6). Observations are shown with red lines. The predicted and real behaviors are represented as blue/star and green/circle lines, respectively.

(Fig. 8.4a4- 8.4e4), the multimodal outputs for $k = 20$ (Fig. 8.4a5- 8.4e5), and $k = 5$ (Fig. 8.4a6- 8.4e6), respectively, where solid lines show behavior observations, and stars and circles ones indicate the forecast behaviors and ground truth, respectively. As observed, CAPHA accurately forecasts the future behavior of SAs in different scenarios and produces several behavior modes. Note that the SVFs extracted from the policy shown in the second and third columns are also multimodal.

8.5 Concluding Remarks

In this chapter, we have introduced an interpretable behavior prediction approach, namely CAPHA, that benefits from maximum entropy IRL to condition the predicted motions of the heterogeneous agents on the environmental information. Hence, our proposed model, CAPHA, could generate behaviors that conform to the surrounding scenes, which is essential for the real-world deployment of automated vehicles. Unlike several previous studies that addressed multimodality by averaging the potential behavior modes or implicitly modeling various behavior modes as latent variables, CAPHA has produced multimodal behaviors by conditioning the SAs' future behaviors on the plans sampled from IRL policies. We have evaluated the performance of CAPHA over two large-scale behavior prediction datasets consisting of heterogeneous agents. And, we have arrived at the following salient points based on the observed results: Quantitative assessments have demonstrated that CAPHA could effectively estimate the distribution of future behavior of SAs for various groups of agents, including vehicles, cyclists, and pedestrians, and achieved state-of-the-art results when compared to the most recent benchmarks according to ADE and FDE metrics. Furthermore, the results on the SDD dataset with more than 60 distinct environmental contexts have revealed the transferability strength of CAPHA, which lies in its reward-based structure. Qualitative evaluation has also confirmed how the introduced model generalizes over unknown contexts. Furthermore, the computational cost analysis has revealed that our model has an inference time of 57ms per sample, making CAPHA suitable for real-time deployment. Nevertheless, this efficiency can be achieved when prior BEV images of the environment and behavior history of the agents are available, which is a challenge for the behavior prediction component of automated vehicles.

Chapter 9

GMP: Goal-Based Multimodal Motion Prediction for AVs

As discussed in Chapter 8, anticipating the future behavior of surrounding agents for an autonomous vehicle requires considering the inherent uncertainty in agents’ behavior, making behavior prediction challenging. To tackle this issue, in this chapter, we propose conditioning the agents’ future motions on both context information and potential multimodal goals. We design a novel Goal-based Motion Prediction (GMP) framework that anticipates the distribution of TAs final goals on which their potential future behaviors are conditioned. We argue that estimating the endpoint distributions while taking account of lane and environmental interdependencies and considering them as behavior conditions can benefit the forecasting model to generate more diverse predictions that also conform to the scene and map layouts. We propose combining contextual and kinematic encodings using graph-based message passing, which focuses on local traversability and long-term dependencies. In brief, our main contributions are summarized as follows:

- We propose GMP, a novel goal-based approach for multimodal motion forecasting which anticipates the distribution of the potential endpoint areas as conditions for trajectory prediction. As a result, GMP considers the inherent uncertainty in surrounding vehicles’ behavior.
- We construct residual blocks of temporal convolutions with skip connections to incorporate the interactions among the target vehicle and surrounding agents. We adopt graph structure to represent scene context and extract local topological features using graph attention. Consequently, GMP considers the interactions and interdependences between various agents as well as the driving environment.

We organize the remainder of this chapter as follows. Section 9.1 presents our problem formulation and the main structure of GMP. Section 9.2 analyzes the performance of GMP and provides comparison results with several recent benchmarks. Section 9.3 summaries obtained results and improvement direction.

9.1 Proposed Model

This section describes behavior prediction formulation and the GMP framework.

9.1.1 Formulation

Given the previous behavior sequences of a TA $H = \{(x, y)_1^h, (x, y)_2^h, \dots, (x, y)_t^h\}$ and its nearby agents $N = \{(x, y)_1^n, (x, y)_2^n, \dots, (x, y)_t^n\}$ where t is the observation length as well as the context information from HD maps denoted as $\mathcal{M}$, behavior prediction aims at precisely generating the TA's future locations up to a fixed horizon T , $F = \{(x, y)_{t+1}, (x, y)_{t+2}, \dots, (x, y)_{t+T}\}$. We intend to obtain the probabilistic distribution of the future behaviors $P(F|\mathcal{M}, H, N)$. The output of GMP is a multimodal set of behaviors for the TA at each scenario $F_o = \{F_k\}_{k \in [1, K]} = \{(F_{t+1}^k, F_{t+2}^k, \dots, F_{t+T}^k)\}_{k \in [1, K]}$ where K is the number of prediction modes. Consequently, the distribution can be estimated using Eq. 9.1.

$$P(F|\mathcal{M}, H, N) \approx \sum_{g \in \mathcal{G}} P(g|\mathcal{M}, H, N) P(F|g, \mathcal{M}, H, N) \quad (9.1)$$

where $\mathcal{G}$ is the set of potential goals, $P(g|\mathcal{M}, H, N)$ is for capturing candidate goals depending on environmental information, and $P(F|g, \mathcal{M}, H, N)$ is for generating final behaviors conditioned on the goals and environmental information. Therefore, we divide the problem into two subtasks, including how to effectively find the representative goals and precisely approximate future behaviors based on those goals.

9.1.2 Goal-based Motion Prediction

An overview of the presented GMP is demonstrated in Fig 9.1. We first represent the motion history of TA and nearby vehicles as well as HD map data as a set of vectorized entities which are then encoded. Different layers of semantic information are then hierarchically aggregated to explore the interaction and interdependencies between various entities, including agents, lanes, and Region-of-Interest (ROI)s. We next conduct

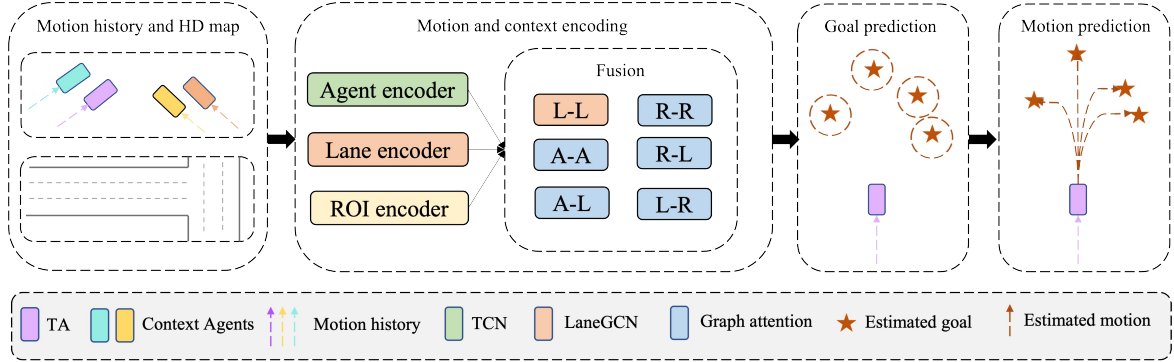


Figure 9.1: Illustration of GMP. I-I denotes the interactions where A, L, and R represent agents, lanes, and ROIs, respectively.

classification and use a decoder to estimate the final endpoints for each TA. Eventually, using a prediction header, we generate TA future behavior conditioned on these absolute endpoints. We will explain each module in the following.

9.1.3 Behavior History Encoding

As illustrated, the first step of GMP is to encode the motion history of the TA and all of its nearby vehicles as well as the scene. Regarding extracting the motion features, each past behavior is demonstrated as a set of positional displacement $\{(x, y)_{t_i} - (x, y)_{t_{i-1}}\}_{t_i=1}^{T+t}$ where $(x, y)_{t_i}$ is agents position at time step t_i . The agents' behaviors are padded with the last data entry in both forward and backward directions if the behavior length is smaller than the observation horizon. We also concatenate the heading information to the agents' behaviors.

We apply a temporal convolutional network to extract the behavior representations. More specifically, we adopt three layers of temporal blocks each of which contains consecutive residual blocks of dilated causal convolutions, Group normalization, ReLU activation, and dropout with output channels of 32, 64, and 128, respectively. Since behavior prediction attempts to anticipate the potential motions of TAs with a horizon of several seconds, agents with larger velocity may have traveled a larger distance. Using dilated convolutions (DC) will allow GMP to capture long-range interconnections for precise forecasting. These DCs leverage the dilation concept described as the number of time steps the convolution filter skips the input, boosting the receptive field, which makes looking to a broader area possible for convolutions. We obtain a DC operation as follows:

$$\mathcal{Y}_m(i) = (H *_{\mathcal{D}} f)_{(i)} \sum_{d=1}^D H_{(i-K_d)} f(K_d) \quad (9.2)$$

where f is the filter, D is the dilation values, and K_d is the d^{th} dilation. The final output will be a temporal mapping with a channel size of 128 for the TA and all of its nearby vehicles.

9.1.4 Context Encoding

To make prediction more realistic and scene compliant, we need to provide the model with geometric and topological semantics extracted from HD maps as well. Therefore, we construct two layers of structured graphs from map data, including one lane graph G_l and one ROI graph G_r . The former concentrates on local short-range traversability, while the latter focuses on topological connections between the lanes and the agents. We further describe the graph construction method for each layer below.

Lane Graph

The behavior of the TA and all of its nearby vehicles is highly contingent upon the topological features of the surrounding map. For instance, there is a slight chance that a TA maneuvers to the right when a right road segment does not exist. Therefore, we construct the lane graph G_l adopting LaneGCN backbone [132] to encode the map representations. In G_l the nodes are lane centerlines while the edges represent the geometric connection between the lanes, i.e., nearby lanes in all four directions, including forward f , backward b , right r , and left l . This way, the model has access to semantic environmental information for more effective behavior prediction. We then apply dilated graph convolutions (DGC) to extract the map encodings in this layer. We compute each DGC as follows:

$$\mathcal{Y}_l = NW_0 + \sum_{i \in \{l, r\}} A_i NW_i + \sum_{d=1}^D \sum_{j \in \{b, f\}} A_j^{K_d} NW_{j, K_d} \quad (9.3)$$

where A and W are the adjacency and weight matrix, respectively, and N is the node features. The output will be a semantic mapping with a size of 128.

ROI Graph

In scenarios where lane nodes do not have direct connectivity but seem geometrically near to one another, G_l may not fully represent the map information. Consequently, we construct another graph denoted as G_r to include ROIs for the TA and surrounding vehicles. This way, GMP also considers the reachable areas for various traffic participants. To construct G_r , we use ROI points, which are centered to the TA, as nodes and their surroundings as the node attributes. The occupancy information for the ROI is also taken into account if an agent is in the proximity of the ROI nodes at a current time step. We apply MLP to extract the ROI representations. More specifically, we use three layers of MLPs followed by max pooling, Group Normalization, and ReLU taking into account dilated connection. The output will be the ROI encodings with the same channel size as lane graphs.

9.1.5 Aggregation

Once encodings are extracted, we need to aggregate this information to take account of the interactions between different vehicles, lane nodes, and ROI nodes. In this work, we adopt graph attention [333] to obtain several types of interactions. More specifically, we construct one fusion module per each existing interaction, such as the one between different agents, agents to lanes, or ROIs to lanes, each of which transfer useful information about the driving context. We aggregate node attributes based on their significance. Consider the interaction between TA and its nearby agents. Given a TA node n_i , we update its attributes by aggregating the attributes from its context agents nodes $j \in \mathcal{N}(i)$:

$$n_{i'} = \sigma(\text{concat}(n_i, \sum_{j \in \mathcal{N}(i)} \alpha_{ij} n_j W_1) W_2) \quad (9.4)$$

where,

$$\alpha_{ij} = \frac{\exp(\sigma(\text{concat}(n_i W_t, n_j W_c) W_A))}{\sum_{j' \in \mathcal{N}(i)} \exp(\sigma(\text{concat}(n_i W_t, n_{j'} W_c) W_A))} \quad (9.5)$$

where α_{ij} is the attention scores representing the importance of vehicle j to i , W values are weight matrices, and σ is a stack of Group Normalization and LeakyReLU. The context agents n_j are queried based on their l_2 distance to the TA node n_i . The output of each module will have 128 as their channel size.

