Deep learning based trajectory prediction for autonomous vehicles

Kaouther Messaoud

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Kaouther Messaoud

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITE

Sujet de la thèse :

Deep Learning based Trajectory Prediction for Autonomous Vehicles

---

Membres du jury:

M. Alexandre ALAHI	Rapporteur
M. Fabien MOUTARDE	Rapporteur
M. David FILLIAT	Examinateur
M. Guillaume SANDOU	Examinateur
M. Mohan TRIVEDI	Examinateur
Mme. Anne VERROUST-BLONDET	Directrice de thèse
M. Fawzi NASHASHIBI	Co-Directeur de thèse
Mme. Itheri YAHIAOUI	Encadrante

Acknowledgements

First, I would like to thank my PhD directors, *Fawzi NASHASHIBI* and *Anne VERROUST-BLONDET*, for welcoming me into their research team and for offering me the opportunity of pursuing PhD studies in the autonomous vehicle and machine learning research domains. Moreover, I am grateful to them and to *Itheri YAHIAOUI* for their valuable advice and support when discussing my raw ideas and working on new papers, which helped me to move forward and accomplish my PhD.

In addition, I would like to thank *Prof. Alexandre ALAHI* and *Prof. Fabien MOUTARDE* for accepting to review my thesis. I also express my gratitude to the rest of my PhD examining committee: *Prof. David FILLIAT*, *Prof. Guillaume SANDOOU* and *Prof. Mohan Trivedi* for their time and interest in my work. I would like to extend special thanks to *Prof. Trivedi* and all the members of the Laboratory for Intelligent and Safe Automobiles (LISA) for welcoming me three months during my thesis at the University of California at San Diego and for helping me find a great environment to work. It was a real honor to work under his supervision and to absorb some of his research methodology. Moreover, the discussions with the members of the team were very enriching and allowed me to acquire scientific maturity.

Then, I would like to thank my colleagues and all members of RITS team at INRIA Paris. Our every day discussions allowed me to extend my scientific knowledge and to find and refine new ideas. They also helped me to improve my language skills. We spent real pleasant moments together. I extend my thanks to all the people I met at INRIA Paris during my PhD who made my journey more beneficial and amusing. Huge thanks go to *Richard JAMES* for his help in reviewing some parts of my thesis and papers.

Finally, I would like to thank my parents, sister and brother for their continuous encouragement and all their sacrifices that allowed me to attain a good educational level and to accomplish this achievement. My heartfelt thanks go to my husband. Actually, the life of a couple who are PhD students can sometimes be complicated with the pressures of work and consecutive deadlines, but he made it pleasant through his endless care, patience, support and love.

Abstract

Trajectory prediction of surrounding agents is essential for autonomous driving in order to effectively perform tactical path planning. In the thesis, we tackle the task of trajectory prediction of a target vehicle in two different environments: one on a highway, and the other in an urban setting. For this purpose, we develop deep learning-based solutions using different input features. First, the target vehicle's past trajectory reveals information about the direction of its short-term future motion and the possible speed change range. Therefore, our solutions exploit this information and model the temporal evolution of the target agent using recurrent neural networks. In addition, the future trajectory of an agent in the scene depends on its interaction with the surrounding agents and the static scene. Thus, our first studies model the influence of the surrounding dynamic agents in a highway environment using two different methods based on relational recurrent neural networks [69] and a multi-head attention mechanism [68, 71]. Since autonomous vehicles face more complex scenarios (intersections, roundabouts) in an urban environment, additional information about the static scene context (road geometry, lane structure...) is required. Therefore, we focus on leveraging cues from the static scene using high fidelity maps in our subsequent work [70]. In addition, in order to account for the uncertainty of the future, we generate multiple predicted plausible trajectories and the probability of occurrence of each one. We also ensure that the predicted trajectories are realistic and compliant with the scene structure, which means that the predicted positions lie within the drivable area. The developed solutions are trained and evaluated using the naturalistic driving datasets NGSIM [44, 45], highD [55] and nuScenes [13].

List of Publications

1. Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. Non-local social pooling for vehicle trajectory prediction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 975–980, 2019a.

2. Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. Relational recurrent neural networks for vehicle trajectory prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1813–1818, 2019b.

3. Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. Attention based vehicle trajectory prediction. *IEEE Transactions on Intelligent Vehicles*, April 2020a.

4. Kaouther MESSAOUD, Nachiket DEO, Mohan M. TRIVEDI and Fawzi NASHASHIBI. "Trajectory Prediction for Autonomous Driving based on Multi-Head Attention with Joint Agent-Map Representation". In *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.

Contents

List of Publications **vii**

List of Figures **xv**

List of Tables **xvii**

List of Abbreviations **xix**

1 Introduction **1**
1.1 Context and Motivation 2
1.2 Problem Description and Objectives 3
1.3 Thesis Outline 6

2 State-Of-The Art **9**
2.1 Overall Motion Prediction Framework 10
2.1.1 Stimuli 10
2.1.1.1 Target agent features 10
2.1.1.2 Environment features 11
2.1.2 Modeling approach 12
2.1.3 Prediction 12
2.1.3.1 Output Forms 12
2.1.3.2 Multi-Modal Output 13
2.2 Classical Methods for Trajectory Prediction 15
2.2.1 Physics-based methods 15
2.2.2 Pattern-based methods 16
2.2.3 Planning-based methods 17
2.3 Deep Learning for Motion Prediction 18
2.3.1 Different Architectures of Sequence Modeling and Prediction . 19
2.3.1.1 Recurrent Sequence-to-Sequence Model 19
2.3.1.2 Convolutional Neural Networks for Trajectory Prediction 21
2.3.1.3 Generative Models for Trajectory Prediction 24
Generative adversarial networks 24
Variational Autoencoders 25

2.3.1.4 Attention Mechanism for Trajectory Prediction 27
2.3.2 Agent Environment Interactions 29
2.3.2.1 Implicit Interaction Modeling 29
2.3.2.2 Explicit Interaction Modeling 29
Social Pooling 29
Convolutional Neural Networks for Interaction Modeling 29
Attention Mechanism for Interaction Modeling 30
Graph Neural Networks 31
2.4 Summary of Trajectory Prediction Deep Learning based Studies 32
2.5 Datasets 35
2.5.1 Next Generation Simulation (NGSIM) 36
2.5.2 highD 37
2.5.3 nuScenes 37

3 Interaction Aware Trajectory Prediction on Highways 39
3.1 Problem Definition 41
3.2 Overview 41
3.2.1 LSTM Encoder-Decoder 42
3.2.2 Multi-Head Attention Mechanism 44
3.2.3 Brief Description of our Proposed Methods 47
3.2.4 Related Work 47
3.2.4.1 Social LSTM 48
3.2.4.2 Convolutional Social Pooling 49
3.3 Relational Recurrent Neural Networks based Methods (L-RRNN and Sc-RRNN) 50
3.3.1 Overall Model 50
3.3.2 Inputs Representation 51
3.3.3 Inputs Embedding 52
3.3.3.1 Scene embedding (Sc-RRNN) 52
3.3.3.2 Per lane representation (L-RRNN) 52
3.3.4 Relational Recurrent Encoder-Decoder 53
3.3.5 Multi-Head Dot Product Attention (MHDPA): 53
3.3.6 Introducing MHA into an LSTM 54
3.3.7 Implementation Details 55
3.4 Multi-head Attention based Method (MHA-LSTM) 55
3.4.1 Overall Model 56
3.4.2 Trajectory Encoder 57
3.4.3 Vehicle Interaction Modules 57
3.4.3.1 α-Attention 58

3.4.3.2 Dot-Product Attention . . . 59
3.4.3.3 Concatenation Attention . . . 59
3.4.4 High Order Interaction . . . 59
3.4.5 Trajectory Generation . . . 60
3.4.6 Another Variant: Local and Non Local Social Pooling . . . 60
3.4.6.1 Convolution Layer . . . 61
3.4.6.2 Non-local Multi-head Attention Mechanism . . . 61
3.4.7 Implementation Details . . . 62
3.5 Experimental Evaluations . . . 62
3.5.1 Evaluation Metric . . . 62
3.5.2 Models Compared . . . 63
3.5.3 Quantitative evaluation . . . 64
3.5.3.1 Overall evaluation of RRNN . . . 64
3.5.3.2 Overall evaluation of MHA-LSTM . . . 65
3.5.3.3 Effects of using multiple attention heads . . . 66
3.5.3.4 Comparison of attention methods . . . 67
3.5.3.5 Error evaluation per lane change . . . 67
3.5.4 Qualitative Analysis of Predictions . . . 68
3.6 Conclusion . . . 71

4 Scene Aware Trajectory Prediction: A Recurrent Approach 73
4.1 Multi-head Attention with Joint Agent Map Representation (MHA-JAM) 75
4.1.1 Input Representation . . . 76
4.1.2 Multimodal Output . . . 76
4.1.3 Encoding Layer . . . 78
4.1.4 Separate Agent-Map plus Joint Conditioning MHA . . . 78
4.1.4.1 Separate Attention: . . . 78
4.1.4.2 Joint Conditioning Attention: . . . 79
4.1.5 Joint Agent-Map Attention . . . 80
4.1.6 Decoding Layer . . . 82
4.1.7 Intention-based Prediction . . . 82
4.1.7.1 Goal directed Trajectory Generation . . . 83
4.1.7.2 Anchor based Trajectory Generation . . . 83
4.1.7.3 Region based Trajectory Generation . . . 84
4.1.8 Dynamically-feasible Prediction . . . 85
4.1.8.1 Dynamically-extended Unicycle Model . . . 85
4.1.8.2 Output Form . . . 86
Mean Positions . . . 86
Covariance of the Positions . . . 86

4.1.9 Loss Functions . . . 87
4.1.9.1 Regression loss . . . 87
4.1.9.2 Classification loss [21] . . . 87
4.1.9.3 Off-road loss . . . 87
4.1.9.4 Diversity loss . . . 88
4.2 Double Attention Based Model . . . 89
4.2.1 Input and Output Representation . . . 89
4.2.2 Encoding and Decoding layers . . . 89
4.2.3 Multi-head Double Attention . . . 89
4.2.4 Double Attention for Scene Interaction (DA-JAM) . . . 90
4.2.4.1 Global features extractions . . . 91
4.2.4.2 Adaptive features distribution . . . 91
4.3 Simultaneous multi-vehicle trajectory prediction . . . 92
4.3.1 Problem Definition . . . 92
4.3.2 Multi-Head Attention for simultaneous multi-vehicle trajectory prediction . . . 93
4.3.3 Multi-Head Double Attention for simultaneous multi-vehicle trajectory prediction . . . 93
4.4 Experimental Analysis and Evaluations . . . 94
4.4.1 Training and Implementation Details . . . 95
4.4.2 Evaluation Metrics . . . 95
4.4.3 Models Compared . . . 96
4.4.3.1 Baselines . . . 96
4.4.3.2 Our proposed Methods . . . 98
4.4.4 Quantitative Evaluation . . . 99
4.4.5 Ablation Experiments . . . 100
4.4.5.1 Importance of input cues . . . 102
4.4.5.2 Advantage of using multiple attention heads . . . 102
4.4.5.3 Effectiveness of joint context representation . . . 102
4.4.5.4 Effectiveness of a specialized attention heads: . . . 103
4.4.5.5 Role of the Off-road loss: . . . 103
4.4.6 Qualitative Evaluation . . . 103
4.4.6.1 MHA-JAM Method . . . 103
4.4.6.2 Intention-based Prediction . . . 105
4.5 Conclusion . . . 108

5 Scene Aware Trajectory Prediction: A Non Recurrent Approach 111
5.1 Transformers and LSTMs for Trajectory Prediction . . . 112
5.2 Multi-Modal Transformer for Trajectory Prediction . . . 113
5.2.1 Model Inputs . . . 113
5.2.2 Model Output . . . 114
5.2.3 Positional Encoding . . . 115
5.2.4 Encoding Layer . . . 116
The trajectory encoding module: . . . 116
The scene feature extractor module: . . . 117
5.2.5 Decoding Layer . . . 117
5.2.5.1 Non Auto-regressive Multimodal Trajectory Generation 118
5.2.5.2 Full Decoder Self-Attention . . . 119
5.2.5.3 Map-Decoder MHA Layer . . . 120
5.2.6 Transformer Refinement . . . 120
5.2.6.1 Latent variable model . . . 120
5.2.6.2 Deterministic Approximation . . . 121
5.2.6.3 Iterative Refinement . . . 121
5.2.7 Classification Layer . . . 122
5.2.8 Multimodal Network Architecture . . . 122
5.2.8.1 Prediction . . . 122
5.2.9 Loss Functions . . . 123
5.2.9.1 Refinement loss . . . 123
5.2.9.2 Total loss . . . 123
5.3 Experimental Analysis and Evaluations . . . 123
5.3.1 Training and Implementation Details . . . 123
5.3.2 Models Compared . . . 124
5.3.3 Quantitative Evaluation . . . 124
5.3.4 Ablation Experiments . . . 124
5.3.4.1 Importance of the refinement approach . . . 125
5.3.4.2 Importance of map information . . . 125
5.3.5 Qualitative Evaluation . . . 125
5.4 Conclusion . . . 127