9.1.6 Goal Prediction

After obtaining the context and behavior encodings, we first calculate the probability distribution of the future goals. We apply classification and compute smooth l_1 loss between the best-anticipated endpoint and the actual goal of the TA, resulting in closer final endpoints to the real one. Furthermore, to address multimodality, we should consider the inherent uncertainty in surrounding agents' behavior, e.g., a driver may conduct different possible maneuvers. Consequently, we generate various headers to extract K behavior modes. Each header contains two parts, including one for the endpoints $\mathcal{G} = g^k_{k \in [1, K]}$ and one for their confidence values $\mathcal{V} = v^k_{k \in [1, K]}$, where g^k and v^k are the k -th anticipated endpoint position and its confidence value for the TA, respectively. We output the probabilities $\hat{\mathcal{H}}$ using true endpoint-centered Gaussians. We train the classification stage leveraging a penalty-reduced logistic regression with focal loss [258], having α and β as hyper-parameters:

$$\mathcal{L}_{cls} = -\frac{1}{B} \sum_b \begin{cases} (1 - \hat{\mathcal{H}}_b)^\alpha \log(\hat{\mathcal{H}}_b) & \text{if } \mathcal{H}_b = 1, \\ (1 - \mathcal{H}_b)^\beta (\hat{\mathcal{H}}_b)^\alpha \log(1 - \hat{\mathcal{H}}_b) & \text{otherwise} \end{cases} \quad (9.6)$$

We further explore for the mode $\hat{k}$ that has the lowest l_2 distance with true endpoint position. For the regression stage, we exploit smooth l_1 loss applied on all TAs:

$$\mathcal{L}_{reg-g} = \frac{1}{\tau} \sum_{i=1}^{\tau} \text{reg}(g_{\tau}^{\hat{k}}, g_{\tau}) \quad (9.7)$$

where τ is the total number of TAs.

9.1.7 Behavior Estimation

Once the encodings are aggregated, we construct a multimodal estimation module to generate the final behaviors of the TA. The behaviors are produced using stacked linear layers with skip connections conditioned on anticipated endpoints and behavior and context encodings. The scores are directly queried from the probability distribution of the node, which has the lowest distance to the true endpoint. We train this stage using smooth l_1 loss applied on all steps in the prediction horizon:

$$\mathcal{L}_{reg-m} = \frac{1}{\tau} \sum_{i=1}^{\tau} \sum_{t=1}^T \text{reg}(F_{\tau t}^{\hat{k}}, F_{\tau t}) \quad (9.8)$$

where T is the prediction horizon and $F_{\tau t}^{\hat{k}}$ is the anticipated location for τ -th TA of the mode $\hat{k}$ at time t .

As all the stages are differentiable, we can train GMP end-to-end with the weighted sum of all loss functions.

9.2 Experiments

9.2.1 Setting

Implementation Details. We train GMP for 64 epochs on two A10 GPUs using a batch size of 32 and Adam optimizer. The learning rate is set to 10^{-4} with decay to 10^{-5} after 30 epochs. Within the behavior encoder, the TCN is used with a kernel size of 3, dropout of 0.2, the output channels of 32, 64, 128, and dilation of 2^l at each layer l . We utilize the vehicles whose distance from the TA is less than 100 m . Furthermore, we employ grid search methodology to systematically select hyper-parameters. No data augmentation or ensemble methods are used.

Dataset and Metrics. We assess GMP on the large-scale Argoverse 1 Motion Forecasting dataset [80], which contains both agents’ trajectories and HD map data for a variety of challenging situations, including changing lanes, intersections, and turning without any overlaps. Given the necessity for detailed information on HD maps for creating lane and ROI graphs, we opt for a new dataset here, diverging from the datasets used in previous chapters, to thoroughly evaluate the efficacy of our model under the specific conditions and requirements posed by high-definition mapping. Argoverse 1 includes more than 300000 real-world driving scenarios each of which provides 5 seconds sequences (i.e., 2 seconds of observation and 3 seconds of future trajectories) sampled at 10 Hz for training and validation samples. For test samples, only the observation part (i.e., two first seconds) is available, and the future behavior of the TA is withheld for the challenge. Regarding the metrics, we follow previous studies and assess GMP on four standard instance-based metrics: (1) Minimum Average Displacement Error (minADE), which measures L2 norm between the best-forecasted behavior and TA’s ground truth, (2) Minimum Final Displacement Error (minFDE) which measures L2 norm between the final step of the best-forecasted behavior and TA’s ground truth, (3) MR which counts the number of prediction behaviors that fall outside the 2-meter distance from the TA’s ground truth at the final step, and (4) Brier minimum Final Displacement Error (brier-minFDE) which adds a probability-based penalty (i.e., $(1 - p)^2$) to minFDA.

Table 9.1: Evaluation results on the test set of Argoverse 1. The best results are shown in bold.

Methods	minADE ₁	minFDE ₁	MR ₁	minADE ₆	minFDE ₆	MR ₆	brier-minFDE ₆
PRIME [248]	1.910	3.821	0.586	1.218	1.558	0.115	2.097
LaneGCN [132]	1.706	3.779	0.591	0.868	1.364	0.163	2.058
mmTransformer [339]	1.774	4.003	0.844	0.837	1.339	0.154	2.033
GOHOME [257]	1.688	3.648	0.572	0.943	1.450	0.104	1.983
Multimodal Transformer [261]	1.735	3.900	0.602	0.837	1.290	0.143	1.939
FRM [126]	1.706	3.748	0.573	0.816	1.267	0.143	1.936
GMP (ours)	1.668	3.648	0.571	0.835	1.217	0.132	1.864

9.2.2 Results

Comparison with Benchmarks

We compare GMP with six benchmark behavior prediction methods as shown in Table 9.1 using the Argoverse 1 test in which Brier-minFDE₆ is the official ranking score, representing the effectiveness of the models in multimodal behavior and probability prediction. Our approach outperforms all comparison benchmarks in terms of this metric. Compared to transformer-based approaches [261, 339] and a model-based method [248], GMP performs better in terms of all seven metrics, indicating its power in multimodal goal estimation and behavior prediction. Our model also outperforms GOHOME [257] and FRM [126] in six of the metrics. GMP reduces minFDE by 16% compared to GOHOME, showing the accuracy of its probability distribution estimation. Furthermore, leveraging more semantic information, GMP significantly improves LaneGCN [132] on all metrics (e.g., about 10% on brier-minFDE₆). Furthermore, the lower errors for multimodal prediction, i.e., $K = 6$, indicates the significance of considering the uncertainty within agents' driving behavior.

Qualitative Analysis

We further analyze the forecasting results of GMP on the validation set for which the true behaviors of TAs are known. We visualize the estimated goals and behaviors for the target agent in different challenging scenarios for $K = 6$ modes. For visualization purposes, we do not show the surrounding agents. As shown, GMP can precisely predict the potential behavior of the TA when turning (e.g., Fig 9.2a, and 9.2j), lane changing (e.g., Fig. 9.2i), or going straight at an intersection (e.g., Fig. 9.2c). Note that Fig. 9.2e illustrates an example of a scenario in which, although the observed behavior of the TA does not explicitly indicate its future behavior, GMP predicts the right-turn maneuver. Moreover, we analyze the probability scores generated by GMP for estimated modes. As illustrated in Fig. 9.3a and 9.3b, GMP takes account of not only the direction, but also the velocity of the TA's future behavior while producing multimodal outputs. The model calculates a low probability for modes the TAs are less likely to take.

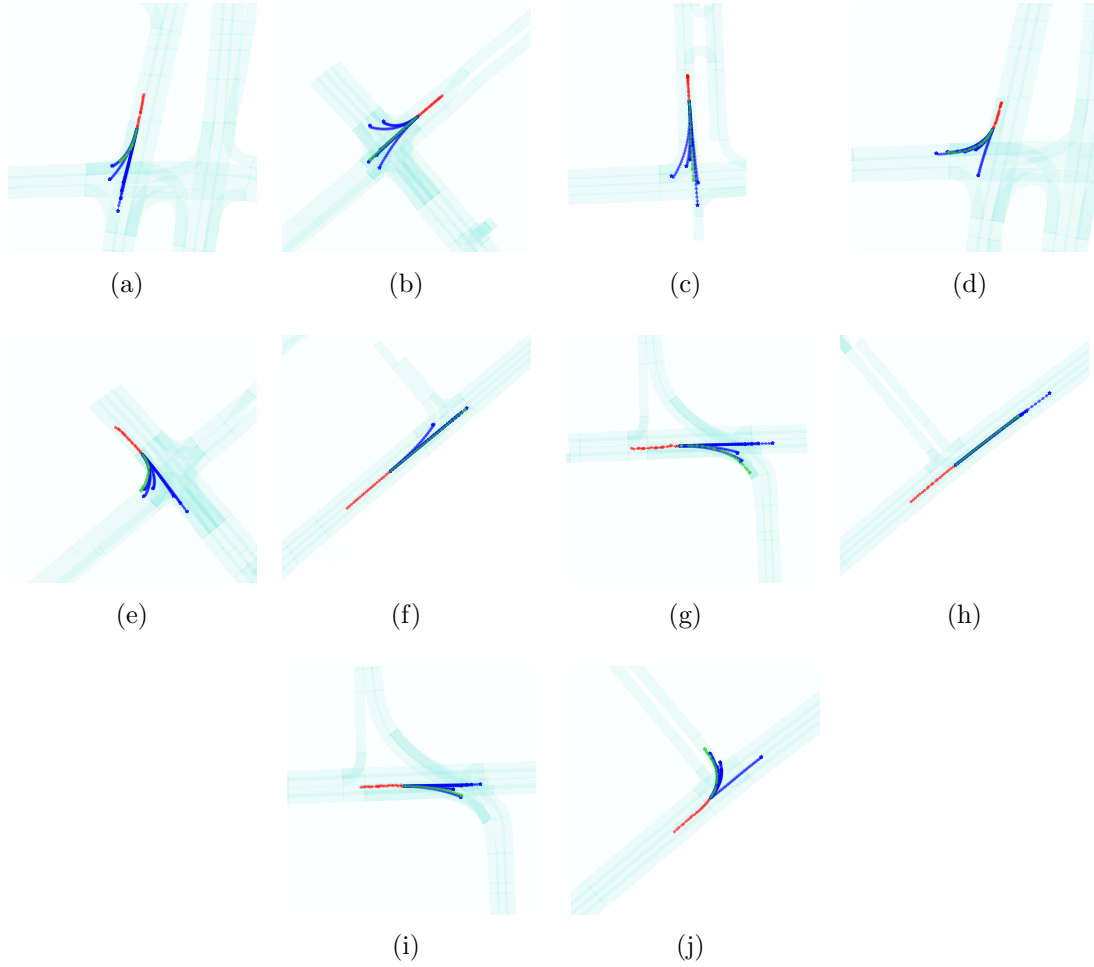


Figure 9.2: Qualitative analysis on Argoverse 1 for various scenarios. The behavior history and $K = 6$ anticipated behaviors are shown in red and blue, respectively. The TAs' true behaviors are shown in green. Stars represent the estimated final endpoint.

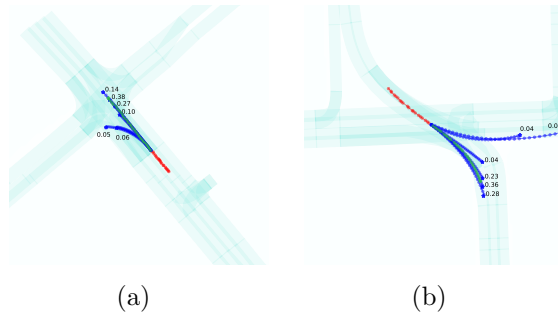


Figure 9.3: Probability scores of $K = 6$ estimated modes.