6 Conclusion and Perspective 129
6.1 Contributions . . . 130
6.2 Discussion . . . 131
6.3 Research Perspectives . . . 132

References 135

List of Figures

1.1 Automated Driving Pipeline [42] 2
1.2 Example of trajectory prediction scenario [42] 4

2.1 Rasterized representation of the static and dynamic scene elements . . . 12
2.2 Comparison of single and multimodal vehicle trajectories predictions . . 13
2.3 Probabilistic multimodal trajectory prediction 14
2.4 Recurrent Network A with input sequence x_t and outputs h_t 20
2.5 RNN encoder-decoder 20
2.6 CNN Layers 22
2.7 Example of GAN based architecture [35] 24
2.8 VAE architecture 26
2.9 The Transformer model architecture [107] 28
2.10 Example of spatio-temporal graph [110] 31
2.11 NGSIM dataset 36
2.12 Highway drone dataset highD [55] 37
2.13 nuScenes [13] dataset 38

3.1 Regular LSTM 42
3.2 LSTM encoder decoder [76] 44
3.3 Computation of One Attention Head Explained 44
3.4 An example of computing MHA using two inputs X and R 45
3.5 Graph Attention Network 46
3.6 Basic LSTM Encoder Decoder Architecture 47
3.7 RRNN Architecture 47
3.8 MHA Pooling based Architecture 48
3.9 Social LSTM [1] 48
3.10 Convolutional Social Pooling [23] 49
3.11 Per Lane Scene Representation RNN (L-RRNN) Model 54
3.12 MHA Pooling based Model 58
3.13 Attention maps for two different lane change maneuvers 69
3.14 Average attention maps for two different lane change maneuvers 71

4.1 MHA with Joint Agent Map representation based Model 77

4.2 Separate agent-map representation for the attention heads 79
4.3 Attention modules in MHA-JAM model 80
4.4 Interaction space Division into distinct Destination regions 83
4.5 Anchor Trajectories 84
4.6 Example of Off-road loss function 88
4.7 Double attention based model with joint agent and map representation . 90
4.8 Double attention based model with joint agent and map representation for simultaneous multiple trajectory prediction 94
4.9 MTP network architecture 97
4.10 CoverNet Trajectory Set 98
4.11 Ablation experiments of MHA-JAM model 101
4.12 Examples of attention maps and trajectories with MHA-JAM (off-road) 104
4.13 Visualisation of average attention maps over different generated maneuvers. 104
4.14 Visualisation of average attention maps and predicted trajectories over different generated maneuvers 106
4.15 Visualisation of average attention maps and predicted trajectories when considering different regions of attention 107
4.16 Visualisation of average attention maps of our basic MHA-JAM method 108
4.17 Visualisation of average attention maps when using the diversity loss . . 108
4.18 Visualisation of trajectory prediction near an intersection using different MHA-JAM based methods 109

5.1 Spatio-Temporal Multimodal Transformer **SF-M-TF** 114
5.2 Regular Auto-regressive Transformer 118
5.3 Causal vs Full self-attention 119
5.4 Spatio-Temporal Multimodal Transformer with refinement (**ST-M-TF-R**) 121
5.5 Visualisation of predicted trajectories using different transformer models 126

List of Tables

2.1 Summary of Trajectory Prediction Approaches 33
2.2 Overview of the motion trajectories datasets 35

3.1 Root mean squared prediction error (RMSE) in meters over a 5 second prediction horizon for the models using highD dataset 64
3.2 RMSE in meters over a 5-second prediction horizon for the models . . . 66
3.3 RMSE in meters over a 5-second prediction horizon for different numbers of attention heads on the highD dataset 67
3.4 RMSE in meters over a 5-second prediction horizon for different attention operations on the highD dataset 67
3.5 Longitudinal and lateral errors in meters over a 5-second prediction horizon for different maneuvers on HighD dataset 68
3.6 Neighbor vehicles states. 69

4.1 Results of comparative analysis on nuScenes dataset, over a prediction horizon of 6-seconds 99
4.2 MinADE and MinFDE with different numbers of attention heads (L) . . 102
4.3 Results of comparative analysis on nuScenes dataset, over a prediction horizon of 6-seconds 105
4.4 Results of comparative analysis of intension based methods on nuScenes dataset, over a prediction horizon of 6-seconds using 3 prediction modes 105

5.1 Results of comparative analysis on nuScenes dataset, over a prediction horizon of 6 seconds 123

List of Abbreviations

AV	Autonomous Vehicle.
BERT	Bidirectional Encoder Representations from Transformer.
BEV	Bird's Eye View.
CF-VAE	Conditional Flow Variational Auto-Encoder.
ConvLSTM	Convolutional LSTM.
CV	Constant Velocity.
CVAE	Conditional Variational Auto-Encoder.
DA	Double Attention.
DL	Deep Learning.
FC	Fully Connected.
GA	Global Attention.
GAN	Generative Adversarial Network.
GAT	Graph Attention Network.
GNN	Graph Neural Network.
GRU	Gated Recurrent Unit.
HMM	Hidden Markov Model.
IMM	Interacting Multiple Model.
IRL	Inverse Reinforcement Learning.
LF	Lane Following.
LLC	Left Lane Change.
LSTM	Long Short-Term Memory.
MDP	Markov Decision Process.
MHA	Multi-Head Attention.
MHA-JAM	MHA with Joint Agent and Map.
MHA-JAM (JAH)	MHA-JAM with Joint-Attention-Heads.

MHA-SAM . . . MHA with Separate Agent and Map.

MHDPA . . . . . Multi-Head Dot Product Attention.

MTP . . . . . Multiple-Trajectory Prediction.

NLL . . . . . Negative Log-Likelihood.

NLP . . . . . Natural Language Processing.

NN . . . . . Neural Network.

RLC . . . . . Right Lane Change.

RMC . . . . . Relational Memory Core.

RMSE . . . . . Root of the Mean Squared Error.

RN . . . . . Relational Network.

RNN . . . . . Recurrent Neural Network.

RRNN . . . . . Relational Recurrent Neural Network.

SAE . . . . . Society of Automotive Engineers.

STAR . . . . . Spatio-Temporal grAph tRansformer.

S-LSTM . . . . . Social LSTM.

TF . . . . . Transformer.

VAE . . . . . Variational Auto-Encoder.

WAE . . . . . Wasserstein Auto-Encoder.

1 Introduction

Contents

1.1 Context and Motivation **2**
1.2 Problem Description and Objectives **3**
1.3 Thesis Outline **6**

1.1 Context and Motivation

During recent decades, automobile manufacturers have been consistently working on improving the driving experience and making road vehicles safer by developing driver assistance technologies. In order to evaluate the extent of advances in driver assistance technology, six levels of autonomy have been defined by the Society of Automotive Engineers (SAE). These levels range from 0, which corresponds to fully manual driving, to 5 the fully autonomous which is the ultimate goal of the recent research made both by the automotive industry and institutions. Today's transportation systems have intermediate autonomy levels going from the ability to control both steering and accelerating/decelerating, to environmental detection and decision making capabilities, to intervening in critical situations. However, they have significant room for improvement in order to reach the full automation level.

To better understand the technological challenges facing automated driving, we introduce the building blocks of autonomous driving systems: a perception module, a prediction module and a planning module (cf. Figure 1.1).

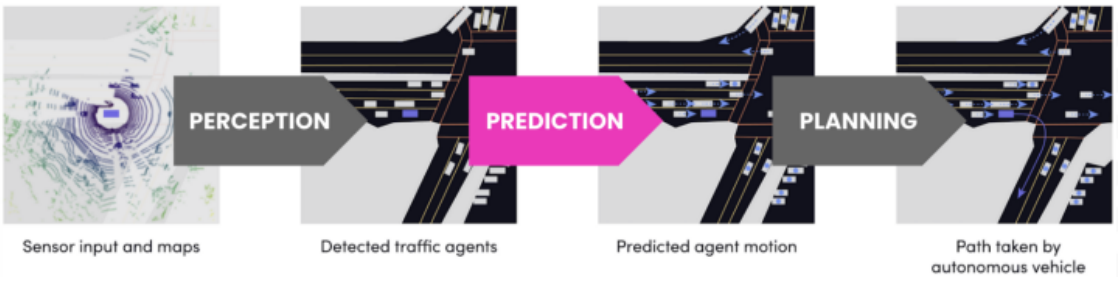


Figure 1.1: Automated Driving Pipeline [42]

In order to perceive the surrounding environment, multiple exteroceptive sensors such as cameras, LiDARs, radars and ultrasonic sensors are mounted on the vehicle to scan the environment. The perception module examines the raw sensor data, detects and identifies the static and dynamic scene objects; the different parts of the road structure (drivable area, lanes etc.) and the other road users (cars, cyclists, pedestrians, etc.). Once the perception task is completed, it outputs an estimation of the location of each detected object. For a fully automated driving in complex urban scenes or high speed scenarios, estimating the current location alone is insufficient to be able to interact with those objects. Therefore, a prediction module is required to forecast the future behaviors of the surrounding agents, i.e. future maneuvers or trajectories. Based on these predictions and on the final destination, the appropriate ego trajectory is planned and translated to the actuators.

Each of the previously described modules requires further research and development in order to meet the safety requirements of advanced driving autonomy levels. In particular,

predicting the future behavior of agents surrounding an Autonomous Vehicle (AV) is one of the main open challenges in order to reach a fully automated driving. Forecasting the future motion of nearby road users is crucial to understand the surrounding interactions and make reasonable decisions. However, the uncertainty of the future and the variety of human behavior in response to different situations are challenging for an AV, especially since the traffic rules are not always respected.

On the other hand, the recent advances in sensor modalities and technologies have significantly contributed to the enhancement of the quality and availability of measurement and information for AVs. In addition, the increased performance of embedded system architectures and softwares enhances their computational power and their effectiveness to meet real-time driving automation. It also allows the use of complex deep learning architectures. Deep learning has produced great achievements in numerous fields such as computer vision, Natural Language Processing (NLP) and time series modeling and prediction. These recent technological and scientific advances have already promoted today's successes of autonomous driving to a notable extent. Nonetheless, further research is essential to address the remaining challenges.

1.2 Problem Description and Objectives

AVs are presumed to navigate in highly uncertain and interactive environments shared with other dynamic agents. For an efficient integration of AVs on roads, human-like reasoning and decision making in complex traffic situations is necessary. Human drivers make decisions based on their implicit reasoning about how surrounding drivers will behave in the future. As communication between vehicles is not always possible, AVs must perceive and anticipate the intentions of surrounding vehicles in order to plan comfortable proactive motions and avoid urgent reactive decisions and conflicts with others. In fact, motion prediction helps AVs understand possible future situations and decide about a future behavior that minimizes the possible risks accordingly.

Figure 1.2 gives an example that shows the importance of motion prediction; An AV wants to turn left while another car is approaching from the opposite direction. The AV needs to predict the possible trajectories of the other car which are turning right or continue straight on and infer the likelihood of each trajectory. Indeed, if the car continues straight on it could interfere with the AV's left turn. In order for the AV to perform this maneuver safely, the AV should evaluate the possibility of interference and plan its motion accordingly.