9.3 Concluding Remarks

In this chapter, we introduced a novel goal-based method for multimodal behavior prediction, namely GMP, which leveraged hierarchical graph representations to model the driving environment. GMP used temporal and graph convolutions to encode the context, and graph attention to model the interactions between various entities in the scene, leading to more accurate and feasible results. Experimental and visualization analysis confirms the strong ability of GMP in precise multimodal behavior prediction while preserving the feasibility of the behaviors.

Chapter 10

Conclusion and Future Directions

10.1 Contribution Summary

In this thesis, we have extensively explored driving behavior analysis and prediction, shedding light on its potential to enhance road safety, intelligent transportation systems, automated vehicles, and driving performance. We have designed a number of novel frameworks to enhance driver identification, driver verification, impostor detection, and agent behavior prediction.

Initially, we devised a framework modeling for driver identification. Through this framework, we critically assessed the performance of several deep learning and classical machine learning approaches for driver identification. The critical evaluation results highlighted the importance of capturing temporal dependencies within the driving time series data and the impact of the sliding window technique on model performance, offering guidance for designing new driver identification models.

Then, we proposed DriverRep, a novel method addressing challenges related to labeled data availability and scalability for driver identification. Different from related work, DriverRep efficiently extracted driver embeddings from data in an unsupervised manner, allowing for successful driver identification even in scenarios where extensive labeled data is lacking. The results demonstrated the potential of DriverRep in capturing intrinsic driving characteristics and outperforming state-of-the-art driver identification schemes.

Furthermore, we designed a novel deep learning-based architecture for driver identification, verification, and impostor detection that excelled in capturing distinguishing characteristics and identifying drivers. The evaluation results revealed the manifold benefits of our innovative design, ranging from its capacity to decrease extensive prepro-

cessing through temporal convolutions, its reduced computational complexity relative to recurrent-based architectures, and its potential to eliminate the necessity for model retraining, thereby enhancing overall generalization capabilities.

Subsequent chapters focused on agent behavior predictions for automated vehicles, addressing pivotal challenges of predictions within highly complex scenarios, including social reasoning and multimodality. In unsignalized intersection scenarios, we first introduced a deep learning-based method for predicting driver intentions using temporal convolutional networks. The results confirmed its effectiveness in capturing long-term dependencies. Expanding this, we developed a novel methodology to predict potential target vehicle behavior, generating multimodal predictions. Rigorous evaluations demonstrated its superiority over baselines and state-of-the-art methods. Additionally, we proposed an interaction-aware framework with spatial-temporal graph convolutional networks, effectively addressing social reasoning in complex scenarios and outperforming several state-of-the-art models.

Moreover, we designed a novel behavior prediction approach, namely STAG, to enhance social reasoning in behavior prediction. By capturing spatio-temporal correlations through graph convolutions and attention mechanisms, and anticipating future behaviors with temporal convolutions, STAG overcame the limitations of fixed interactions among agents. The model excelled in capturing spatio-temporal interactions within heterogeneous and homogeneous scenarios, outperforming several benchmarks and generalizing seamlessly across diverse environmental contexts.

Next, we proposed a novel context-aware prediction model, called CAPHA, that leverages maximum entropy inverse reinforcement learning to generate scene-compliant behaviors. Extensive evaluation using diverse agent groups exhibited its capability in accurately estimating future behavior distributions while considering uncertainty and scene conformability. The model transferability across various environmental contexts further highlighted its effectiveness. Also, its efficient inference time positioned CAPHA as a suitable solution for real-time deployment.

Finally, we introduced a hierarchical graph-based approach, namely GMP, that combines graph representations and attention mechanisms to enhance multimodal motion prediction accuracy while maintaining feasibility. By modeling interactions between entities in the driving environment, GMP showcased remarkable potential in precise multimodal motion prediction.

10.2 Future Research Directions

While this thesis has made several advancements in the field of driving behavior analysis and prediction, there remain several promising directions for future research. These directions can further contribute to the improvement of intelligent transportation systems and autonomous vehicles.

- **Multi-Agent behavior prediction.** Recent behavior prediction models adopt a modeling approach where the distribution over potential future behaviors is treated as independent for each individual agent. Extending the scope of prediction models to incorporate interactions and dependencies between multiple agents, particularly in complex traffic scenarios, presents a significant challenge and opportunity. It is important to distinguish this task from interaction-aware behavior prediction, which aims to capture specific interactions among agents while predicting individual trajectories within these interactions. On the other hand, the objective of multi-agent behavior prediction is to predict the behaviors of all agents simultaneously, regardless of the nature of their interactions. Developing advanced multi-agent models that can capture cooperative and adversarial behaviors among various traffic participants could enhance the accuracy of behavior prediction models.
- **Scene understanding and occlusion handling.** Occlusion occurs when the view of the ego vehicle is obstructed by other objects, vehicles, or obstacles in the environment, leading to limited or incomplete information about the surrounding agents behaviors. This limitation can significantly impact the accuracy of behavior predictions and the decision-making process for autonomous vehicles. To overcome the occlusion challenge, advanced techniques need to be developed that can effectively infer the intentions and motions of occluded agents based on partial or indirect information. One approach could involve leveraging sensor fusion techniques, such as combining data from cameras, LiDAR, radar, and other sensors to create a more comprehensive view of the environment. Another potential direction is to integrate communication technologies between vehicles and infrastructures. These technologies can facilitate the exchange of relevant information among vehicles and with the surrounding infrastructure, mitigating the challenges introduced by occlusion. Furthermore, incorporating probabilistic reasoning and uncertainty estimation into behavior prediction models can help account for the uncertainty caused by occlusion.

- **Context modeling with dynamic factors.** Ego-centric modeling, which involves computing trajectories from the perspective of the ego vehicle, can become computationally intensive in crowded and complex traffic scenarios. This is due to the need for repeated calculations for each surrounding agent. To address this challenge, an approach can be adopted where the scene is encoded only once but in a global frame of reference. By doing so, the computational load associated with processing dense scenes can be significantly reduced. More recent studies, including our proposed approach GMP, have shifted from ego-centric to global frame modeling. However, this research direction is still in its infancy. More advanced techniques are necessary to incorporate dynamic factors like temporary traffic rules or weather conditions. Current context modeling often focuses on static scene elements and agent behaviors, but real-world driving scenarios are subject to changing conditions. One approach could involve building dynamic maps that effectively fuse various data modalities, such as sensor data, high-definition maps, and vehicle-to-vehicle communication. This could lead to more accurate and robust behavior prediction models.
- **Integration of driving styles.** Drivers exhibit a wide range of driving behaviors that are influenced by a multitude of factors, including road conditions, traffic flow, weather, time of day, and personal preferences. These factors contribute to the dynamic nature of driving styles, where individuals adapt their behavior to navigate diverse driving scenarios effectively. For instance, drivers may switch between cautious driving in congested traffic and more assertive driving on open highways. Moreover, personal driving habits and moods can further diversify driving styles, leading to variations in acceleration, braking, lane changes, and interaction with other road users. Incorporating driving style detection models within driver identification systems has the potential to contribute to more reliable and robust identification outcomes across varying driving contexts.
- **Multiple impostor detection.** Taking the impostor detection research forward, it could be valuable to examine multiple impostors and their potential interactions within the driver identification and verification system. While the present work has primarily concentrated on single impostor detection, addressing scenarios where multiple unauthorized users attempt to deceive the driver identification and verification system presents a potential direction for enhancement. Developing methods and algorithms that can effectively identify and differentiate among mul-

tiple impostors concurrently is a complex challenge. This extension would require devising novel strategies to handle the increased complexity and potential interactions among multiple impostors, contributing to a more comprehensive and robust framework.

- **Federated learning for impostor detection.** Implementing a robust impostor detection solution requires addressing several challenges, including accurate differentiation of the authorized driver and the impostor, and effective adaption to various scenarios since the driving context may change due to factors such as traffic conditions, weather, and road type. Moreover, the system must handle the possibility of the authorized driver's behavior evolving over time due to changes in driving habits or personal preferences. Federated learning, which aggregates data from multiple sources without sharing raw data, can enhance the accuracy of the detection model while preserving driver privacy. This approach not only mitigates privacy risks associated with centralized models but also enables continuous learning and adaptation to diverse driving patterns. These efforts will explore novel strategies for detecting and mitigating adversarial attempts to deceive the model, ensuring the robustness and reliability of driver identification and impostor detection in real-world and connected vehicle environments.

Bibliography

- [1] W. H. Organization, “Global status report on road safety 2018,” tech. rep., World Health Organization, Dec. 2018.
- [2] T. Canada, “National collision database.” <https://wwwapps2.tc.gc.ca/Saf-Sec-Sur/7/NCDB-BNDC/p.aspx?l=en>. Accessed: May 02, 2023.
- [3] J. Huang, W. C. Lin, and Y.-K. Chin, “Adaptive vehicle control system with driving style recognition and road condition recognition,” Jan. 28 2010. US Patent App. 12/179,250.
- [4] A. Riener and A. Ferscha, “Supporting implicit human-to-vehicle interaction: Driver identification from sitting postures,” in *Symposium on Veh. Comput. Syst. (isvcs 2008)*, p. 10, 2008.
- [5] R. Chen, M. F. She, X. Sun, L. Kong, and Y. Wu, “Driver recognition based on dynamic handgrip pattern on steering wheel,” in *ACIS International Conf. SNPD*, pp. 107–112, 2011.
- [6] Z. Mahmood, T. Ali, S. Khattak, S. U. Khan, and L. T. Yang, “Automatic vehicle detection and driver identification framework for secure vehicle parking,” in *International Conf. FIT*, pp. 6–11, 2015.
- [7] J.-D. Wu and S.-H. Ye, “Driver identification using finger-vein patterns with radon transform and neural network,” *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5793–5799, 2009.
- [8] J. D. Wu and S. H. Ye, “Driver identification based on voice signal using continuous wavelet transform and artificial neural network techniques,” *Expert Syst. Appl.*, vol. 36, no. 2, pp. 1061–1069, 2009.
- [9] U. Fugiglando, E. Massaro, P. Santi, S. Milardo, K. Abida, R. Stahlmann, F. Netter, and C. Ratti, “Driving behavior analysis through can bus data in an uncontrolled environment,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, pp. 737–748, Feb 2019.

- [10] J. Hu, L. Xu, X. He, and W. Meng, "Abnormal driving detection based on normalized driving behavior," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6645–6652, 2017.
- [11] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Trans. Intell. Transp. Syst.*, 2020.
- [12] R. Fuller, "Towards a general theory of driver behaviour," *Accid. Anal. & Prev.*, vol. 37, no. 3, pp. 461–472, 2005.
- [13] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural Comput.*, vol. 11, no. 1, pp. 229–242, 1999.
- [14] J. Törnros, "Driving behaviour in a real and a simulated road tunnel—a validation study," *Accid. Anal. & Prev.*, vol. 30, no. 4, pp. 497–503, 1998.
- [15] NHTSA, "Distracted driving 2018," tech. rep., NHTSA, Apr. 2020.
- [16] T. I. E. Agency, "Tracking transport 2020 report." <https://www.iea.org/reports/tracking-transport-2020>. Accessed: July 27, 2020.
- [17] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Privacy Enhanc. Tech.*, no. 1, pp. 34–50, 2016.
- [18] S. M. Iranmanesh, H. Nourkhiz Mahjoub, H. Kazemi, and Y. P. Fallah, "An adaptive forward collision warning framework design based on driver distraction," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3925–3934, 2018.
- [19] B. Shi, L. Xu, J. Hu, Y. Tang, H. Jiang, W. Meng, and H. Liu, "Evaluating driving styles by normalizing driving behavior based on personalized driver modeling," *IEEE Trans. Syst., Man, Cybern.*, vol. 45, no. 12, pp. 1502–1508, 2015.
- [20] Y. Xun, J. Liu, N. Kato, Y. Fang, and Y. Zhang, "Automobile driver fingerprinting: A new machine learning based authentication scheme," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1417–1426, 2020.
- [21] J. Zhang, Z. Wu, F. Li, C. Xie, T. Ren, J. Chen, and L. Liu, "A deep learning framework for driving behavior identification on in-vehicle can-bus sensor data," *Sensors*, vol. 19, no. 6, p. 1356, 2019.
- [22] U. Kiencke, S. Dais, and M. Litschel, "Automotive serial controller area network," in *SAE Int.*, pp. 823–828, SAE Int., feb 1986.