In this thesis, we tackle the task of trajectory prediction in different environments (highway, urban) by introducing various deep learning-based architectures. The first

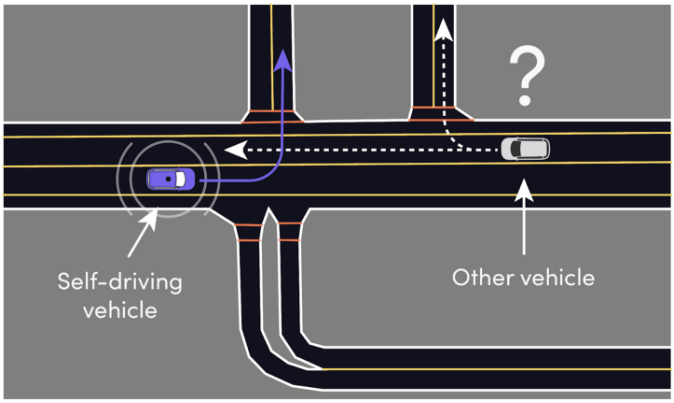


Figure 1.2: Example of trajectory prediction scenario [42]

step here is to select the cues that provide the useful information to infer the future behavior of a target vehicle. The target vehicles' past states give relevant information about the direction of its short-term future motion and the possible speed change range. However, the trajectory taken by each vehicle in the future is not only dependent on its own state history, the surrounding environment determines the possible future motion as well. Indeed, the presence and actions of the neighboring vehicles have a great influence on a vehicle's behavior. For example, if a vehicle v_1 proceeding a vehicle v_2 in the same lane decelerates, the vehicle v_2 should at some point react to this change by adjusting its speed to maintain the safety distance or performing a lane change if possible. In addition, when we consider an urban environment, the static scene structure including road and lane structure, sidewalks and crosswalks present important cues that determine the possible ways and the general patterns.

The second challenge is to design an adequate Neural Network (NN) that aggregates the relevant features from the input cues and models the dependencies between them. Generally, to address the task of trajectory prediction, the first task is to model the dynamics and the temporal dependencies of the past motion of the agents in the scene. Second, the interactions between the neighboring vehicles should be analyzed in order to retrieve information about the social context that influences the target agent's future trajectory. To do so, a human-like reasoning consists on paying selective attention to a subset of surrounding vehicles that most influence the target vehicle's future trajectory while paying less attention to other vehicles. For example, a vehicle performing a lane change maneuver will pay more attention to the vehicles in the target lane than those in the other lanes. Consequently, its future behavior could be more dependent on distant vehicles in the target lane than the close ones in the other lanes.

A major challenge in trajectory prediction is the high variability in the two context cues; the static scene elements and the agents in a traffic scene can have various

configurations. These two cues are tightly linked; each can inform the most salient parts of the other. For example, the presence of a nearby pedestrian could make a crosswalk in the path of the vehicle of interest a salient part of the static scene, as opposed to if there were no nearby pedestrians. On the other hand, the presence of a crosswalk would make a nearby pedestrian on the sidewalk a more salient part of the input context as opposed to if there was no crosswalk. This dependency between the two cues should be considered when addressing the trajectory prediction task.

Another challenge in trajectory prediction is the multimodality of the distribution of future trajectories. The inherent uncertainty of the future imposes that there is not one single correct possible future trajectory prediction. Given a history of an agent's motion and a similar context, there are multiple possible and plausible motions of this agent in the future. In fact, the future trajectory of an agent in the scene depends on the agent's goal and behavioral characteristics or driving style and on the interaction with the surrounding road users. However, the goals and behaviors are not externally observable. Therefore, distinct possible goals could be defined and for each potential goal or behavior, there could be a distinct possible future trajectory or mode. Even humans reason by accounting for multiple possibilities of future motions and put more focus on the most probable one. This evokes the problem of defining the different intentions corresponding to the different possible trajectories in a specific context. In addition, it raises the question about the plausibility of the predicted trajectories and how we evaluate it.

To address the above challenges, we propose different deep learning based methods that employ various input cues. Then, we investigate different ways of modeling the interactions between the neighboring agents and the static scene context. Since the attention mechanism has great achievements in performing relational reasoning, we focus, in this work, on vehicles interactions modeling using various techniques of attention mechanism such as soft attention [6], multi-head dot product attention [107], double attention [18], etc. We also explore various ways of mounting the attention blocks to handle different sub-tasks of a trajectory prediction framework. Moreover, we incorporate the uncertainty regarding the actions of human drivers using multimodal trajectory predictions. We also study the use of different implicit and explicit distinct generic intentions associated to each prediction mode in order to enhance the diversity of the predicted trajectories.

The trajectory presents spatial and temporal information. In this thesis, we denote the interactions with the surrounding agents and with the map as spatial information and the motion evolution as temporal information. Since each agent motion depends on the other surrounding agents and the static scene elements, the spatial and temporal information are highly correlated.

1.3 Thesis Outline

The organization and contents of the chapters of this thesis are presented below.

In Chapter 2, we analyze the state-of-the-art studies addressing the problem of trajectory prediction. First, we describe the overall structure of a trajectory prediction framework with a focus on the cues considered to infer the future motion in Section 2.1. In addition, we consider the output forms (trajectories, maneuvers, etc.) and the different methods of generating multimodal predictions. Then, we examine the different methods of trajectory prediction starting by a brief description of the main classical approaches in Section 2.2. Section 2.3 details the Deep Learning (DL) based methods and the architectures deployed i.e. LSTMs, CNNs, Transformers, etc. Specifically, we investigate the different methods of trajectory encoding and generation and interaction modeling between the target agent and the static and dynamic context elements. Finally, we describe publicly available naturalistic driving datasets and we compare them. We also give further details about the datasets that we use in our experiments to evaluate our proposed methods.

In Chapter 3, we tackle the task of trajectory prediction on a highway. To do so, we exploit two main cues; the past trajectory of the target vehicle and those of its surrounding agents. A commonly encountered challenge here is modeling the interactions between the agents and analyzing their influence on the future trajectory of the target vehicle. Therefore, we propose two different methods of trajectory prediction with a focus on interaction modeling; the first one, described in Section 3.3, applies simultaneous spatial and temporal dependencies modeling at each time step based on Relational Recurrent Neural Networks (RRNNs). It is composed of two main blocks: a Long Short-Term Memory (LSTM) and a Multi-Head Attention (MHA) module. The second method MHA-LSTM (cf. Section 3.4) is composed of the same building blocks. However, these blocks are mounted differently which constitutes the singularity of each method. Our MHA-LSTM applies attention on high level representations of the agents' trajectories which enable its interpretable interaction modeling. Indeed, the visualisation of attention maps makes it possible to infer which agents most influence the target agent's future trajectory. We also investigate the advantages of each approach using experiments on two publicly available naturalistic driving datasets in Section 3.5.

In Chapter 4, we consider the task of trajectory prediction in an urban environment. For this purpose, we augment the MHA based method with additional information about the static scene structure using a Bird Eye View (BEV) high definition map. In other words, we apply MHA on a joint representation of the static scene and the surrounding agents (cf. Section 4.1). We also take into account the uncertainty of the future by proposing a multimodal solution. Actually, we use each attention head to generate

a possible future trajectory. Then, in order to enhance the diversity of the predicted trajectories and ensure that they cover all the possible paths, we augment our model by different methods of defining drivers intentions. In other words, we attribute a distinct intention to each prediction mode to avoid the mode collapse issue. In Section 4.2, we propose to replace the MHA with alternative attention based modules such as the Double Attention (DA) network in order to reduce the complexity of the model while generating competitive predictions. To evaluate our proposed methods, we compare them to recent state of the art methods using different metrics. Experiments and ablation studies are conducted in Section 4.4.

Chapter 5 addresses the task of trajectory prediction in an urban environment using an architecture that is better suited to the real time requirements of an AV. Section 5.2 presents our proposed non auto-regressive solution based mainly on Transformer blocks. This architecture is interesting since it has the ability of parallel implementation on GPUs during the training and the testing phase as well. We also augment the Transformer architecture with an additional layer to simultaneously model the spatial and temporal information. In addition, we built a multimodal solution based on a set predefined intentions in form of anchor trajectories that we pass as inputs to the Transformer decoder.

Finally, Chapter 6 summarizes our contributions and gives an overview of future work.

2
State-Of-The Art

Contents

2.1 Overall Motion Prediction Framework 10
2.1.1 Stimuli 10
2.1.2 Modeling approach 12
2.1.3 Prediction 12
2.2 Classical Methods for Trajectory Prediction 15
2.2.1 Physics-based methods 15
2.2.2 Pattern-based methods 16
2.2.3 Planning-based methods 17
2.3 Deep Learning for Motion Prediction 18
2.3.1 Different Architectures of Sequence Modeling and Prediction 19
2.3.2 Agent Environment Interactions 29
2.4 Summary of Trajectory Prediction Deep Learning based Studies . 32
2.5 Datasets 35
2.5.1 Next Generation Simulation (NGSIM) 36
2.5.2 highD 37
2.5.3 nuScenes 37

Motion prediction is a key task in many domains such as service robotics, advanced surveillance systems and autonomous vehicles. Consequently, it has been addressed in the literature in different contexts, for different dynamic objects and from different perspectives. Considering various approaches, we notice that if one is applied in a particular context, it can also be extended to another with minimal adjustments. Therefore, even though we are tackling here the task of trajectory predictions for autonomous vehicles, in this section, we consider the studies related to trajectory prediction of a target agent in general, which represents a dynamic object of interest such as a robot, pedestrian, cyclist, car or other. In addition, numerous motion prediction methods have recently been proposed. Here, we give an overview of the deployed methods, focusing on deep learning-based methods.

2.1 Overall Motion Prediction Framework

We follow Rudenko et al. [87] who divide the motion prediction problem into three main components: stimuli, modeling approach and prediction.

2.1.1 Stimuli

The stimuli are the features that influence and help to determine the future intention of the target agent. They are mainly composed of target agent cues and static and dynamic environment features.

2.1.1.1 Target agent features

A future agent's motion is a continuation of a previous behavior since each motion results from a relatively slow (or fast) progressive change in the speed and direction that obeys physics laws of motion. Therefore, the main observable target agent cues consist of its past state observations (positions, orientation, velocities, acceleration, etc.). Moreover, the type or class of the agent considered (pedestrian, cyclist, car, truck ...) represents an important cue to predict its future motion since for each type of agent there is a limit to the possible variations of its speed and, therefore, the possible patterns.
In the state-of-the-art, different target agents cues have been utilized. Lenz et al. [61] use as input to their model only the current state of a set of neighboring vehicles in order to achieve the Markov Property. However, most existing studies [2, 23, 24, 68, 69, 82] use a sequence of past features to benefit from extra temporal information and exploit the characteristics of the motion variation (direction, speed, pattern...) in the prediction task.

2.1.1.2 Environment features

The environment features are composed of:

- Static elements including static obstacles and environment geometry.
- Dynamic elements representing the other traffic users.

Static scene elements (such as road, lanes, crosswalks ...) help to determine the reachable areas and therefore the possible patterns of the target agent motion. Moreover, the surrounding dynamic agents interact with each other and with the target agent. They have an influence on its future motion. Different state-of-the-art studies exploit static scene features [75, 90] or dynamic ones [1, 23, 24, 35, 110, 116] or both [16, 21, 28, 38, 59, 81, 85, 89, 120] to infer the motion of the target agent.
Context features are captured using the sensors mounted on the target agent or at a specific location in the scene. They can be fed to the model as raw sensor data [14, 15, 75] such as camera images (bird's-eye view images of the scene [89, 90, 100]) or LiDAR data points [75, 83, 117] to tackle the tasks of detection and intention prediction simultaneously. Other studies deal with converting raw data to input features prior to the prediction task and use a high fidelity map [16, 21, 28, 38, 81] obtained after image segmentation. Static features are fed to the prediction model as a single image of the scene [16, 21, 28, 38, 81, 89, 120] at the prediction time or as a sequence of past scene states [85].
Most of state-of-the-art studies present a dynamic context (composed of surrounding agents) represented as a state vector or a sequence of states (positions, velocities, accelerations, class, etc.). Differently, Ridel et al. [85] represent the past trajectories of agents as a sequence of binary 2-D grids.
Djuric et al. [28] build a rasterized representation based on the high-definition map and the surrounding agents of each vehicle (cf. Figure 2.1). It represents each agent by a sequence of oriented bounding boxes corresponding to its position and orientation at each past timestep. In addition, the static scene is represented by an image with various map elements such as lanes, cross-walks and boundaries. To highlight the difference between them, each map element is assigned a distinct form and an RGB color. The drivable surface and the crosswalks are represented by two differently colored polygons while lanes describing the driving path are exhibited by boundaries and directed lines. This representation is adequate for CNNs. It was later used in several state-of-the-art approaches [16, 21, 81].

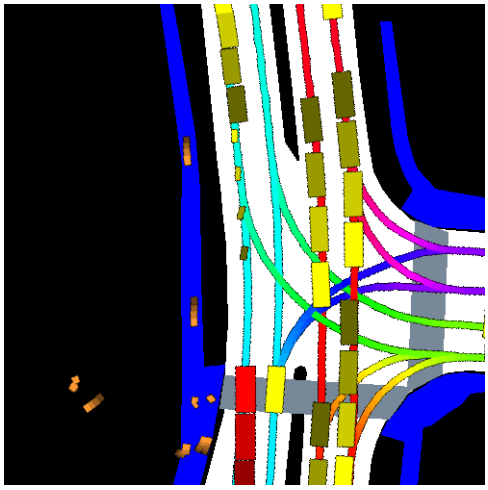


Figure 2.1: Rasterized representation of the static and dynamic scene elements

2.1.2 Modeling approach

With the recent development of deep learning-based methods and their great achievements in numerous fields, most of recent studies on trajectory prediction are based on deep learning methods. Therefore, we divide motion modeling approaches into two main categories, recent modeling methods mainly based on deep learning approaches and classical ones deploying hand-crafted features. Classical motion modeling methods subdivide the prediction approaches by considering the agent motion representation [87]. Physics-based methods use dynamical models based on the physics laws of motion. Pattern-based methods learn to generate trajectories from the agents' motions. Planning-based methods infer motion intention based on a policy considering rational agents. Recent methods are categorized with respect to the deep learning architecture and the approach used (recurrent, convolutional, generative or attention based) and their interaction modeling form.

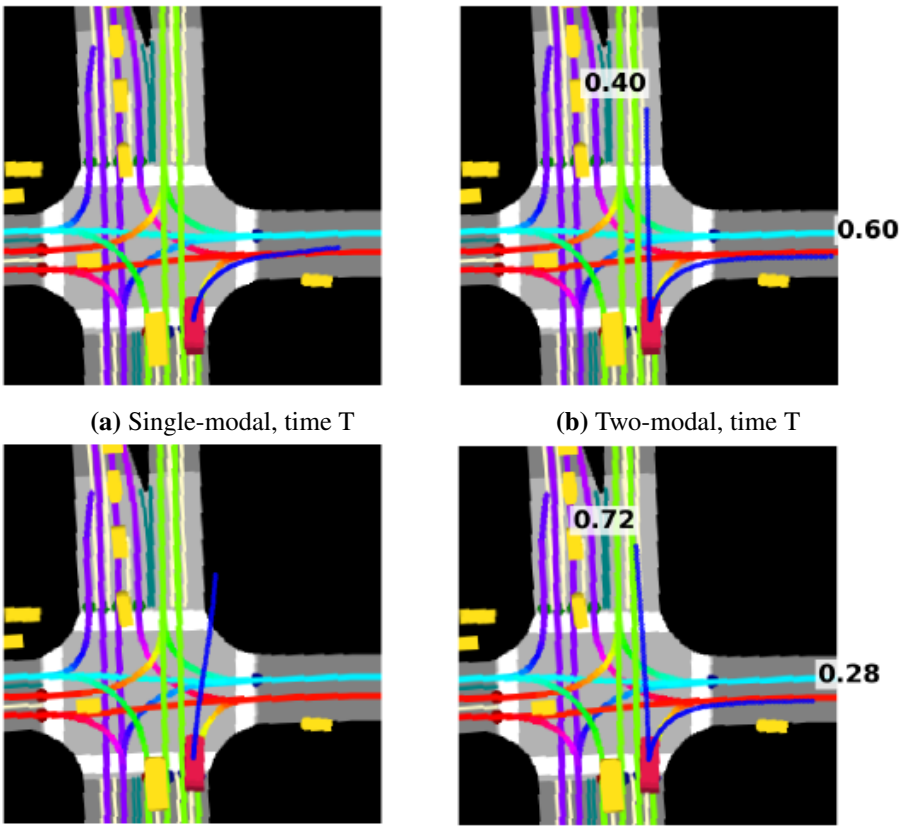
2.1.3 Prediction

2.1.3.1 Output Forms

Vehicle intention prediction is divided into two main types: maneuver [27, 82] and trajectory prediction [2, 8, 113]. The former generates a high-level representation of the motion such as lane changing and lane keeping. The latter outputs the predicted agent's states over time. These two types are combined in some studies [23, 24, 40]. Indeed, maneuvers are first inferred in [40]. Then, trajectories are generated as realisations of the selected maneuver.

Furthermore, state-of-the-art trajectory prediction studies develop models that generate

either a prediction for a single agent using an agent-centric approach [23, 24, 85] and, then, repeat the same reasoning to generate predictions of a bigger number of agents or generate predictions for multiple agents in one-shot [35, 76]. The latter approach commonly exploits important scene and agents' features to reason simultaneously about the behaviors of multiple agents.
Different forms of outputs can be computed in the motion prediction task. In [2, 35], the exact future positions are predicted. Other studies [1, 23] generate the mean and covariance over the positions in order to estimate the uncertainty in the prediction. Kim et al. [50] generate the future motion prediction of vehicles as probabilities over an occupancy grid map.



(a) Single-modal, time T **(b)** Two-modal, time T

(c) Single-modal, time T + 3 **(d)** Two-modal, time T + 3

Figure 2.2: Comparison of single and multimodal future vehicle trajectories predictions [21]; predicted trajectories are marked in blue, with their probabilities indicated at the end of the trajectories.

2.1.3.2 Multi-Modal Output

The inherent uncertainty of the future imposes that there is not one single correct possible future trajectory prediction. Given a history of an agent's motion and a similar context,

there are multiple possible and plausible motions of this agent in the future. In fact, the future trajectory of an agent in the scene depends on the agent's goal and behavioral characteristics or driving style and on the interaction with the surrounding road users. However, the goals and behaviors are not externally observable. Therefore, distinct possible goals could be defined and for each potential goal or behavior, there could be a distinct possible future trajectory or mode. Even humans reason by accounting for multiple possibilities of future motions and put more focus on the most probable one.
Cui et al. [21] show that a model predicting a single trajectory (single-modal) leads to *mode-averaging*. Such a model learns an average behavior that could represent a non plausible prediction. Figure 2.2 shows an example of a single-modal and a multimodal (two-modal) predictions given an input trajectory in an intersection. At the a prediction time T, the single modal model predicts a turning right maneuver (cf. Figure 2.2a) while the multimodal model infers the two possible future maneuvers; going straight on and turning right with a higher likelihood (cf. Figure 2.2b). Three seconds later, as the vehicle moves straight, the single-modal model generates a non plausible trajectory suggesting that the vehicle would drive outside of the derivable area (cf. Figure 2.2c). In this case, the predicted trajectory presents an average motion between going straight on and turning right maneuvers. The two-modal model succeeds in predicting two plausible trajectories corresponding to two possible maneuvers. It infers the probability of each predicted trajectory reasonably (cf. Figure 2.2d). In sum, multi-modality assures that for a given input trajectory, there are multiple possible future predictions that draw a probability distribution over different output sequences.

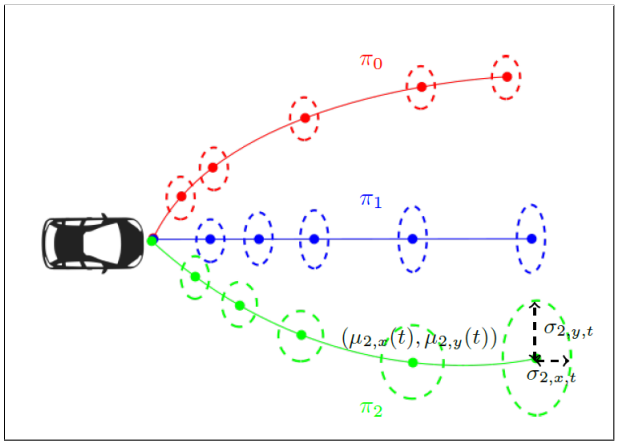


Figure 2.3: Probabilistic multimodal trajectory prediction where each trajectory is represented by a sequence of two-dimensional Gaussian distributions [12]. The three trajectories (red, blue and green) present the three possible modes predicted and the Gaussian distributions present the uncertainty within each mode.

Recent approaches learn one to many mappings from input context to multiple future trajectories. Some multi-modal techniques require labeling modes before the training. Deo et al. [23] attribute one of six maneuvers to each trajectory in the training set and learn to classify these modes and generate one trajectory per mode for each agent. Other deployed approaches sample generative models such as Conditional Variational Autoencoder (CVAE) [9, 59, 63, 91] and Generative Adversarial Networks (GANs) [35, 54, 89, 120]. Other methods sample a stochastic policy learnt by imitation or inverse reinforcement learning [25, 65]. Ridel *et al.* [85] predict the probability distributions over grids and generate multiple trajectory samples. Others [16, 21] utilise a mixture model generating a fixed number L of plausible trajectories for an agent, where each trajectory is represented by a sequence of two-dimensional Gaussians and a probability associated with each of the L trajectories (cf. Figure 2.3).

2.2 Classical Methods for Trajectory Prediction

Existing approaches can be classified in three families according to the motion model used: physics-based methods, pattern-based methods and planning-based methods. In the following, we describe the main methods deployed in each class.

2.2.1 Physics-based methods

For these approaches, the future trajectory of an agent is predicted by propagating forward its past trajectory using explicit physics-based dynamic or kinematic models [7, 60, 102, 108]. Linear motion models are often used for motion prediction. These models assume that the target agent moves with Constant Velocity (CV) or with constant acceleration (CA). Ammoun and Nashashibi [4] use a CV motion model to forecast agents' future trajectories and then estimate the collision risk. They also account for the uncertainties in the agents' states using multivariate Gaussian distributions. The uncertainty is propagated in time using a Kalman filter. Similarly, in the task of collision risk estimation, Rummelhard et al. [88] apply a CV model. Then, they project the predicted states in an occupancy grid that is transformed to a risk grid. Elnagar [30] also uses a Kalman filter for motion prediction of dynamic obstacles using a constant acceleration model. However, assuming that the agents' motion is linear is not realistic, especially for complex motions, and limits these methods' prediction horizon. Therefore, another approach [46, 47] was deployed to model the general motion of agents. It consists in defining different motion modes, such as linear movements with CV, accelerating, decelerating or turning maneuvers. Each mode is characterised by a different dynamic regime. Combined together, these modes can compose a sequence of complex motions.

The challenge in this method is to select the modes that describe the best the future trajectory knowing the agent's recent states and its static and dynamic scene context. To do so, other studies [46, 47] adopt a set of motion models using the Interacting Multiple Model (IMM). A Kalman filter with the corresponding dynamics is associated to each model. A probability distribution over the deployed Kalman filters is updated with each observation. The predictions are computed as a weighted combination of the motion predicted by each filter. As there are multiple factors that influence the behavior of an agent, there is no specific analytical expression of the distribution on the predicted states. Therefore, other studies [11] use Monte Carlo sampling to approximate this distribution. Broadhurst et al. [11] assign a state update equation (evolution model) based on an input signal to each object in the traffic scene (obstacle, pedestrian, car). Then, they randomly sample the input variables of these models in order to generate plausible future trajectories. They also introduce additional constraints so that: the predicted trajectories remain inside the drivable region, do not result in a collision with other objects, and satisfy the dynamic limits of the cars. Other factors are also taken into consideration in predicting the trajectories distributions such as the level of risk and comfort.
In general, physics-based methods are basically built upon the motion's low level properties. Consequently, they are restricted to short-term motion prediction since they lack the modeling of interaction and context.

2.2.2 Pattern-based methods

Pattern-based methods learn the behaviors of agents by fitting function approximators to trajectory data. Aoude et al. [5] combine a physics-based approach with Gaussian Processes based motion patterns to generate probabilistically weighted feasible motions of the surrounding vehicles. Other methods subdivide the vehicle trajectory into a finite set of typical patterns named maneuvers. They define the set of maneuvers considered differently and deploy methods of maneuver recognition and prediction. Tran and Firl [105] generate Gaussian process regression models which describe all the possible situations. Then, they identify the vehicle maneuvers by comparing the likelihoods of the observed track to the constructed non-parametric models. They extend their method to generate multimodal prediction by applying the Monte Carlo method. Hermes et al. [37] cluster the motion patterns into maneuvers using a mean-shift technique with a rotationally-invariant distance metric. They predict a vehicle's trajectory by matching its observation data to the center of the trajectory cluster with the highest likelihood while adopting a particle filtering approach. Schlechtriemen et al. [94] use a predefined set of maneuvers (lane following, lane change). Then, they deploy a Naive Bayes Classifier to estimate the probability of a sample of trajectory features belonging to each maneuver.