- [23] C. Zoran, C. Marinoiu, and M. Vladoiu, “Driving style analysis using data mining techniques,” *Int. J. Comput. Commun.*, vol. 5, no. 5, pp. 654–663, 2010.
- [24] A. Mudgal, S. Hallmark, A. Carriquiry, and K. Gkritza, “Driving behavior at a roundabout: A hierarchical bayesian regression analysis,” *Transp. Res. D, Transp. Environ.*, vol. 26, pp. 20–26, 2014.
- [25] A. Chowdhury, T. Chakravarty, A. Ghose, T. Banerjee, and P. Balamuralidhar, “Investigations on driver unique identification from smartphone’s gps data alone,” *J. Adv. Transp.*, vol. 2018, 2018.
- [26] V. Manzoni, A. Corti, P. De Luca, and S. M. Savaresi, “Driving style estimation via inertial measurements,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 777–782, IEEE, 2010.
- [27] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M. C. González, “Safe driving using mobile phones,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1462–1468, 2012.
- [28] J. Carmona, F. García, D. Martín, A. d. l. Escalera, and J. M. Armingol, “Data fusion for driver behaviour analysis,” *Sensors*, vol. 15, no. 10, pp. 25968–25991, 2015.
- [29] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, “Mobile phone based drunk driving detection,” in *Conf.on Pervas. Comput. Technol. Healthcare*, pp. 1–8, 2010.
- [30] R. Sun, Q. Cheng, F. Xie, W. Zhang, T. Lin, and W. Y. Ochieng, “Combining machine learning and dynamic time wrapping for vehicle driving event detection using smartphones,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–14, 2019.
- [31] G. Castignani, T. Derrmann, R. Frank, and T. Engel, “Smartphone-based adaptive driving maneuver detection: A large-scale evaluation study,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, pp. 2330–2339, Sep. 2017.
- [32] D. A. Johnson and M. M. Trivedi, “Driving style recognition using a smartphone as a sensor platform,” in *IEEE Conf. Intell. Transp. Syst.*, (USA), pp. 1609–1615, IEEE, Oct 2011.
- [33] J. Engelbrecht, M. J. Booysen, G.-J. van Rooyen, and F. J. Bruwer, “Survey of smartphone-based sensing in vehicles for intelligent transportation system applications,” *IET Intell. Trans. Sy.*, vol. 9, no. 10, pp. 924–935, 2015.
- [34] A. Das, M. N. Khan, and M. M. Ahmed, “Detecting lane change maneuvers using shrp2 naturalistic driving data: a comparative study machine learning techniques,” *Accid. Anal. & Prev.*, vol. 142, p. 105578, 2020.

- [35] N. Khairdoost, M. Shirpour, M. A. Bauer, and S. S. Beauchemin, “Real-time maneuver prediction using lstm,” *IEEE Trans. Intell. Veh.*, pp. 1–1, 2020.
- [36] H. Chen and L. Chen, “Support vector machine classification of drunk driving behaviour,” *Int. J. Envir. Res. Pub. health*, vol. 14, no. 1, p. 108, 2017.
- [37] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr, “A combined model- and learning-based framework for interaction-aware maneuver prediction,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1538–1550, 2016.
- [38] R. Coppola and M. Morisio, “Connected car: technologies, issues, future trends,” *ACM Comput. Surv.*, vol. 49, no. 3, pp. 1–36, 2016.
- [39] A. Zyner, S. Worrall, and E. Nebot, “Naturalistic driver intention and path prediction using recurrent neural networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1584–1594, 2020.
- [40] M. Zhang, R. Fu, D. D. Morris, and C. Wang, “A framework for turning behavior classification at intersections using 3d lidar,” *IEEE Trans. Veh. Technol.*, vol. 68, pp. 7431–7442, Aug 2019.
- [41] W. Han, W. Wang, X. Li, and J. Xi, “Statistical-based approach for driving style recognition using bayesian probability with kernel density estimation,” *IET Intell. Trans. Sy.*, vol. 13, no. 1, pp. 22–30, 2019.
- [42] A. Ulahannan, P. Jennings, L. Oliveira, and S. Birrell, “Designing an adaptive interface: Using eye tracking to classify how information usage changes over time in partially automated vehicles,” *IEEE Access*, vol. 8, pp. 16865–16875, 2020.
- [43] W. Wang, J. Xi, A. Chong, and L. Li, “Driving style classification using a semisupervised support vector machine,” *IEEE Trans. Man-Mach Syst.*, vol. 47, no. 5, pp. 650–660, 2017.
- [44] Z. Wang, X. Liao, C. Wang, D. Oswald, G. Wu, K. Boriboonsomsin, M. Barth, K. Han, B. Kim, and P. Tiwari, “Driver behavior modeling using game engine and real vehicle: A learning-based approach,” *IEEE Trans. Intell. Veh.*, 2020.
- [45] O. Dehzangi and S. Masilamani, “Unobtrusive driver drowsiness prediction using driving behavior from vehicular sensors,” in *Conf. Pattern Recog.*, pp. 3598–3603, 2018.
- [46] M. Wang, N. Jeong, K. Kim, S. Choi, S. Yang, S. You, J. Lee, and M. Suh, “Drowsy behavior detection based on driving information,” *Int. J. Auto. Tech.*, vol. 17, no. 1, pp. 165–173, 2016.

- [47] A. El Basiouni El Masri, H. Artail, and H. Akkary, "Toward self-policing: Detecting drunk driving behaviors through sampling can bus data," in *Conf. Elect. Comput. Technol. App.*, pp. 1–5, 2017.
- [48] Z. Li, X. Jin, and X. Zhao, "Drunk driving detection based on classification of multivariate time series," *J. Safety Res.*, vol. 54, pp. 61–e29, 2015.
- [49] M. M. Shirazi and A. B. Rad, "Detection of intoxicated drivers using online system identification of steering behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1738–1747, 2014.
- [50] Z. Li, H. Wang, Y. Zhang, and X. Zhao, "Random forest-based feature selection and detection method for drunk driving recognition," *Int. J. Dist. Sens. Net.*, vol. 16, no. 2, 2020.
- [51] R. Terzi, S. Sagiroglu, and M. U. Demirezen, "Big data perspective for driver/driving behavior," *IEEE Intell. Transp. Syst. Mag.*, 2018.
- [52] S. A. Birrell, M. Fowkes, and P. A. Jennings, "Effect of using an in-vehicle smart driving aid on real-world driver performance," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1801–1810, 2014.
- [53] C. Craye, A. Rashwan, M. S. Kamel, and F. Karray, "A multi-modal driver fatigue and distraction assessment system," *J. Intell. Transport. Sys. Research*, vol. 14, no. 3, pp. 173–194, 2016.
- [54] J. Gwak, A. Hirao, and M. Shino, "An investigation of early detection of driver drowsiness using ensemble machine learning based on hybrid sensing," *Applied Sciences*, vol. 10, no. 8, p. 2890, 2020.
- [55] Z. Chen, Y. Zhang, C. Wu, and B. Ran, "Understanding individualization driving states via latent dirichlet allocation model," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 2, pp. 41–53, 2019.
- [56] D. Chu, Z. Deng, Y. He, C. Wu, C. Sun, and Z. Lu, "Curve speed model for driver assistance based on driving style classification," *IET Intell. Trans. Sy.*, vol. 11, no. 8, pp. 501–510, 2017.
- [57] W. Wang, J. Xi, and D. Zhao, "Driving style analysis using primitive driving patterns with bayesian nonparametric approaches," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 8, pp. 2986–2998, 2019.

- [58] S. Schneegass, B. Pflöging, N. Broy, F. Heinrich, and A. Schmidt, “A data set of real world driving to assess driver workload,” in *conf. Auto. user Inter. Interac. Veh. App.*, pp. 150–157, 2013.
- [59] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, “Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning,” in *Proceedings of the IEEE/CVF CVPR*, 2018.
- [60] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena, “Car that knows before you do: Anticipating maneuvers via learning temporal driving models,” in *IEEE Conf. Comput. Vision*, December 2015.
- [61] E. Santana and G. Hotz, “Learning a driving simulator,” *arXiv preprint arXiv:1608.01230*, 2016.
- [62] A. El Mekki, A. Bouhoute, and I. Berrada, “Improving driver identification for the next-generation of in-vehicle software systems,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7406–7415, 2019.
- [63] M. A. Rahim, L. Zhu, X. Li, J. Liu, Z. Zhang, Z. Qin, S. Khan, and K. Gai, “Zero-to-stable driver identification: A non-intrusive and scalable driver identification scheme,” *IEEE Trans. Veh. Technol.*, vol. 69, pp. 163–171, Jan 2020.
- [64] S. Taamneh, P. Tsiamyrtzis, M. Dcosta, P. Buddharaju, A. Khatri, M. Manser, T. Ferris, R. Wunderlich, and I. Pavlidis, “A multimodal dataset for various forms of distracted driving,” *Scientific data*, vol. 4, p. 170110, 2017.
- [65] J. M. Hankey, M. A. Perez, and J. A. McClafferty, “Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets,” tech. rep., Virginia Tech Transp. Inst., 2016.
- [66] T. A. Dingus, S. G. Klauer, V. L. Neale, A. Petersen, S. E. Lee, J. Sudweeks, M. A. Perez, J. Hankey, D. Ramsey, S. Gupta, *et al.*, “The 100-car naturalistic driving study, phase ii-results of the 100-car field experiment,” tech. rep., United States. Department of Transp. NHTSA, 2006.
- [67] U. D. of Transp., “Safety pilot model deployment data,” tech. rep., US Department of Transp., Oct. 2015.
- [68] M. A. Regan, A. Williamson, R. Grzebieta, J. Charlton, M. Lenne, B. Watson, N. Harworth, A. Rakotonirainy, J. Woolley, R. Anderson, *et al.*, “The australian 400-car naturalistic driving study: Innovation in road safety research and policy,” in *Proc. Australasian Road Safety Res., Policing Educ. Conf.*, pp. 1–13, 2013.