They also integrate this classifier with a Hidden Markov Model (HMM), where each state of the HMM corresponds to one of the maneuvers extracted from the naturalistic driving data in order the improve the classification performance. Houenou et al. [40] design a maneuvers recognition module that proceeds by comparing of the vehicle's path and the shape of the road. Then, they generate different continuous realizations of the predicted maneuver and select the best trajectory that minimizes a comfort-based cost function. The main limitation of these approaches is that they do not model the interactions between the neighboring vehicles on the future trajectory. Kafer et al. [48] tackle the task of joint pairwise vehicle trajectory prediction at intersections. They compare the observed motion pattern to the database and extract, for each vehicle, possible predicted trajectories independently. Then, they jointly compute, for each pair, the probability of possible trajectories.
Most recent studies deploy deep learning-based methods. They will be detailed in the Section 2.3.

2.2.3 Planning-based methods

Planning-based methods tackle the task of sequential decision-making by reasoning on the motion intention of agents. They assume that agents perform rational actions when modeling their motions. These models mainly consider the transitions between the states and the impact of current actions on the future.
Many studies perform goal-informed predictions. They use optimal motion techniques with a hand-crafted cost-function that models the agent's preferences with reference to the level of comfort (smoothness of the motion), the safety of each action, the energy consumption and the adherence to the traffic rules.
Yi et al. [115] build an energy map to model the future behavior of agents moving inside a crowd while accounting for three influence factors; the scene layout, stationary and moving agents. They start by constructing an energy map per influence factor. The general principle consists in having higher energy when the risk of collision is minimal for the possible motion. Therefore, the energy increases with the distance between each position in the scene and the influence factor. Different expressions of the energy were deployed depending on the type of the influence factor. Then, the authors compute a combined influence map by an elementwise multiplication of the three previously generated maps. To predict the future trajectory, the Fast Marching algorithm is applied for a given source and destination position to generate an optimal path based on the energy map.
Karasev et al. [49] tackle the task of trajectory prediction using a jump Markov Decision Process (MDP) with the agent's goal as a hidden variable. The goal is considered as a slowly time-varying variable predicted at each time step using the agent's recent states.

They also model the agent's motion as switching nonlinear dynamical systems. They use an MDP stochastic policy that describes the agent's motion dynamics given the predicted goal and the current state. The predicted trajectory is sequentially generated according to the optimal policy that maximizes a reward function considering the changes in the scene context.

Sierra González et al. [96] deploy an MDP to represent the driver decision-making strategy. They model a vehicle's trajectory by a sequence of states. Then, they build a cost function using a linear combination of static and dynamic features parameterizing each state. In contrast to Karasev's approach [49], Inverse Reinforcement Learning (IRL), accounting for risk-aversive vehicles' interactions, operates to learn the cost function parameters from demonstrations. In addition, Sierra González et al. [97] extend their model introduced in [97], by using Dynamic Bayesian Networks to better model vehicles' interactions.

Planning-based approaches assume that agents are rational and that they always opt for the most comfortable behavior that minimizes the risk and obeys the traffic regulations. This assumption is not always true, which makes us question the performance of these approaches.

Classic trajectory prediction approaches have a lot of limitations since they use hand crafted features. In addition, they are based on relatively simple designed models that fail to extract and efficiently exploit the dependencies between the target agent and the static and dynamic scene context and then, to generate long-term trajectory predictions.

2.3 Deep Learning for Motion Prediction

Different deep learning architectures were designed to tackle specific tasks in different domains. For instance, recurrent networks and attention mechanisms were mainly deployed for language modeling in natural language processing and convolutional networks are extensively used to solve the problem of image classification and object recognition in computer vision. These architectures have been also extended and adapted to other domains like financial forecasting and climate understanding. They are also deployed in the autonomous driving domain and specifically, for vehicle motion forecasting.

In this section, we present different deep learning-based approaches and we describe how they are adapted for the trajectory prediction task. Most interaction aware trajectory prediction frameworks are composed of three main modules: (i) A feature encoding module: extracts important information from the inputs (target and surrounding agents motion dynamics, scene representation). (ii) An interaction modeling module: models the

interaction between the target agent and the surrounding agents and/or the environment. (iii) A trajectory generator module: receives the representations produced by the two previously described modules and generates predictions about the future motion of the target agent.
These modules can be separate or combined depending on the architecture adopted. In addition, the input feature encoder and the trajectory generator of each model are tightly coupled. Therefore, we divide this section into two main parts. The first one describes the different architectures deployed to extract salient features, model the agent motion and generate a prediction. The second part examines the interaction between the agent and its surrounding environment.

2.3.1 Different Architectures of Sequence Modeling and Prediction

In this part, we describe the different deep neural networks architectures deployed for sequence modeling and prediction. Recurrent Neural Networks (RNNs) and their variants (Long Short Term Memories (LSTMs) and Gated Recurrent Units (GRU)) are the most popular networks used in the task of trajectory predictions. We describe the usage in their regular form or as an encoder-decoder. Then, CNNs and their extensions to handle time series (temporal CNNs, convolutional LSTMs) are detailed. Trajectory prediction is also considered as a sequence generation task. Therefore, different forms of generative models are used to produce multi-modal plausible predictions (details in 2.3.1.3). Finally, we present the attention mechanism and the different manners in which it is used for sequence modeling and prediction.

2.3.1.1 Recurrent Sequence-to-Sequence Model

Data collected over successive periods of time are denoted as a time series. Motion prediction can be treated as a time series regression or classification problem. RNNs and their variants are the main reason behind the significant advances in sequence modeling and generation. They have shown promising results in diverse domains such as natural language processing and speech recognition. Therefore, RNN-based approaches have also been deployed in the tasks of maneuver and trajectory prediction. Unlike other neural networks, they consider sequential information and model the dependency in inputs. They act by performing the same operations for every input item of a sequence while taking into consideration the computation of the previous input item (cf. Figure 2.4).

LSTMs are a particular implementation of RNNs. They are characterised by their ability to capture and represent long-term relations in sequential data. They proceed by

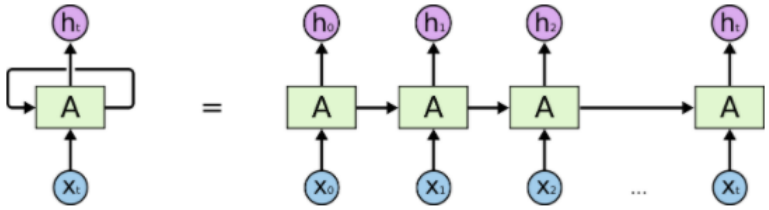


Figure 2.4: Recurrent Network A with input sequence x_t and outputs h_t

passing information about the previous steps to the next step of the sequence using a hidden state vector. The latter stores information about the previously seen sequence data. It acts as the LSTM's memory.

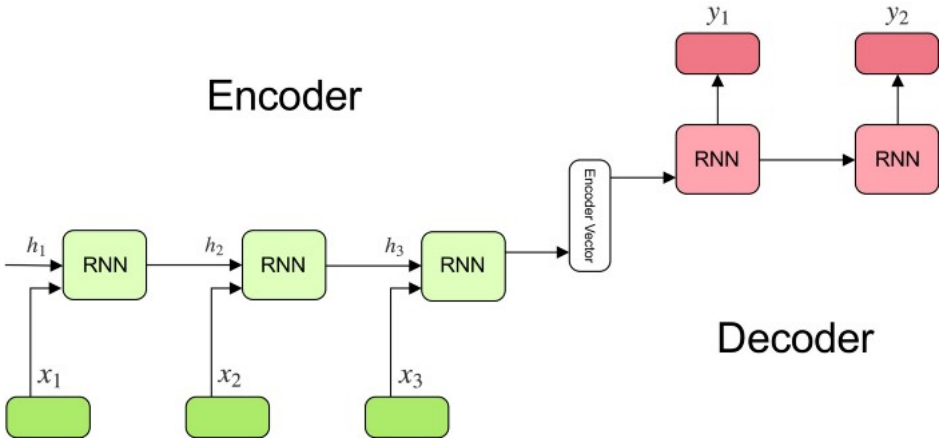


Figure 2.5: RNN encoder-decoder

LSTMs have recently been deployed for predicting agents' future behaviors. Different LSTM-based models have been designed going from a simple LSTM with one or more layers in [2, 50, 61, 82, 121] to different types of combinations and extensions: A dual LSTM architecture was adopted in [113]: the first LSTM extracts high-level driver behavior succeeded by a second one for continuous trajectory generation. Pfeiffer et al. [80] use three input channels, each one includes an LSTM network. The first one encodes the target agent's past states. The second input features representing a 2D occupancy grid, encoding the static obstacle information and the third encodes the states of surrounding agents presented in a special hybrid grid. The three output vectors of the LSTMs are fed sequentially to a common LSTM. On top of the latter LSTM, two successive fully connected layers are mounted. The last layer outputs the sequence of the future time steps states at once. Alahi et al. [1] use a social LSTM where each of the surrounding agents' motion is represented by an LSTM cell and interactions are modeled by social pooling (further details about interaction modeling in section 2.3.2.2). Bartoli

et al. [8] extend the social LSTM to model the interaction between the target agent and static scene context.
Subsequent studies [76, 106] consider trajectory prediction as a sequence-to-sequence learning task that maps the past trajectory to the future one. They employ the encoder-decoder architecture in order to extract salient information about the past trajectory pattern and generate the future trajectory sequence. The LSTM encoder receives the embedding of the input sequence of the past states of the target agent and generates a compact vector which represents the salient information of the past trajectory. The LSTM decoder receives a fixed length vector representing the encoding vector with or without information about static or dynamic objects of the scene context. Then, it recursively produces the future trajectory (Figure 2.5).
In the work of Park et al. [76], the LSTM encoder receives the trajectories of the surrounding vehicles as well as a sequence of the past states of the ego vehicle and generates a compact vector that represents the salient information of the past trajectory. Varshneya et al. [106] feed the LSTM encoder with a stack composed of the agent state and the interaction with the static and dynamic representation at each past time step. To generate each future state, the LSTM decoder attends to the encoder hidden states using a soft attention mechanism. Tang and Salakhutdinov [100] jointly predict the future trajectories of multiple agents. To do so, they employ a probabilistic framework using a variational LSTM-based encoder-decoder with a discrete latent vector for each agent.
Other studies [43, 54, 91, 110] opt for simultaneous spatial and temporal reasoning. They use LSTMs as models of spatio-temporal graphs (details in section 2.3.2.2).
Recurrent networks are also combined with different neural network architectures in numerous studies. Almost all studies considering the spatial scene context as an input to their network employ a CNN to generate the input scene image representation. Therefore, recurrent networks are coupled with CNNs in numerous studies [59, 84, 89, 91, 106, 114, 120]. Furthermore, recurrent architectures are employed in motion prediction approaches using generative models such as Generative adversarial networks (GANs) [35, 54, 89, 120] and variational auto-encoder (VAE) [9, 59, 63, 91] to generate multi-modal predictions. Combining LSTM with other deep learning architectures helps to better model the spatio-temporal dependencies and thereby, enhances the prediction performance.

2.3.1.2 Convolutional Neural Networks for Trajectory Prediction

CNNs are mainly deployed for image features representation and related fields like image classification, segmentation and object detection. Various architectures of CNNs are widely applied in building algorithms among them we mention AlexNet, VGGNet, GoogLeNet, ResNet, etc. The role of the CNN is to extract salient features from the

input image by applying successive layers of convolutional and pooling operations. A convolutional layer consists in sliding a learnt filter over the input feature spatial dimension and applying it on each covered local part (Figure 2.6). Applying a same filter to the input generates a feature map indicating the strength of the detected feature in this input. While the first convolutional layers capture the Low-Level features like color, edges, etc, deeper layers extract the High-Level features as well. This results in a wholesome understanding of the input images.
The Pooling layers are responsible of extracting dominant features and discarding the noisy activations. In addition, they reduce the spatial size of the input features considered. This results in decreasing the computational power of feature learning and increasing the efficiency of the training process. The two main types of Pooling deployed in CNNs are max pooling and average pooling. The max pooling and average return the maximum value and the average value respectively from the part of the features covered by the Kernel.

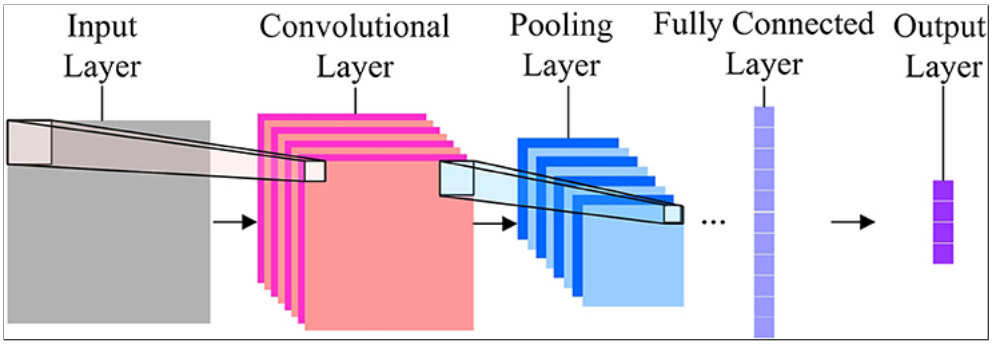


Figure 2.6: CNN Layers

As CNNs are mainly applied to images, the image representation of the static or/and dynamic context is the suitable input to a CNN. Recent state-of-the-art studies [16, 21, 28, 81] deploy deep CNNs to predict vehicle trajectories. The CNN, fed with the rasterized representation of the scene, simultaneously infers relevant features and generates a set of possible trajectories. In the rasterized representation of the scene, vehicle state history is represented by a sequence of bounding boxes. Hong et al. [38] deploy a similar map representation of the scene and combine it with a grid-based representation of the temporal evolution of the target and surrounding agents in the scene. They build a fused complex scene context presented as a convolutional grid. They feed the formed 3D tensor to a CNN to jointly encode the spatio-temporal information (agents dynamics and interactions, and scene context). Then, they compare two different decoding architectures. The first one consists of a “one-shot” prediction of the entire output sequence. The second deploys

an RNN-decoder which sequentially generates a distribution at each future step. The CNN does not explicitly relate the sequence of the agent states. This makes us question its ability to model the temporal evolution of dynamic agents' motion.
As a remedy, Nikhil et al. [74] deploy a variation of CNN as a sequence-to-sequence model, able to capture the spatio-temporal aspect of the input data. The sequence-to-sequence CNN is composed of a hierarchy of temporal convolutions (convolutions are computed across time). Predictions at each timestamp are based only on the elements from the current timestamp or earlier in the previous layer. Unlike RNNs where inputs are processed sequentially, convolutions are performed in parallel by the application of the same filter in each layer, meaning that all time-steps are processed simultaneously.
Convolutional LSTMs (ConvLSTMs) [95] also use convolutional operations to represent spatio-temporal data such as a sequence of images. They are built by integrating convolutional structures in the form of the regular LSTMs. An LSTM captures the dependencies between the input sequence of vectors. It proceeds by sequentially passing the vector of hidden state from one cell to the next and storing the past sequence of important features in it. However, the LSTM architecture can only handle the temporal aspect of the input sequence, but, it fails to model the spatial aspect to some extent. The convLSTM maintains the same general structure of the LSTM while introducing convolutional operations in it to deal with the spatial information in the image sequence. Unlike the LSTM, the input to the convLSTM cell are 3D image features processed by convolutional layers. Furthermore, instead of the 1D hidden state vector used in LSTMs, the extracted features flow through the ConvLSTM cells in the form of a 3D tensor (same as the input dimension). This also requires modifications in the internal structures of the ConvLSTM cells where the internal matrix multiplications are exchanged with convolution operations.
Recent studies [64, 85] use ConvLSTM for trajectory prediction. While Li et al. feed the ConvLSTM with only motion dynamics expressed by a three-dimensional tensor of displacements at each time step, Ridel et al. [85] use a concatenation of bird's-eye views of the static context with motion dynamics grid as input to the ConvLSTM. Considering the static scene context enables the convLSTM to generate more realistic predictions compliant with the scene structure.
In a nutshell, CNNs and temporal CNNs have generally faster training than RNNs. However, they have limits in simultaneously modeling spatial and temporal dependencies. convLSTMs represent well the spatio-temporal dependencies, but since they are based on a recurrent cell, they have comparable training time to the RNNs. They perform well in the trajectory prediction task, especially when they model both the scene context and agents' interactions.

2.3.1.3 Generative Models for Trajectory Prediction

Generative models consider the training data as samples from a data distribution. They aim to learn a representation of that distribution estimate called a model distribution. Generative models can be divided into two categories, models that explicitly learn the model distribution and others only capable of generating samples from it. Generative adversarial networks [33] (GAN), variational auto-encoder [53] (VAE) and their conditional variants are approaches to generative modeling utilizing deep learning methods like CNNs and RNNs. They have achieved impressive performances in representation learning and distribution approximation tasks.

Generative adversarial networks

GANs don't explicitly learn the distribution of the data. However, they are capable of generating new realistic samples from the model distribution. A GAN is composed of two sub-models, a generator and a discriminator, trained together using a minimax game. The generator model takes a random vector as input and generates new examples. The discriminator is fed by samples from the dataset and others generated by the generator network. It learns to classify these examples as real (from the dataset) or fake (from the generator). The training of GAN-based models enables the generator to learn to generate plausible examples. This can be detected when the discriminator is fooled by about fifty percent of the generated examples. Gupta et al. [35] were the first to introduce the LSTM-based recurrent encoder-decoder based GAN for multi-modal trajectory prediction. The encoder-decoder-based generator receives as input past trajectories. It encodes the history of the agent, computes agents' interaction vector using social pooling and generates the future trajectory. The discriminator, fed with the ground truth and predicted trajectories, classifies them as plausible or not. (cf. Figure 2.7)

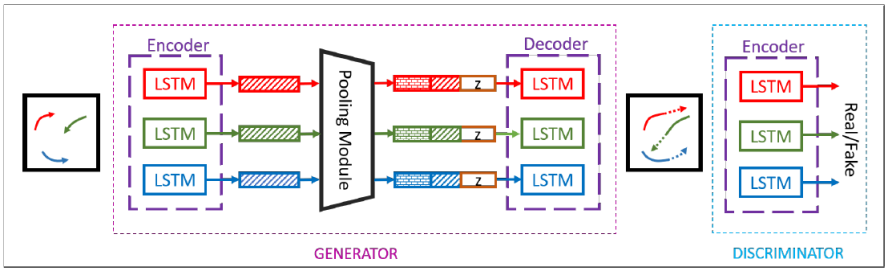


Figure 2.7: Example of GAN based architecture [35]

Other subsequent studies [3, 54, 89, 120] also deploy GANs for trajectory prediction with both scene context and social interactions. They generate multi-modal predictions based on GANs couplet with different deep learning architectures such as LSTMs and

CNNs. SoPhie [89], Social Ways [3] and MATF GAN [120] methods use an LSTM for agent motion encoding and a CNN for map feature extraction then they compute the agents and scene interaction features and feed them to the LSTM generator. The discriminator, in the SoPhie [89], is composed of an LSTM that inputs a randomly selected trajectory sample either from the ground truth or generated one and decides whether it is a plausible prediction or not. Unlike SoPhie [89], the MATF GAN [120] discriminator inputs the ground truth past trajectories concatenated with either the ground truth or the generated trajectory. This helps the discriminator to make better decisions based on the continuity of the overall trajectory. Kosaraju et al. [54] build a more extended model Social-BiGat, based on conditional GAN with one generator and two forms of discriminators; the first one is specialized in local agent scale, and the other in a global scene-level scale. Similar to SoPhie [89] and MATF GAN [120], the generator is composed of an LSTM encoder for the agent's past states, a CNN to model the scene context, an interaction extractor block and an LSTM decoder that receives the agents and scene encoded features and generates a predicted trajectory. Differently, the local discriminator comprises encoder LSTMs operating on agents while the global encloses LSTM encoders for agents' dynamics and a CNN for scene features representations. We note that, unlike SoPhie [89] and MATF GAN [120], the global discriminator in Social-BiGat [54] is conditioned on combined agent and scene interaction information. This incites the generator to produce more scene compliant trajectories. GAN-based models generally overcome the difficulty of approximating intractable probabilistic computations, but they suffer from mode collapse problems.

Variational Autoencoders

The Variational Autoencoder (VAE) is made up of an encoder-decoder made up of deep learning networks such as CNNs and LSTMs. This encoder-decoder architecture creates a bottleneck (reduced representation) for data which ensures that only the important features are retained and used to reconstruct or generate data. The encoder produces a compressed representation from its input features as a distribution over the latent space. The decoder learns to reconstruct the original data or to generate new data. It receives a point sampled from the distribution over the latent space and outputs the parameters of the probability distribution of the generated data. The distribution over the latent space is enforced to be close to a standard normal distribution in order to ensure the continuity and the completeness of the latent space using a regularizer (cf. Figure 2.8).

Recent studies [59, 91] use a Conditional VAE (CVAE) [98] to produce position distributions for trajectory prediction. The DESIRE framework [59] consists of two main modules. The first is a CVAE-based GRU encoder-decoder that generates multiple

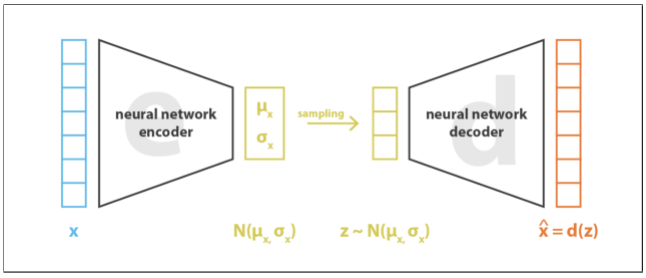


Figure 2.8: VAE architecture

predictions based on agents' past states. These predictions, along with context, are fed to the second module. The latter is a GRU-based module responsible for ranking and refining the predictions by assigning a reward function to each prediction and proposing a displacement vector to regress it sequentially at each time-step. In VAE, the encoded training distribution is forced to match the prior one. However, the two distributions do not match as expected in many cases since VAE assumes a standard Gaussian prior on the latent variables. This causes a model bias and makes us question its ability to capture multi-modal distributions.

As a remedy, Salzmannet al. [91] develop a CVAE and LSTM encoder-decoder-based model for trajectory prediction. Here, a scene is represented as a directed spatiotemporal graph, agents and their interactions as nodes and edges (cf. details in Section 2.3.2.2). They produce the predicted trajectory distribution by deploying a discrete categorical latent variable which presents the high-level latent behavior. As the latent distribution is discrete, it enhances the interpretability of the model and the consistency between the encoded latent distribution and the prior. Li et al. [63] use a double attention graph (two attention blocks one on topological and the other on temporal features) based on Wasserstein auto-encoder [103](WAE). This auto-encoder deploys a different regularizer than the VAE's: it minimizes a variant of the Wasserstein distance between the latent distribution and the prior one. This induces a better match between the two distributions. Bhattacharyya et al. [9] propose Conditional Flow Variational Autoencoders (CF-VAE), a novel variant of CVAE. In this variant, the latent prior is transformed by layers of conditional non-linear normalizing flows to a more complex prior distribution. Thus, a new regularization method of CVAE is deployed to train the CF-VAE. Both the variational posterior distribution and the conditional prior are jointly optimized. The variational posterior distribution tries to match the conditional prior and vice-versa. The CF-VAE with visual scene information is applied for multi-modal trajectory prediction and seems to have a good performance compared to the regular CVAE.

Generative models produce multiple plausible trajectories adapted to new situations (not encountered during the training) by sampling from random latent variables. But

they have two obvious limitations: The first one is the difficulty of deciding about the number of samples required to cover all the possible motions in practice. The other limitation is that the generated trajectories are considered equal without any information about the probability of each one.

2.3.1.4 Attention Mechanism for Trajectory Prediction

Most of the encoder and decoder blocks are composed of RNNs and/or CNNs. In this part, we present the attention mechanism deployed as a building block of an encoder and/or decoder and next in 2.3.2.2, we will present how it is used for interaction modeling.