- [69] H. Abut, H. Erdoğan, A. Erçil, A. B. Çürüklü, H. C. Koman, F. Tas, A. Ö. Argunşah, B. Akan, H. Karabalkan, E. Çökelek, *et al.*, “Data collection with” uyanik”: too much pain; but gains are coming,” *Sabancı University*, 2007.
- [70] E. Romera, L. M. Bergasa, and R. Arroyo, “Need data for driver behaviour analysis? presenting the public uah-driveset,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 387–392, Nov 2016.
- [71] B. I. Kwak, J. Woo, and H. K. Kim, “Know your master: Driver profiling-based anti-theft method,” in *Conf. Privacy, Security and Trust (PST)*, pp. 211–218, IEEE, 2016.
- [72] A. Mohammadnazar, R. Arvin, and A. J. Khattak, “Classifying travelers’ driving style using basic safety messages generated by connected vehicles: Application of unsupervised machine learning,” *Transp. Res. Part C Emerg. Technol.*, vol. 122, p. 102917, 2021.
- [73] O. Taubman-Ben-Ari, M. Mikulincer, and O. Gillath, “The multidimensional driving style inventory—scale construct and validation,” *Accid. Anal. & Prev.*, vol. 36, no. 3, pp. 323–332, 2004.
- [74] C.-H. Yang, C.-C. Chang, and D. Liang, “A novel gmm-based behavioral modeling approach for smartwatch-based driver authentication,” *Sensors*, vol. 18, no. 4, p. 1007, 2018.
- [75] B. I. Kwak, M. L. Han, and H. K. Kim, “Driver identification based on wavelet transform using driving patterns,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2400–2410, 2021.
- [76] C. Marina Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, “Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, pp. 666–676, March 2018.
- [77] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, “Argoverse 2: Next generation datasets for self-driving perception and forecasting,” in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.
- [78] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, “The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections,” in *IEEE IV*, pp. 1929–1934, IEEE, 2020.
- [79] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens,

- and D. Anguelov, “Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9710–9719, October 2021.
- [80] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, “Argoverse: 3d tracking and forecasting with rich maps,” in *Proceedings of the IEEE/CVF CVPR*, 2019.
- [81] A. Zyner, S. Worrall, and E. M. Nebot, “Acr five roundabouts dataset: Naturalistic driving at unsignalized intersections,” *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 4, pp. 8–18, 2019.
- [82] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, *et al.*, “Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps,” *arXiv preprint arXiv:1910.03088*, 2019.
- [83] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, “The apolloscape open dataset for autonomous driving and its application,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2702–2719, 2019.
- [84] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nusenes: A multimodal dataset for autonomous driving,” *arXiv preprint arXiv:1903.11027*, 2019.
- [85] R. Krajewski, J. Bock, L. Kloecker, and L. Eckstein, “The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems,” in *Proceedings of Intell. Transp. Syst. (ITSC)*, pp. 2118–2125, IEEE, 2018.
- [86] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, “Learning social etiquette: Human trajectory understanding in crowded scenes,” in *Proceedings of IEEE/CVF ECCV*, pp. 549–565, Springer, 2016.
- [87] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Proceedings of the IEEE/CVF CVPR*, 2012.
- [88] J. Colyar and J. Halkias, “Us highway 101 dataset,” Tech. Rep. FHWA-HRT07-030, Federal Highway Administration (FHWA), 2007.
- [89] J. Colyar and J. Halkias, “Us highway i-80 dataset,” Tech. Rep. FHWA-HRT-06-137, Federal Highway Administration (FHWA), 2006.

- [90] M. M. Bejani and M. Ghatee, “A context aware system for driving style evaluation by an ensemble learning on smartphone sensors data,” *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 303–320, 2018.
- [91] A. Campolina, A. Boukerche, and A. A. F. Loureiro, “Characterizing car trips through information theory metrics,” in *Proceedings of ACM Conference MSWiM*, p. 241–245, ACM, 2019.
- [92] V. C. Magaña and M. Muñoz-Organero, “Artemisa: A personal driving assistant for fuel saving,” *IEEE Trans. Mobile Comput.*, vol. 15, no. 10, pp. 2437–2451, 2016.
- [93] D. Dörr, D. Grabengieser, and F. Gauterin, “Online driving style recognition using fuzzy logic,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 1021–1026, 2014.
- [94] A. Campolina, A. Boukerche, and A. A. F. Loureiro, “Context and location awareness in eco-driving recommendations,” in *NOMS 2020 - 2020 IEEE/IFIP NOMS*, pp. 1–6, 2020.
- [95] A. Aksjonov, P. Nedoma, V. Vodovozov, E. Petlenkov, and M. Herrmann, “Detection and evaluation of driver distraction using machine learning and fuzzy logic,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2048–2059, 2019.
- [96] J. Nilsson, J. Fredriksson, and E. Coelingh, “Rule-based highway maneuver intention recognition,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 950–955, 2015.
- [97] P. Lytrivis, G. Thomaidis, and A. Amditis, “Cooperative path prediction in vehicular environments,” in *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pp. 803–808, IEEE, 2008.
- [98] M. Brännström, E. Coelingh, and J. Sjöberg, “Model-based threat assessment for avoiding arbitrary vehicle collisions,” *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 658–669, 2010.
- [99] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. Dietmayer, “Imm object tracking for high dynamic driving maneuvers,” in *IEEE Intelligent Vehicles Symposium, 2004*, pp. 825–830, IEEE, 2004.
- [100] B. Jin, B. Jiu, T. Su, H. Liu, and G. Liu, “Switched kalman filter-interacting multiple model algorithm based on optimal autoregressive model for manoeuvring target tracking,” *IET Radar, Sonar & Navigation*, vol. 9, no. 2, pp. 199–209, 2015.
- [101] M. Althoff and A. Mergel, “Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1237–1247, 2011.

- [102] K. Okamoto, K. Berntorp, and S. Di Cairano, “Driver intention-based vehicle threat assessment using random forests and particle filtering,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13860–13865, 2017.
- [103] G. Li, S. E. Li, B. Cheng, and P. Green, “Estimation of driving style in naturalistic highway traffic using maneuver transition probabilities,” *Transp. Res. C: Emerg. Technol.*, vol. 74, pp. 113–125, 2017.
- [104] J. Xie, A. R. Hilal, and D. Kulic, “Driver distraction recognition based on smartphone sensor data,” in *Conf Syst., Man, Cybern.*, pp. 801–806, 2018.
- [105] L. Moreira-Matias and H. Farah, “On developing a driver identification methodology using in-vehicle data recorders,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, pp. 2387–2396, Sep. 2017.
- [106] N. Virojboonkiate, A. Chanakitkarnchok, P. Vateekul, and K. Rojviboonchai, “Public transport driver identification system using histogram of acceleration data,” *J Adv. Transp.*, 2019.
- [107] S. Ezzini, I. Berrada, and M. Ghogho, “Who is behind the wheel? driver identification and fingerprinting,” *J. Big Data*, vol. 5, no. 1, p. 9, 2018.
- [108] G. Rigas, Y. Goletsis, and D. I. Fotiadis, “Real-time driver’s stress event detection,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 221–234, 2012.
- [109] L. Yan, Z. Huang, Y. Zhang, L. Zhang, D. Zhu, and B. Ran, “Driving risk status prediction using bayesian networks and logistic regression,” *IET Intell. Trans. Sy.*, vol. 11, no. 7, pp. 431–439, 2017.
- [110] C. R. Bingham, S. Bao, C. Flannagan, A. K. Pradhan, *et al.*, “Using naturalistic driving performance data to develop an empirically defined model of distracted driving,” tech. rep., Nextrans, 2016.
- [111] Y. Zheng and J. H. L. Hansen, “Lane-change detection from steering signal using spectral segmentation and learning-based classification,” *IEEE Trans. Intell. Veh.*, vol. 2, no. 1, pp. 14–24, 2017.
- [112] M. A. Rahim, J. Liu, Z. Zhang, L. Zhu, X. Li, and S. Khan, “Who is driving? event-driven driver identification and impostor detection through support vector machine,” *IEEE Sens. J.*, pp. 1–1, 2020.

- [113] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, “Behavior classification algorithms at intersections and validation using naturalistic data,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 601–606, 2011.
- [114] H. M. Mandalia and M. D. D. Salvucci, “Using support vector machines for lane-change detection,” in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 49, pp. 1965–1969, SAGE Publications Sage CA: Los Angeles, CA, 2005.
- [115] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, “Learning-based approach for online lane change intention prediction,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 797–802, IEEE, 2013.
- [116] X. Wang and C. Xu, “Driver drowsiness detection based on non-intrusive metrics considering individual specifics,” *Accid. Anal. & Prev.*, vol. 95, pp. 350–357, 2016.
- [117] Y. Yao, X. Zhao, H. Du, Y. Zhang, G. Zhang, and J. Rong, “Classification of fatigued and drunk driving based on decision tree methods: a simulator study,” *Int. J. Envir. Res. Pub. health*, vol. 16, no. 11, p. 1935, 2019.
- [118] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.
- [119] Y. Zhao, A. Mammeri, and A. Boukerche, “A novel real-time driver monitoring system based on deep convolutional neural network,” in *2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pp. 1–7, 2019.
- [120] M. M. Bejani and M. Ghatee, “Convolutional neural network with adaptive regularization to classify driving styles on smartphones,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 543–552, 2020.
- [121] S. H. Sánchez, R. F. Pozo, and L. A. H. Gómez, “Deep neural networks for driver identification using accelerometer signals from smartphones,” in *Conf. Business Inform. Syst.*, pp. 206–220, Springer, 2019.
- [122] D. Jeong, M. Kim, K. Kim, T. Kim, J. Jin, C. Lee, and S. Lim, “Real-time driver identification using vehicular big data and deep learning,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 123–130, 2018.
- [123] J. Xu, S. Pan, P. Z. H. Sun, S. Hyeong Park, and K. Guo, “Human-factors-in-driving-loop: Driver identification and verification via a deep learning approach using psychological behavioral data,” *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3383–3394, 2023.

- [124] S. Würtz and U. Göhner, “Driving style analysis using recurrent neural networks with lstm cells,” *J. Adv. Inform. Tech.*, vol. 11, no. 1, 2020.
- [125] S. M. Kouchak and A. Gaffar, “Detecting driver behavior using stacked long short term memory network with attention layer,” *IEEE Trans. Intell. Transp. Syst.*, 2020.
- [126] D. Park, H. Ryu, Y. Yang, J. Cho, J. Kim, and K.-J. Yoon, “Leveraging future relationship reasoning for vehicle trajectory prediction,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [127] J. Li, H. Ma, Z. Zhang, J. Li, and M. Tomizuka, “Spatio-temporal graph dual-attention network for multi-agent prediction and tracking,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 10556–10569, 2022.
- [128] M. L. Bernardi, M. Cimitile, F. Martinelli, F. Mercaldo, *et al.*, “Driver and path detection through time-series classification,” *Journal of Advanced Transportation*, vol. 2018, 2018.
- [129] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” *IEEE Trans. Neural. Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2020.
- [130] Y. Wu, G. Chen, Z. Li, L. Zhang, L. Xiong, Z. Liu, and A. Knoll, “Hsta: A hierarchical spatio-temporal attention model for trajectory prediction,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 11, pp. 11295–11307, 2021.
- [131] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, “Vectornet: Encoding hd maps and agent dynamics from vectorized representation,” in *Proceedings of the IEEE/CVF Proceedings of IEEE/CVF CVPR*, pp. 11525–11533, 2020.
- [132] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, “Learning lane graph representations for motion forecasting,” in *Proceedings of IEEE/CVF ECCV*, pp. 541–556, Springer, 2020.
- [133] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, and F. Nashashibi, “Attention based vehicle trajectory prediction,” *IEEE Trans. Intell. Vehicles*, vol. 6, no. 1, pp. 175–185, 2020.
- [134] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *Proceedings of the IEEE Proceedings of IEEE/CVF CVPR*, pp. 2255–2264, 2018.
- [135] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

- [136] H. Li, H. Wang, L. Liu, and M. Gruteser, “Automatic unusual driving event identification for dependable self-driving,” in *ACM Conf. Embed. Net. Sens. Syst.*, pp. 15–27, 2018.
- [137] J. Hu, X. Zhang, and S. Maybank, “Abnormal driving detection with normalized driving behavior data: A deep learning approach,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 6943–6951, 2020.
- [138] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, “It is not the journey but the destination: Endpoint conditioned trajectory prediction,” in *Proceedings of IEEE/CVF ECCV*, pp. 759–776, Springer, 2020.
- [139] H. Cheng, W. Liao, X. Tang, M. Y. Yang, M. Sester, and B. Rosenhahn, “Exploring dynamic context for multi-path trajectory prediction,” in *Proceedings of IEEE ICRA*, pp. 12795–12801, IEEE, 2021.
- [140] S. Levine, “Reinforcement learning and control as probabilistic inference: Tutorial and review,” *arXiv preprint arXiv:1805.00909*, 2018.
- [141] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, “Deep reinforcement learning for autonomous driving: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.
- [142] P. Wang, C.-Y. Chan, and A. de La Fortelle, “A reinforcement learning based approach for automated lane change maneuvers,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 1379–1384, IEEE, 2018.
- [143] K. Guo, W. Liu, and J. Pan, “End-to-end trajectory distribution prediction based on occupancy grid maps,” in *Proceedings of the IEEE/CVF CVPR*, pp. 2242–2251, 2022.
- [144] N. Deo and M. M. Trivedi, “Trajectory forecasts in unknown environments conditioned on grid-based plans,” *arXiv preprint arXiv:2001.00735*, 2020.
- [145] A. El Khatib, C. Ou, and F. Karray, “Driver inattention detection in the context of next-generation autonomous vehicles design: A survey,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–14, 2019.
- [146] S. Birrell, J. Taylor, A. McGordon, J. Son, and P. Jennings, “Analysis of three independent real-world driving studies: A data driven and expert analysis approach to determining parameters affecting fuel economy,” *Transp. Res. D, Transp. Environ.*, vol. 33, pp. 74–86, 2014.

- [147] T. Wakita, K. Ozawa, C. Miyajima, K. Igarashi, K. Itou, K. Takeda, and F. Itakura, "Driver identification using driving behavior signals," *IEICE Trans. Inf. Syst.*, vol. 89, no. 3, pp. 1188–1194, 2006.
- [148] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura, "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.
- [149] H. Qian, Y. Ou, X. Wu, X. Meng, Y. Xu, *et al.*, "Support vector machine for behavior-based driver identification system," *Journal of Robotics*, vol. 2010, 2010.
- [150] C. H. Espino-Salinas, H. Luna-García, J. M. Celaya-Padilla, J. A. Morgan-Benita, C. Vera-Vasquez, W. J. Sarmiento, C. E. Galván-Tejada, J. I. Galván-Tejada, H. Gamboa-Rosales, and K. O. Villalba-Condori, "Driver identification using statistical features of motor activity and genetic algorithms," *Sensors*, vol. 23, no. 2, p. 784, 2023.
- [151] A. Boukerche, *Algorithms and protocols for wireless sensor networks*. Wiley Online Library, 2009.
- [152] A. Boukerche, H. A. Oliveira, E. F. Nakamura, and A. A. Loureiro, "Vehicular ad hoc networks: A new challenge for localization-based systems," *Computer communications*, vol. 31, no. 12, pp. 2838–2849, 2008.
- [153] A. Boukerche and M. S. M. A. Notare, "Behavior-based intrusion detection in mobile phone systems," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1476–1490, 2002.
- [154] R. R. M. Putri, C. H. Yang, C. C. Chang, and D. Liang, "Smartwatch-based open-set driver identification by using gmm-based behavior modeling approach," *IEEE Sensors Journal*, pp. 1–1, 2020.
- [155] A. Wahab, C. Quek, C. K. Tan, and K. Takeda, "Driving profile modeling and recognition based on soft computing approach," *IEEE trans. neural netw.*, vol. 20, no. 4, pp. 563–582, 2009.
- [156] E. Öztürk and E. Erzin, "Driver status identification from driving behavior signals," in *Digital signal processing for in-vehicle systems and safety*, pp. 31–55, Springer, 2012.
- [157] S. Jafarnejad, G. Castignani, and T. Engel, "Revisiting gaussian mixture models for driver identification," in *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–7, IEEE, 2018.

- [158] Y.-J. Yang, C.-M. Chao, C.-C. Yeh, and C.-Y. Lin, “Wfid: Driver identity recognition based on wi-fi signals,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 1, pp. 679–688, 2023.
- [159] M. V. Martínez, I. Del Campo, J. Echanobe, and K. Basterretxea, “Driving behavior signals and machine learning: A personalized driver assistance system,” in *Proceedings of Intell. Transp. Syst. (ITSC)*, pp. 2933–2940, IEEE, 2015.
- [160] D. Hallac, A. Sharang, R. Stahlmann, A. Lamprecht, M. Huber, M. Roehder, R. Sosič, and J. Leskovec, “Driver identification using automobile sensor data from a single turn,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 953–958, Nov 2016.
- [161] E. Kristianto and P.-C. Lin, “Automobile theft detection by driving behavior identification using deep autoencoder,” in *2021 International Conference on Security and Information Technologies with AI, Internet Computing and Big-data Applications*, pp. 191–200, Springer, 2022.
- [162] Z. Ding, D. Xu, C. Tu, H. Zhao, M. Moze, F. Aioun, and F. Guillemard, “Driver identification through heterogeneity modeling in car-following sequences,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17143–17156, 2022.
- [163] J. Lim, Y. Baek, and B. Chae, “Temporal early exiting with confidence calibration for driver identification based on driving sensing data,” *IEEE Access*, vol. 10, pp. 132095–132107, 2022.
- [164] S. Hernández Sánchez, R. F. Pozo, and L. A. H. Gómez, “Driver identification and verification from smartphone accelerometers using deep neural networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 97–109, 2022.
- [165] L. Zeng, M. Al-Rifai, M. Nolting, and W. Nejdl, “Towards building reliable deep learning based driver identification systems,” in *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 766–773, IEEE, 2022.
- [166] C. Ravi, A. Tigga, G. T. Reddy, S. Hakak, and M. Alazab, “Driver identification using optimized deep learning model in smart transportation,” *ACM Transactions on Internet Technology*, vol. 22, no. 4, pp. 1–17, 2022.
- [167] N. Abdennour, T. Ouni, and N. Ben Amor, “Driver identification using only the can-bus vehicle data through an rcn deep learning approach,” *Rob. Auton. Syst.*, vol. 136, p. 103707, 2021.
- [168] H. Dang and J. Fürnkranz, “Driver information embedding with siamese lstm networks,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 935–940, 2019.

- [169] J. Chen, Z. Wu, and J. Zhang, “Driver identification based on hidden feature extraction by using adaptive nonnegativity-constrained autoencoder,” *Applied Soft Comput.*, vol. 74, pp. 1–9, 2019.
- [170] T. Tanprasert, C. Saiprasert, and S. Thajchayapong, “Combining unsupervised anomaly detection and neural networks for driver identification,” *J Adv. Transp.*, vol. 2017, 2017.
- [171] M. V. Martínez, J. Echanobe, and I. del Campo, “Driver identification and impostor detection based on driving behavior signals,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 372–378, Nov 2016.
- [172] S. Jafarnejad, G. Castignani, and T. Engel, “Towards a real-time driver identification mechanism based on driving sensing data,” in *IEEE Conf. Intell. Transp. Syst.*, pp. 1–7, 2017.
- [173] I. Del Campo, R. Finker, M. V. Martinez, J. Echanobe, and F. Doctor, “A real-time driver identification system based on artificial neural networks and cepstral analysis,” in *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 1848–1855, IEEE, 2014.
- [174] J. Echanobe, I. del Campo, and M. V. Martínez, “Design and optimization of a neural network-based driver recognition system by means of a multiobjective genetic algorithm,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3745–3750, IEEE, 2016.
- [175] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a” siamese” time delay neural network,” *Advances in neural information processing systems*, pp. 737–737, 1994.
- [176] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, vol. 2, Lille, 2015.
- [177] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Proceedings of the IEEE Proceedings of IEEE/CVF CVPR*, pp. 1701–1708, 2014.
- [178] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Proceedings of IEEE/CVF CVPR*, pp. 815–823, 2015.
- [179] K. Liu, S. Gao, P. Qiu, X. Liu, B. Yan, and F. Lu, “Road2vec: Measuring traffic interactions in urban road system from massive travel routes,” *ISPRS International Journal of Geo-Information*, vol. 6, no. 11, p. 321, 2017.

- [180] S. Jafarnejad, *Machine learning-based methods for driver identification and behavior assessment: applications for CAN and floating car data*. PhD thesis, Univ. of Luxembourg, Jan. 2020.
- [181] J. Yang, R. Zhao, M. Zhu, D. Hallac, J. Sodnik, and J. Leskovec, “Driver2vec: Driver identification from automotive data,” in *6th Workshop on Mining and Learning from Time Series (MiLeTS)*, Aug 2020.
- [182] H. Cai, Z. Hu, Z. Chen, and D. Zhu, “A driving fingerprint map method of driving characteristic representation for driver identification,” *IEEE Access*, vol. 6, pp. 71012–71019, 2018.
- [183] A. Bouhout, R. Oucheikh, K. Boubouh, and I. Berrada, “Advanced driving behavior analytics for an improved safety assessment and driver fingerprinting,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2171–2184, 2019.
- [184] G. Kar, S. Jain, M. Gruteser, J. Chen, F. Bai, and R. Govindan, “Predriveid: pre-trip driver identification from in-vehicle data,” in *ACM/IEEE Symposium on Edge Comput.*, pp. 1–12, 2017.
- [185] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, D. J. Ramsey, *et al.*, “The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data,” *United States. National Highway Traffic Safety Administration*, 2006.
- [186] M. A. Regan, C. Hallett, and C. P. Gordon, “Driver distraction and driver inattention: Definition, relationship and taxonomy,” *Accid. Anal. & Prev.*, vol. 43, no. 5, pp. 1771–1781, 2011.
- [187] T. A. Ranney, W. R. Garrott, and M. J. Goodman, “Nhtsa driver distraction research: Past, present, and future,” tech. rep., SAE Tech., 2001.
- [188] Y. Dong, Z. Hu, K. Uchimura, and N. Murayama, “Driver inattention monitoring system for intelligent vehicles: A review,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 596–614, 2011.
- [189] S. Kaplan, M. A. Guvensan, A. G. Yavuz, and Y. Karalurt, “Driver behavior analysis for safe driving: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3017–3032, 2015.
- [190] M. Shahverdy, M. Fathy, R. Berangi, and M. Sabokrou, “Driver behavior detection and classification using deep convolutional neural networks,” *Expert Syst. Appl.*, vol. 149, p. 113240, 2020.

- [191] H. Zhang, C. Wu, X. Yan, and T. Z. Qiu, "The effect of fatigue driving on car following behavior," *Transp. Res. F. traffif psychol. behav.*, vol. 43, pp. 80–89, 2016.
- [192] Y. Liao, S. E. Li, W. Wang, Y. Wang, G. Li, and B. Cheng, "Detection of driver cognitive distraction: A comparison study of stop-controlled intersection and speed-limited highway," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1628–1637, 2016.
- [193] A. Kashevnik, I. Lashkov, and A. Gurtov, "Methodology and mobile application for driver behavior analysis and accident prevention," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2427–2436, 2020.
- [194] M. Schöler, C. Onnen, and C. Bielaczek, "A fuzzy-system for a classification of the driver behavior and the driving situation," *IFAC Proceedings Volumes*, vol. 30, no. 8, pp. 693–698, 1997.
- [195] A. F. for Traffic Safety, "2019 traffic safety culture index," tech. rep., AAA Foundation for Traffic Safety, June 2020.
- [196] D. Zhong, P. Sun, and A. Boukerche, "Empirical study and analysis of the impact of traffic flow control at road intersections on vehicle energy consumption," in *ACM MobiWac*, p. 21–28, ACM, 2020.
- [197] S. Yang, W. Wang, F. Zhang, Y. Hu, and J. Xi, "Driving-style-oriented adaptive equivalent consumption minimization strategies for hevs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9249–9261, 2018.
- [198] Y. Ma, Z. Li, Y. Li, H. Li, and R. Malekian, "Driving style estimation by fusing multiple driving behaviors: a case study of freeway in china," *Clust. Comput.*, vol. 22, pp. 8259–8269, 2019.
- [199] A. Boukerche, A. Martirosyan, and R. Pazzi, "An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks," *Mobile Networks and Applications*, vol. 13, pp. 614–626, 2008.
- [200] A. Boukerche and S. Nikolettseas, "Protocols for data propagation in wireless sensor networks," *Wireless communications systems and networks*, pp. 23–51, 2004.
- [201] S. Garg, K. Kaur, S. Batra, G. Kaddoum, N. Kumar, and A. Boukerche, "A multi-stage anomaly detection scheme for augmenting the security in iot-enabled applications," *Future Generation Computer Systems*, vol. 104, pp. 105–118, 2020.