Attention mechanism was introduced by Bahdanau et al. [6] for natural language processing purposes. It directly relates distant features, learns dependencies and estimates the relative importance of the keys' vectors with regard to the query. Therefore, it enables the network to put more focus on the relevant keys relative values. The Transformer [107] is an encoder-decoder architecture used for sequence-to-sequence learning. It replaces RNNs, usually deployed for sequence modeling, and it is composed mainly of Multi-Head Attention blocks mounted on successive layers. As opposed to recurrent models, which read the input sequentially, the transformer encoder reads the entire input sequence at once. In order to preserve the positional information of the Transformer's input sequence, positional encoding is applied on this sequence embedding. In contrast to the RNN encoder that outputs a single vector, the Transformer encoder maps the input sequence to a sequence of latent vectors. It consists of a stack of identical layers. Each layer is composed of two sub-layers (Figure 2.9). The first one is multi-head self-attention mechanism where multiple attention heads jointly attend to information from different representations at different positions. In the second sub-layer, all aggregated features are combined using a position-wise fully connected feed-forward network and fed to the next layer or to the decoder. The latter receives the encoded sequence and generates an output. It also consists of a stack of identical layers. In addition to the components of each encoder layer, the decoder has a third sub-layer consisting of multi-head attention over the encoder's output, in which every position in the decoder attends over the input sequence.

Bidirectional Encoder Representations from Transformers (BERT) [26] is derived from the regular Transformer and has been recently employed in many NLP tasks. It is only composed of a Transformer encoder and it trains and infers using a masking mechanism.
The Transformer and BERT have been recently deployed for trajectory prediction [32, 116]. Giuliari et al. [32] employ both the Transformer and the BERT networks. They

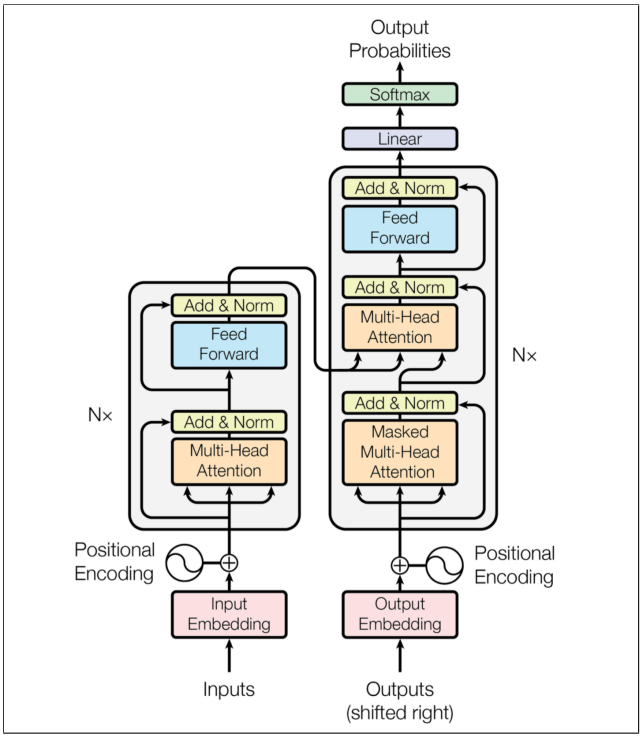


Figure 2.9: The Transformer model architecture [107]

use the transfer encoder to encode each agent's past trajectory and the decoder to predict its future motion without considering agents or scene interactions. Similarly, they deploy a BERT encoder and they show that the Transformer performs better than BERT for trajectory prediction. Differently, Yu et al. [116] account for agents interactions and present each set of surrounding agents as a graph. They tackle the task of the spatio-temporal modeling of graph sequences with a Transformer architecture. They use attention as the only building block of their STAR (Spatio-Temporal grAph tRansformer) framework for trajectory prediction. In the STAR framework, each agent's motion is modeled with a temporal Transformer (instead of the RNN used in most recent trajectory prediction studies), which captures the temporal dependencies. They also use a spatial Transformer to model agents' interactions. The STAR framework extracts the spatio-temporal interactions among agents by interleaving between the spatial Transformer and the temporal Transformer.

The main advantage of Transformers is their fast training since they largely perform their computations in parallel. However, their abilities to model the recurrency in the input sequence and to generalize to inputs not encountered during training are questioned [22, 104].

2.3.2 Agent Environment Interactions

Each agent's motion depends on the surrounding agents' recent motions and the scene around them. Modeling these spatio-temporal dependencies is a task that many state-of-the-art studies have approached using different methods.

2.3.2.1 Implicit Interaction Modeling

One of the most important parts in a driver intention prediction model is the surrounding vehicles' interaction extractor. It is also conceived differently in the state-of-the-art. Some existing studies [2, 24, 50, 61, 82] implicitly infer the dependencies between vehicles. They feed a sequence of surrounding vehicles' features as inputs to their model. Then, they assign to the LSTM the task of learning the influence of surrounding vehicles on the target vehicle's motion. However, the LSTM doesn't perform well in capturing both spatial and temporal dependencies. Therefore, more explicit interaction modeling methods are deployed.

2.3.2.2 Explicit Interaction Modeling

Other approaches explicitly model the vehicles' interactions using several combinations of networks.

Social Pooling

Alahi et al. [1] introduced the social LSTM concept for the pedestrian trajectory prediction task. They encode the motion of each agent using an LSTM block. Then, they extract the interactions between agents by sharing the hidden states between all the LSTMs corresponding to a set of neighboring pedestrians (cf. further details about social pooling in Section 3.2.4.1). Hou et al [39] use a structural-LSTM network to learn high-level dependencies between vehicles. Similarly to social LSTM, they attribute one LSTM to each vehicle. The spatial-neighboring LSTMs share their cell and hidden states by a radial connection; the output states of the LSTMs are treated recurrently in a deeper layer. Finally, the decoder receives the outputs of the latter and generates all the predicted trajectories.

Convolutional Neural Networks for Interaction Modeling

Deo et al. [23] extend the social pooling technique and deploy it for the vehicle trajectory prediction task. They use an LSTM encoder to generate a representation of each vehicle's trajectory. Then, they use convolutional layers applying successive local operations on the outputs from the encoders followed by a maxpool layer. Therefore, they generate a

context vector that consists of a compact representation of the vehicles' interactions. But successive local operations are not always sufficient. Furthermore, the generated context vector is independent of the target vehicle's state (cf. further details about social pooling in Section 3.2.4.2). Zhao et al. [120] extend convolutional social pooling to simultaneous multi-agent trajectory prediction. They use an LSTM to generate trajectories encodings of the surrounding agents. They apply a CNN on the bird's-eye view image of the scene to extract the context features. Then, they place the trajectories encodings on top of the extracted features grid based on their positions to form a combined representation of the context. This representation is fed to a CNN that generates the trajectory of each of the agents considered. Subsequent studies [16, 21, 28, 38, 81] feed a CNN with a 3D tensor combining scene map image and additional information about surrounding agents' past motions. They assign to the CNN the task of simultaneously capturing the spatio-temporal dependencies required to predict the future motion of the target agent. The limitation of using only a CNN for trajectory prediction is that it doesn't explicitly model the temporal evolution of the agents' motions. Ridel et al. [85] propose a more explicit way to model both the spatial and temporal evolution of the agents in the scene using a convLSTM.

Attention Mechanism for Interaction Modeling

Attention mechanism (cf. Section 2.3.1.4) is known for its ability to model dependencies between features. Sadeghian et al. [90] use an LSTM to model the past motion of the target agent and use its hidden state to attend to the salient regions top-view image of the navigation scene at each time step. However, they don't model interactions between agents. In a subsequent study [89], they use an LSTM encoder to encode the trajectories of surrounding agents and two attention blocks. The first attends to salient features in the scene context. The second block, denoted social attention, extracts agents' important features based on their motions encodings. However, they don't directly capture the relation between the target agent and the surrounding agents since they use the soft attention mechanism [6]. Similarly, Amirian et al. [3] use the social attention block to model agents' interactions.

In our work [68, 71], we were the first to use the multi-head attention mechanism (MHA) [107] to model the interactions between agents in the task of trajectory prediction. Each attention head learn to capture a specific aspect of dependencies between the surrounding agents. Subsequent works [51, 67] deploy MHA to jointly forecast the trajectories for all vehicles of the scene. While Kim et al. [51] use a simple encoder-decoder architecture based on MHA, Mercat et al. [67] deploy two MHA layers; the first is mounted after the trajectories encoding to incorporate interactions between vehicles at the current time. The second MHA layer is added after the prediction step at forecasting time.

Yu et al. [116] employ Transformers (TFs) to model the spatio-temporal evolution of the agents' motions. They alternate between temporal and spatial TFs. In this thesis, we investigate the use of MHA and TFs for interaction modeling with the surrounding context in the trajectory prediction framework.

Graph Neural Networks

Graph neural networks (GNNs) are widely employed in relational reasoning. Many recent studies [43, 54, 91, 110] use it in the trajectory prediction task in order to model the agents' interactions. They present context as a graph, where agents' states are nodes and their relationships are edge features. In addition to the spatial edges, Vemula et al. [110] attribute a temporal edge to each node to present the temporal evolution of each agent's motion. They associate an RNN to each node and edge (Figure 2.10). Then, they use an attention module to compute a dot product attention between the temporal RNN and the neighboring spatial RNN hidden states for each node. The computed attention vector is fed to the node RNN that sequentially generates the predicted trajectory.

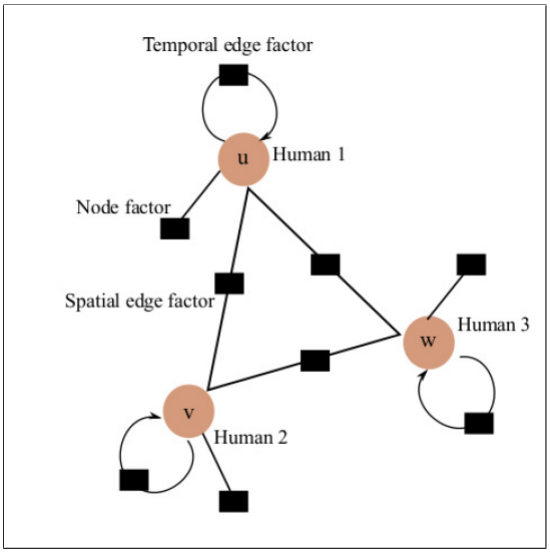


Figure 2.10: Example of spatio-temporal graph [110]

Salzmann et al. [91] deploy a directed graph where edge information is element-wise summed from neighboring agents of the same class at each time step. Then, the aggregated features are fed to an LSTM with weights shared across all edges of the same type. The encodings from all edge types connected to the target agent node are aggregated using an additive attention to obtain a context representation vector. Huang et al. [43] use an LSTM encoder to generate a spatio-temporal representation of each agent's motion while considering the interactions between agents. They use Graph Attention Network

(GAT) [109] over the hidden states of the encoder at each time step and feed it as input to the encoder at the next time step. In all previous methods, spatial interactions are captured by the graph at each time-step, which is computationally expensive. Kosaraju et al. [54] propose using an LSTM to encode agents' trajectories and then they apply GAT to model the interactions between agents only at the prediction time.

2.4 Summary of Trajectory Prediction Deep Learning based Studies

A trajectory prediction study consists first in selecting the input cues that reveal information about the future motion of the target agent. Table 2.1 shows that state-of-the-art studies use different forms of the input cues. Most of the studies rely on information about the target agent's past trajectory (past traj.) [1, 2, 23, 32, 35, 50, 59, 74, 76, 89–91, 116, 120]. Additional information about surrounding agents is used as input to the network in order to be able to make interaction aware predictions [1, 2, 16, 21, 23, 32, 35, 59, 76, 85, 89, 91, 116, 120]. Static scene elements presented as bird's-eye view (BEV) scene images or high fidelity maps are also exploited to generate predictions more compliant with the scene structure [16, 21, 59, 85, 89–91, 120]. We notice that the best performance is achieved when using the target agent's past motion, surrounding agents' information and scene elements [16, 21, 59, 85, 89, 91, 120].

The second step in trajectory prediction is extracting salient features from the inputs. To do so, different networks are used to model input trajectories as time series: LSTMs [1, 2, 50, 90], LSTMs auto-encoders [23, 35, 59, 76, 89, 91, 120], temporal convolutions (temp. CNN) [74], transformers (TF.) [32, 116]. The most deployed ones are LSTMs and LSTM auto-encoders since they are mainly designed to model the temporal dependencies between the elements of an input sequence. An LSTM or an LSTM autoencoder can be deployed in two different ways. It is either fed with a sequence of vectors composed of the past states of the target agents and the surrounding ones, and therefore it generates a vector representing all the agents considered [2, 76]. It can also receive a sequence of each agent's trajectory separately and produce a representation vector for each agent [1, 23, 35, 59, 89, 91, 120]. We note the latter LSTMs as S-LSTMs. Temporal convolutions are also employed [74] and they achieve competitive results with the LSTM-based models while reducing inference time. Transformers have been recently introduced and deployed in time series modeling [32, 116]. They achieve more parallelization in sequence training than LSTMs. Since static scene information is mainly represented as images, CNNs are usually used to extract salient features from these images [59, 85, 89–91].