- [202] R. W. Coutinho, A. Boukerche, L. F. Vieira, and A. A. Loureiro, “A novel void node recovery paradigm for long-term underwater sensor networks,” *Ad Hoc Networks*, vol. 34, pp. 144–156, 2015.
- [203] F. A. Silva, A. Boukerche, T. R. B. Silva, L. B. Ruiz, E. Cerqueira, and A. A. Loureiro, “Vehicular networks: A new challenge for content-delivery-based applications,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 1, pp. 1–29, 2016.
- [204] NHTSA, “2018 fatal motor vehicle crashes: Overview,” tech. rep., NHTSA, Oct. 2019.
- [205] Z. Li, K. Zhang, B. Chen, Y. Dong, and L. Zhang, “Driver identification in intelligent vehicle systems using machine learning algorithms,” *IET Intell. Trans. Sys.*, vol. 13, no. 1, pp. 40–47, 2019.
- [206] K. H. Lee, K. H. Baek, S. B. Choi, N. T. Jeong, H. U. Moon, E. S. Lee, H. M. Kim, and M. W. Suh, “Development of three driver state detection models from driving information using vehicle simulator; normal, drowsy and drunk driving,” *Int. J. Automot. Technol.*, vol. 20, no. 6, pp. 1205–1219, 2019.
- [207] Z. Wang, P. Sun, Y. Hu, and A. Boukerche, “A novel mixed method of machine learning based models in vehicular traffic flow prediction,” in *Proceedings of ACM MSWiM*, pp. 95–101, 2022.
- [208] B. Alipour, L. Tonetto, R. Ketabi, A. Yi Ding, J. Ott, and A. Helmy, “Where are you going next? a practical multi-dimensional look at mobility prediction,” in *Proceedings of ACM Conference MSWiM*, pp. 5–12, 2019.
- [209] Y. Xing, C. Lv, X. Mo, Z. Hu, C. Huang, and P. Hang, “Toward safe and smart mobility: energy-aware deep learning for driving behavior analysis and prediction of connected vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4267–4280, 2021.
- [210] Y. Xing, C. Lv, and D. Cao, “Personalized vehicle trajectory prediction based on joint time-series modeling for connected vehicles,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1341–1352, 2019.
- [211] G. Agamennoni, J. I. Nieto, and E. M. Nebot, “A bayesian approach for driving behavior inference,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 595–600, IEEE, 2011.
- [212] M. Brand, N. Oliver, and A. Pentland, “Coupled hidden markov models for complex action recognition,” in *Proceedings of IEEE computer society Proceedings of IEEE/CVF CVPR*, pp. 994–999, IEEE, 1997.

- [213] T. Gindele, S. Brechtel, and R. Dillmann, “A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments,” in *Proceedings of Intell. Transp. Syst. (ITSC)*, pp. 1625–1631, IEEE, 2010.
- [214] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, “Driver intent inference at urban intersections using the intelligent driver model,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 1162–1167, IEEE, 2012.
- [215] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.
- [216] S. Dai, L. Li, and Z. Li, “Modeling vehicle interactions via modified lstm models for trajectory prediction,” *IEEE Access*, vol. 7, pp. 38287–38296, 2019.
- [217] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, “Trafficpredict: Trajectory prediction for heterogeneous traffic-agents,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 6120–6127, 2019.
- [218] W. Ding and S. Shen, “Online vehicle trajectory prediction using policy anticipation network and optimization-based context reasoning,” in *Proceedings of IEEE ICRA*, pp. 9610–9616, IEEE, 2019.
- [219] M. Koschi and M. Althoff, “Set-based prediction of traffic participants considering occlusions and traffic rules,” *IEEE Trans. Intell. Vehicles*, vol. 6, no. 2, pp. 249–265, 2020.
- [220] W. Ding, J. Chen, and S. Shen, “Predicting vehicle behaviors over an extended horizon using behavior interaction network,” in *Proceedings of IEEE ICRA*, pp. 8634–8640, IEEE, 2019.
- [221] N. Deo and M. M. Trivedi, “Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 1179–1184, IEEE, 2018.
- [222] Z. Sheng, Y. Xu, S. Xue, and D. Li, “Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving,” *IEEE Trans. Intell. Transp. Syst.*, 2022.
- [223] X. Mo, Z. Huang, Y. Xing, and C. Lv, “Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network,” *IEEE Trans. Intell. Transp. Syst.*, 2022.
- [224] L. Shi, L. Wang, C. Long, S. Zhou, M. Zhou, Z. Niu, and G. Hua, “Sgcn: Sparse graph convolution network for pedestrian trajectory prediction,” in *Proceedings of the IEEE/CVF CVPR*, pp. 8994–9003, 2021.

- [225] D. Cao, J. Li, H. Ma, and M. Tomizuka, “Spectral temporal graph neural network for trajectory prediction,” in *Proceedings of IEEE ICRA*, pp. 1839–1845, IEEE, 2021.
- [226] S. Carrasco, D. F. Llorca, and M. A. Sotelo, “Scout: Socially-consistent and understandable graph attention network for trajectory prediction of vehicles and vrus,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 1501–1508, 2021.
- [227] H. Cheng, W. Liao, M. Y. Yang, B. Rosenhahn, and M. Sester, “Amenet: Attentive maps encoder network for trajectory prediction,” *ISPRS J. Photogramm. Remote Sens.*, vol. 172, pp. 253–266, 2021.
- [228] J. Li, F. Yang, M. Tomizuka, and C. Choi, “Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning,” *Adv. Neural Inf. Process.*, vol. 33, pp. 19783–19794, 2020.
- [229] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, and F. Nashashibi, “Non-local social pooling for vehicle trajectory prediction,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 975–980, 2019.
- [230] X. Li, X. Ying, and M. C. Chuah, “Grip: Graph-based interaction-aware trajectory prediction,” in *Proceedings of Intell. Transp. Syst. (ITSC)*, pp. 3960–3966, IEEE, 2019.
- [231] N. Deo and M. M. Trivedi, “Convolutional social pooling for vehicle trajectory prediction,” in *Proceedings of the IEEE/CVF CVPR Workshops*, pp. 1468–1476, 2018.
- [232] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in *Proceedings of IEEE ICRA*, pp. 4601–4607, IEEE, 2018.
- [233] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE/CVF CVPR*, pp. 961–971, 2016.
- [234] F. Diehl, T. Brunner, M. T. Le, and A. Knoll, “Graph neural networks for modelling traffic participant interaction,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 695–701, 2019.
- [235] Z. Sheng, Y. Xu, S. Xue, and D. Li, “Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17654–17665, 2022.
- [236] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, “Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories,” *arXiv preprint arXiv:1810.07225*, 2018.

- [237] X. Li, X. Ying, and M. C. Chuah, “Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving,” *arXiv preprint arXiv:1907.07792*, 2019.
- [238] P. Lv, W. Wang, Y. Wang, Y. Zhang, M. Xu, and C. Xu, “Ssagcn: Social soft attention graph convolution network for pedestrian trajectory prediction,” *IEEE Trans Neural Netw Learn Syst*., pp. 1–15, 2023.
- [239] L. F. Chiara, P. Coscia, S. Das, S. Calderara, R. Cucchiara, and L. Ballan, “Goal-driven self-attentive recurrent networks for trajectory prediction,” in *Proceedings of the IEEE/CVF CVPR*, pp. 2518–2527, 2022.
- [240] K. Mangalam, Y. An, H. Girase, and J. Malik, “From goals, waypoints & paths to long term human trajectory forecasting,” in *Proceedings of the IEEE/CVF ICCV*, pp. 15233–15242, 2021.
- [241] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, “Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction,” *arXiv preprint arXiv:1910.05449*, 2019.
- [242] J. Liang, L. Jiang, and A. Hauptmann, “Simaug: Learning robust representations from simulation for trajectory prediction,” in *Proceedings of IEEE/CVF ECCV*, pp. 275–292, Springer, 2020.
- [243] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, “Multi-agent tensor fusion for contextual trajectory prediction,” in *Proceedings of the IEEE/CVF CVPR*, pp. 12126–12134, 2019.
- [244] Y. Lu, W. Wang, X. Hu, P. Xu, S. Zhou, and M. Cai, “Vehicle trajectory prediction in connected environments via heterogeneous context-aware graph convolutional networks,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–13, 2022.
- [245] A. Bhattacharyya, M. Hanselmann, M. Fritz, B. Schiele, and C.-N. Straehle, “Conditional flow variational autoencoders for structured sequence prediction,” *arXiv preprint arXiv:1908.09008*, 2020.
- [246] P. Bhattacharyya, C. Huang, and K. Czarnecki, “Ssl-lanes: Self-supervised learning for motion forecasting in autonomous driving,” in *Conference on Robot Learning*, pp. 1793–1805, PMLR, 2023.
- [247] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, “Hivt: Hierarchical vector transformer for multi-agent motion prediction,” in *Proceedings of the IEEE/CVF CVPR*, pp. 8823–8833, 2022.

- [248] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, “Learning to predict vehicle trajectories with model-based planning,” in *Conference on Robot Learning*, pp. 1035–1045, PMLR, 2022.
- [249] V. Lefkopoulos, M. Menner, A. Domahidi, and M. N. Zeilinger, “Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 1, pp. 80–87, 2021.
- [250] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, “Probabilistic trajectory prediction with gaussian mixture models,” in *2012 IEEE IV*, pp. 141–146, 2012.
- [251] C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Yong, J.-D. Yoder, C. Tay, K. Mekhnacha, and A. Nègre, “Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety,” *IEEE Intell. Transp. Syst. Magazine*, vol. 3, no. 4, pp. 4–19, 2011.
- [252] T. Gu, G. Chen, J. Li, C. Lin, Y. Rao, J. Zhou, and J. Lu, “Stochastic trajectory prediction via motion indeterminacy diffusion,” in *Proceedings of the IEEE/CVF CVPR*, pp. 17113–17122, 2022.
- [253] O. Makansi, E. Ilg, O. Cicek, and T. Brox, “Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction,” in *Proceedings of the IEEE/CVF CVPR*, pp. 7144–7153, 2019.
- [254] 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,” in *IEEE IV*, pp. 165–170, IEEE, 2021.
- [255] T. Westny, J. Oskarsson, B. Olofsson, and E. Frisk, “Mtp-go: Graph-based probabilistic multi-agent trajectory prediction with neural odes,” *arXiv preprint arXiv:2302.00735*, 2023.
- [256] M. Fu, T. Zhang, W. Song, Y. Yang, and M. Wang, “Trajectory prediction-based local spatio-temporal navigation map for autonomous driving in dynamic highway environments,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–12, 2021.
- [257] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, “Gohome: Graph-oriented heatmap output for future motion estimation,” in *Proceedings of IEEE ICRA*, pp. 9107–9114, IEEE, 2022.
- [258] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, “Home: Heatmap output for future motion estimation,” in *Proceedings of Intell. Transp. Syst. (ITSC)*, pp. 500–507, IEEE, 2021.

- [259] W. Zhu, Y. Liu, M. Zhang, and Y. Yi, “Reciprocal consistency prediction network for multi-step human trajectory prediction,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–11, 2023.
- [260] F. Janjoš, M. Dolgov, and J. M. Zöllner, “Starnet: Joint action-space prediction with star graphs and implicit global-frame self-attention,” in *IEEE IV*, pp. 280–286, IEEE, 2022.
- [261] Z. Huang, X. Mo, and C. Lv, “Multi-modal motion prediction with transformer-based neural network for autonomous driving,” in *Proceedings of IEEE ICRA*, pp. 2605–2611, IEEE, 2022.
- [262] C. He, L. Chen, L. Xu, C. Yang, X. Liu, and B. Yang, “Irlsot: Inverse reinforcement learning for scene-oriented trajectory prediction,” *IET Intell. Transp. Syst.*, vol. 16, no. 6, pp. 769–781, 2022.
- [263] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data,” in *Proceedings of IEEE/CVF ECCV*, pp. 683–700, Springer, 2020.
- [264] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, “Sophie: An attentive gan for predicting paths compliant to social and physical constraints,” in *Proceedings of the IEEE/CVF CVPR*, pp. 1349–1358, 2019.
- [265] C. Tang and R. R. Salakhutdinov, “Multiple futures prediction,” *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
- [266] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, “Desire: Distant future prediction in dynamic scenes with interacting agents,” in *Proceedings of the IEEE/CVF CVPR*, pp. 336–345, 2017.
- [267] C. Saiprasert, T. Pholprasit, and S. Thajchayapong, “Detection of driving events using sensory data on smartphone,” *J Intell. Transp. Syst Res.*, vol. 15, no. 1, pp. 17–28, 2017.
- [268] J. Xie, A. R. Hilal, and D. Kulić, “Driving maneuver classification: A comparison of feature extraction methods,” *IEEE Sens. J.*, vol. 18, no. 12, pp. 4777–4784, 2018.
- [269] N. Li, T. Misu, A. Tawari, A. Miranda, C. Suga, and K. Fujimura, “Driving maneuver prediction using car sensor and driver physiological signals,” in *ACM Conf. Multimod. Interac.*, pp. 108–112, 2016.

- [270] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, “Automec: Lstm-based user mobility prediction for service management in distributed mec resources,” in *Proceedings of ACM Conference MSWiM*, pp. 155–159, 2020.
- [271] C. Ou and F. Karray, “Deep learning-based driving maneuver prediction system,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1328–1340, 2020.
- [272] B. Häfner, J. Jiru, H. Schepker, G. A. Schmitt, and J. Ott, “Proposing cooperative maneuvers among automated vehicles using machine learning,” in *Proceedings of ACM Conference MSWiM*, pp. 173–180, 2021.
- [273] Y. Xing, C. Lv, H. Wang, D. Cao, and E. Velenis, “An ensemble deep learning approach for driver lane change intention inference,” *Transp. Res. Part C Emerg. Technol.*, vol. 115, p. 102615, 2020.
- [274] B. Häfner, J. Jiru, H. Schepker, G. A. Schmitt, and J. Ott, “Preventing failures of cooperative maneuvers among connected and automated vehicles,” in *Proceedings of ACM Conference MSWiM*, pp. 5–12, 2021.
- [275] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, “Exploiting map information for driver intention estimation at road intersections,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 583–588, IEEE, 2011.
- [276] V. Gadepally, A. Krishnamurthy, and U. Ozguner, “A framework for estimating driver decisions near intersections,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 637–646, 2013.
- [277] M. B. Younes and A. Boukerche, “Toward a smooth vehicular traffic at round road - intersections§,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2018.
- [278] A. Bender, J. R. Ward, S. Worrall, and E. M. Nebot, “Predicting driver intent from models of naturalistic driving,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 1609–1615, 2015.
- [279] A. Zyner, S. Worrall, J. Ward, and E. Nebot, “Long short term memory for driver intent prediction,” in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, pp. 1484–1489, 2017.
- [280] D. Yi, J. Su, C. Liu, and W.-H. Chen, “Trajectory clustering aided personalized driver intention prediction for intelligent vehicles,” *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3693–3702, 2019.

- [281] F. Martinelli, F. Mercaldo, A. Orlando, V. Nardone, A. Santone, and A. K. Sangaiah, "Human behavior characterization for driving style recognition in vehicle system," *Computers & Electrical Engineering*, vol. 83, p. 102504, 2020.
- [282] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.
- [283] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE access*, vol. 6, pp. 1662–1669, 2017.
- [284] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [285] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, pp. 3111–3119, 2013.
- [286] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Advances in Neural Information Processing Systems*, pp. 4650–4661, 2019.
- [287] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *J. Mach. Learn. Res.*, vol. 9, no. 11, 2008.
- [288] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of ICLR*, 2016.
- [289] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Proceedings of IEEE/CVF CVPR*, pp. 770–778, 2016.
- [290] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.
- [291] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*, pp. 1578–1585, IEEE, 2017.
- [292] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.

- [293] J. L. Elman, “Distributed representations, simple recurrent networks, and grammatical structure,” *Machine learning*, vol. 7, pp. 195–225, 1991.
- [294] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [295] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv preprint arXiv:1409.1259*, 2014.
- [296] D. Long, R. Zhang, and Y. Mao, “Prototypical recurrent unit,” *Neurocomputing*, vol. 311, pp. 146–154, 2018.
- [297] Z. Zhao, L. Zhou, Y. Luo, and K. Li, “Emergency steering evasion assistance control based on driving behavior analysis,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 457–475, 2019.
- [298] C. You, J. Lu, and P. Tsiotras, “Nonlinear driver parameter estimation and driver steering behavior analysis for adas using field test data,” *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 5, pp. 686–699, 2017.
- [299] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, “Temporal convolutional networks: A unified approach to action segmentation,” in *Proceedings of IEEE/CVF ECCV*, pp. 47–54, Springer, 2016.
- [300] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [301] N. Aljeri and A. Boukerche, “A distributed topology discovery protocol for software-defined vehicular networks,” in *Proceedings of the 17th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN)*, pp. 17–24, 2020.
- [302] P. Sun and A. Boukerche, “Challenges and potential solutions for designing a practical pedestrian detection framework for supporting autonomous driving,” in *ACM MobiWac*, p. 75–82, ACM, 2020.
- [303] N. Aljeri and A. Boukerche, “An adaptive traffic-flow based controller deployment scheme for software-defined vehicular networks,” in *Proceedings of ACM Conference MSWiM*, pp. 191–198, 2020.

- [304] C. Celes, A. Boukerche, and A. A. F. Loureiro, “Generating and analyzing mobility traces for bus-based intelligent vehicular networks,” *IEEE Trans. Veh. Technol.*, pp. 1–16, 2023.
- [305] A. Zyner, S. Worrall, and E. Nebot, “A recurrent neural network solution for predicting driver intention at unsignalized intersections,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1759–1764, 2018.
- [306] H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, Y. Yamamoto, A. Yamashita, and H. Asama, “Lane-change detection based on vehicle-trajectory prediction,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1109–1116, 2017.
- [307] S. Klingelschmitt, M. Platho, H.-M. Groß, V. Willert, and J. Eggert, “Combining behavior and situation information for reliably estimating multiple intentions,” in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 388–393, IEEE, 2014.
- [308] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv preprint arXiv:1803.01271*, 2018.
- [309] N. Aljeri and A. Boukerche, “An efficient heuristic switch migration scheme for software-defined vehicular networks,” *JPDC*, vol. 164, pp. 96–105, 2022.
- [310] C. Celes, A. Boukerche, and A. A. Loureiro, “From mobility traces to knowledge: Design guidance for intelligent vehicular networks,” *IEEE Netw.*, 2020.
- [311] N. Aljeri and A. Boukerche, “Advice-loc: An adaptive vehicle-centric location management scheme for intelligent connected cars,” *Ad Hoc Netw.*, vol. 107, p. 102223, 2020.
- [312] R. S. de Sousa, A. Boukerche, and A. A. F. Loureiro, “A novel scalable framework to reconstruct vehicular trajectories from unreliable gps datasets,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–0, 2023.
- [313] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” *arXiv preprint arXiv:1709.04875*, 2017.
- [314] M. S. Shirazi and B. T. Morris, “Trajectory prediction of vehicles turning at intersections using deep neural networks,” *Mach. Vis. Appl.*, vol. 30, no. 6, pp. 1097–1109, 2019.
- [315] M. Sha and A. Boukerche, “Performance evaluation of cnn-based pedestrian detectors for autonomous vehicles,” *Ad Hoc Networks*, vol. 128, p. 102784, 2022.
- [316] X. Zhang, H. Chen, W. Yang, W. Jin, and W. Zhu, “Pedestrian path prediction for autonomous driving at un-signalized crosswalk using w/cdm and msfm,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 3025–3037, 2021.

- [317] Z. Huang, P. Sun, and A. Boukerche, "Towards a practical pedestrian detection method for supporting autonomous driving," in *ICC 2022 - IEEE International Conference on Communications*, pp. 1506–1511, 2022.
- [318] L. Chen, S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu, and F.-Y. Wang, "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, 2021.
- [319] C. Celes, A. Boukerche, and A. A. F. Loureiro, "On the design of bus-based vehicular networks: Mobility generation and data dissemination," in *2022 IEEE ICC*, pp. 1512–1517, 2022.
- [320] M. B. Younes and A. Boukerche, "Towards a sustainable highway road-based driving protocol for connected and self-driving vehicles," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 1, pp. 235–247, 2022.
- [321] M. B. Younes and A. Boukerche, "A novel traffic characteristics aware and context prediction protocol for intelligent connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 9897–9908, 2023.
- [322] A. Boukerche, K. R. L. Jucá, J. B. Sobral, and M. S. M. A. Notare, "An artificial immune based intrusion detection model for computer and telecommunication systems," *Parallel Computing*, vol. 30, no. 5-6, pp. 629–646, 2004.
- [323] A. Boukerche, R. B. Machado, K. R. Jucá, J. B. M. Sobral, and M. S. Notare, "An agent based and biological inspired real-time intrusion detection and security model for computer network operations," *Computer Communications*, vol. 30, no. 13, pp. 2649–2660, 2007.
- [324] A. Boukerche and M. S. M. A. Notare, "Neural fraud detection in mobile phone operations," in *Parallel and Distributed Processing: 15 IPDPS 2000 Workshops Cancun, Mexico, May 1–5, 2000 Proceedings 14*, pp. 636–644, Springer, 2000.
- [325] N. Deo, A. Ranges, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Trans. Intell. Veh.*, vol. 3, no. 2, pp. 129–140, 2018.
- [326] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *Neurocomputing*, vol. 399, pp. 491–501, 2020.
- [327] N. Radwan, W. Burgard, and A. Valada, "Multimodal interaction-aware motion prediction for autonomous street crossing," *Int J Rob Res*, vol. 39, no. 13, pp. 1567–1598, 2020.

- [328] M.-P. Dubuisson and A. Jain, “A modified hausdorff distance for object matching,” in *Proc. of 12th int. conf. on Pattern Recognition*, vol. 1, pp. 566–568 vol.1, 1994.
- [329] F. Altché and A. de La Fortelle, “An lstm network for highway trajectory prediction,” in *Proceedings of Intell. Transp. Syst. (ITSC)*, pp. 353–359, IEEE, 2017.
- [330] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, “Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4883–4894, 2019.
- [331] J. Kim, T. Kim, S. Kim, and C. D. Yoo, “Edge-labeling graph neural network for few-shot learning,” in *Proceedings of the IEEE/CVF Proceedings of IEEE/CVF CVPR*, pp. 11–20, 2019.
- [332] X. Zhang, C. Xu, X. Tian, and D. Tao, “Graph edge convolutional neural networks for skeleton-based action recognition,” *IEEE Trans. Neural. Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3047–3060, 2019.
- [333] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [334] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.
- [335] C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy, “Stochastic prediction of multi-agent interactions from partial observations,” *arXiv preprint arXiv:1902.09641*, 2019.
- [336] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, “Implicit latent variable model for scene-consistent motion forecasting,” in *Proceedings of IEEE/CVF ECCV*, pp. 624–641, Springer, 2020.
- [337] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, “Maximum entropy inverse reinforcement learning,” in *AAAI*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.
- [338] E. Jang, S. Gu, and B. Poole, “Categorical reparametrization with gumble-softmax,” in *Proceedings of ICLR*, OpenReview. net, 2017.
- [339] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, “Multimodal motion prediction with stacked transformers,” in *Proceedings of the IEEE/CVF CVPR*, pp. 7577–7586, 2021.
- [340] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *Proceedings of ICML*, pp. 6105–6114, PMLR, 2019.