Next, in some studies [1, 23, 35, 59, 74, 89–91, 116, 120] the dependencies between

Table 2.1: Summary of Trajectory Prediction Approaches

Methods	Input Representation			Feature Extraction			Interaction		Generative	Output		Dataset
	Target	Agents	Scene	Target	Agent	Scene	Agent	Scene		Form	Multi-modal	
[50]	past traj.(pos., vel.)	No	No	LSTM	No	No	No	No	No	grid	+ Proba.	private
[2]	past traj. (pos., vel.)		No	LSTM		No	No	No	No	traj.	No	NGSIM
[76]	past traj. (pos., vel.)		No	LSTM auto-enc.		No	No	No	No	grid	+ Proba.	private
[1]	past traj. (pos.)		No	S-LSTM		No	Add.	No	No	traj.	No	ETH [79], UCY [62]
[23]	past traj.		No	S-LSTM auto-enc.		No	CNN	No	No	traj. + man.	+ Proba.	NGSIM
[35]	past traj.		No	S-LSTM auto-enc.		No	FC.	No	GAN	traj.	Yes	ETH [79], UCY [62]
[120]	past traj.		BEV image	S-LSTM auto-enc.		CNN	CNN		GAN	traj.	Yes	NGSIM, ETH [79], UCY [62]
[90]	past traj. (pos.)	No	BEV image	LSTM	No	CNN	No	Attn.	No	traj.	No	private
[89]	past traj. (pos.)		BEV image	S-LSTM auto-enc.		CNN	Attn.	Attn.	GAN	traj.	Yes	ETH [79], UCY [62], SDD [86]
[59]	past traj.		map	S-LSTM auto-enc.		CNN	Attn	Concat.	CVAE	traj.	Yes	KITTI [31], SDD [86]
[91]	past traj. (pos., vel.)		map	S-LSTM auto-enc.		CNN	Graph	Concat.	CVAE	traj.	+ Proba.	ETH [79], UCY [62], nuScenes
[16, 21]	target past state + rasterized (map + agents)			CNN					No	traj.	+ Proba.	private, CARLA [29], SDD [86]
[74]	past traj. (pos.)	No	No	temp. CNN	No	No	No	No	No	traj.	No	ETH [79], UCY [62]
[85]	occupancy grid		BEV image	U-Net		ResNet	convLSTM		No	traj.	+ Proba.	SDD [86]
[32]	past traj. (pos.)	No	No	TF.	No	No	No	No	No	traj.	No	ETH [79], UCY [62]
[116]	past traj. (pos.)		No	TF.		No	TF.	No	No	traj.	No	ETH [79], UCY [62]
[69]	past traj. (pos., vel., acc.)		No	RRNN		No	RRNN	No	No	traj.	No	NGSIM, highD
[68, 71]	past traj. (pos., vel., acc.)		No	S-LSTM		No	MHA	No	No	traj.	+ Proba	NGSIM, highD
[70]	past traj. (pos., vel., acc.)		map	S-LSTM		CNN	MHA		No	traj.	+ Proba	nuScenes

We denote trajectories (traj.), position (pos.), velocity (vel.), acceleration (acc.), bird's-eye view (BEV), auto-encoder (auto-enc.), social LSTM (S-LSTM), temporal CNN (temp. CNN), transformer (TF.), relational recurrent networks (RRNN), additional pooling (Add.), fully connected network (FC.), attention mechanism (Attn.), features concatenation (Concat.), maneuvers (man.), probability (Proba.)

the target agent, the surrounding agents and the map are explicitly modeled using the features extracted in the previous step and exploited to generate a compact representation of the context (context encoding). Interactions with surrounding agents are modeled using additional pooling (Add.) [1], fully connected layer (FC.) [35], CNN [23, 120], attention mechanism (Attn.) [59, 89, 90] and Transformers (TF.) [116]. Additional pooling (Add.) and fully connected layer (FC.) fail to adequately model the spatial locations of the agents. CNNs consider the configuration of the agents in the scene. However, they produce agents' encoding independently of the target agent's motion. Attention mechanism and Transformers are more efficient in dependency modeling since they directly relate distant agents' features and evaluate their relation.

Interactions with the scene elements are represented using attention mechanism [89, 90] or by simply adding the map extracted feature vector to the context encoding [59, 91].

The interactions with nearby agents and the scene can also be modeled simultaneously using CNN [120] or convLSTM [85]. Simultaneous interaction modeling tends to improve the trajectory prediction since it implicitly represents the dependencies between the surrounding agents and the scene.

The previous two steps: feature extraction and interaction modeling are combined in some studies [16, 21] using a CNN, which receives a rasterized representation of the scene and agents. This representation exhibits the temporal evolution of agents' states as a sequence of bounding boxes. A CNN does not explicitly model the temporal dependencies in the agents motions, which can induce a degradation in the prediction performance.

The final step is motion prediction. It consists in generating the predicted intention of the target agent in the form of a trajectory, maneuvers (man.) or occupancy grid. In order to account for the uncertainty of the future, multi-modal predictions are deployed in [16, 21, 23, 35, 50, 59, 76, 85, 89, 91, 120]. Multiple trajectories are generated using different methods such as generating samples using generative approaches (GAN, CVAE) [35, 59, 89, 91, 120]. The main advantage of these approaches is their ability to generalise to unseen situations. However, they are unable to infer the probability of each generated sample in most cases [35, 59, 89, 120]. Therefore, other probabilistic approaches [16, 21, 23] that generate multiple possible trajectories with their corresponding probabilities are deployed to better handle the prediction task.

Table 2.1 also shows our main contributions [69–71]. They were developed concurrently with many of the state-of-the-art studies described above.

Table 2.2: Overview of the motion trajectories datasets

Dataset	Location	Sensors	Scene description	Duration	Tracks	HD maps	Rate	Release year
NGSIM [44, 45]	Highways	Fixed camera network	US Highway 101, Interstate 80	90 min	–	None	10 Hz	2007
highD [55]	Highways	Fixed camera (Drone)	6 different locations, 60 recordings	16.5 h	110500	None	25 Hz	2018
nuScenes [13]	Urban	Dynamic camera, Lidar	1000 scenes, each	6 h	40000	Yes	2 Hz	2020
Lyft 5 [41]	Urban	Dynamic camera, Lidar	170000 scenes, each 25 sec	1118 h	–	Yes	10 Hz	2020
ApolloScape [111]	Urban	Dynamic camera, Lidar	103 scenes	2h	81800	Yes	10 Hz	2019
Argoverse [17]	Urban	Dynamic camera, Lidar	324000 scenes	320h	324557	Yes	10 Hz	2019
Interaction [118]	Urban	Fixed camera (Drone)	11 locations	16.5 h	40054	Yes	10 Hz	2020
inD [10]	Urban (Intersections)	Fixed camera (Drone)	4 locations	10 h	11500	None	25 Hz	2020
rounD [56]	Urban (roundabouts)	Fixed camera (Drone)	3 locations	6 h	13 746	Yes	25 Hz	2020

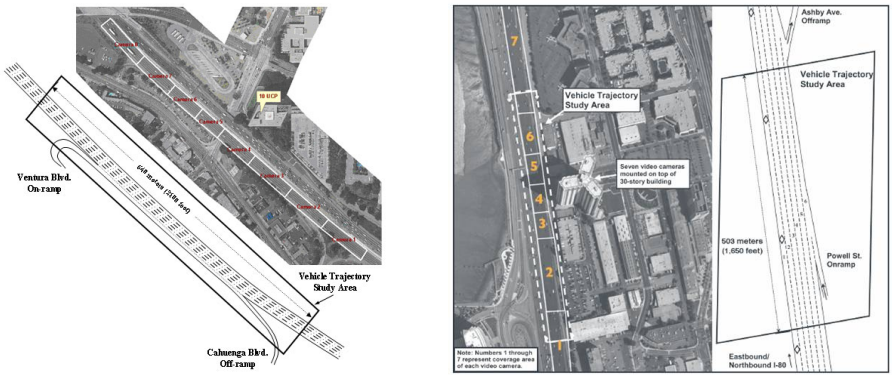
2.5 Datasets

Many naturalistic vehicle trajectory datasets of varying sizes and characteristics have recently been released. They have have been recorded in different environments (highways/ urban) using different types of sensors (Camera/ Lidar/ radars..) deployed in different manners (fixed in a drone, dynamic). Table 2.2 presents the different characteristics of the existing vehicle trajectory datasets. In our work, we aim to tackle the trajectory prediction task in different types of environments. We selected the datasets according to various criteria; the environment (highway, urban), the size of the data, the placement of the

deployed sensors (fixed, dynamic), the data provided (trajectories, velocities, maps), etc. We started our experiments using datasets recorded on highways (NGSIM [44, 45] and highD [55]). Then, we exploited a dataset of an urban environment. Trajectory prediction in an urban environment requires ample information about the scene structure and road geometry. Thus, we need a dataset with a high definition map. Furthermore, we opted for a dataset captured using sensors mounted on a vehicle navigating different scenes since it offers a more realistic representation of the trajectory prediction task and makes it possible to train models capable of dealing with different prediction scenarios. Therefore, we select nuScenes dataset. Moreover, its data format enables us to make relatively long term (prediction horizon greater than 5 seconds) trajectory predictions compared to Argoverse [17] (prediction horizon of 3 seconds). It also has a fair size of recorded data compared to the ApolloScape [111] dataset.
In the following, we further describe the datasets that we use in our experiments.

2.5.1 Next Generation Simulation (NGSIM)

NGSIM [44, 45] is a large, publicly available dataset captured in 2005 at 10Hz recording US Highway 101 and Interstate Freeway 80. It presents mild, moderate and congested real freeway traffic conditions with mainly two types of vehicles (cars, trucks). It is widely studied and used in the literature, especially for the task of future intention prediction of vehicles [2, 23, 24, 61, 82]. We use this dataset to compare our model with the state-of-the-art (Figure 2.11).



(a) US Highway 101 **(b)** Interstate Freeway 80

Figure 2.11: NGSIM dataset

We adopt the same data split into train (75%) and test (25%) sets as Deo et al. [23] and many other subsequent studies. We split the trajectories into segments of 8s of the trajectories composed of a track history of 3s and a prediction horizon of 5s. We downsample each segment to get only 5 fps to reduce the complexity of the model.

2.5.2 highD

highD [55] was captured in 2017 and 2018. It was recorded by camera-equipped drones from an aerial perspective of six different German highways at 25 Hz. It is composed of 60 recordings of about 17 minutes each, covering a segment of about 420m of two-way roads (Figure 2.12).

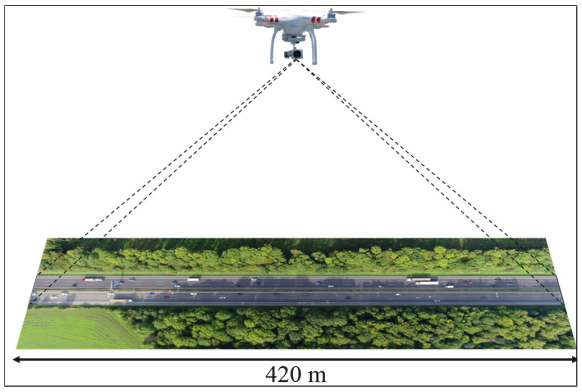


Figure 2.12: Highway drone dataset highD [55]

highD consists of vehicle position measurements from six different highways with 110 000 vehicles (about 12 times as many vehicles as NGSIM) and a total driven distance of 45 000 km. This dataset is of great importance since it has 5 600 recorded complete lane changes and presents recent driver behaviors. It presents different traffic scenarios and conditions since it is recorded at different times of the day and different recording sites.

2.5.3 nuScenes

The nuScenes [13] dataset was captured using vehicle mounted camera, radar and lidar sensors driving through Boston and Singapore. It comprises 1000 *scenes*, each of which is a 20 second record, capturing complex inner city traffic with different agent types, such as pedestrians, bicycles, cars, trucks, etc. Each scene includes agent detection boxes and tracks hand-annotated at 2 Hz, as well as high definition maps of the scenes. These scenes present a diverse set of locations, times and weather conditions.
An official benchmark split is imposed for the nuScenes prediction challenge, with 32,186 prediction instances in the train set, 8,560 instances in the validation set, and 9,041 instances in the test set.

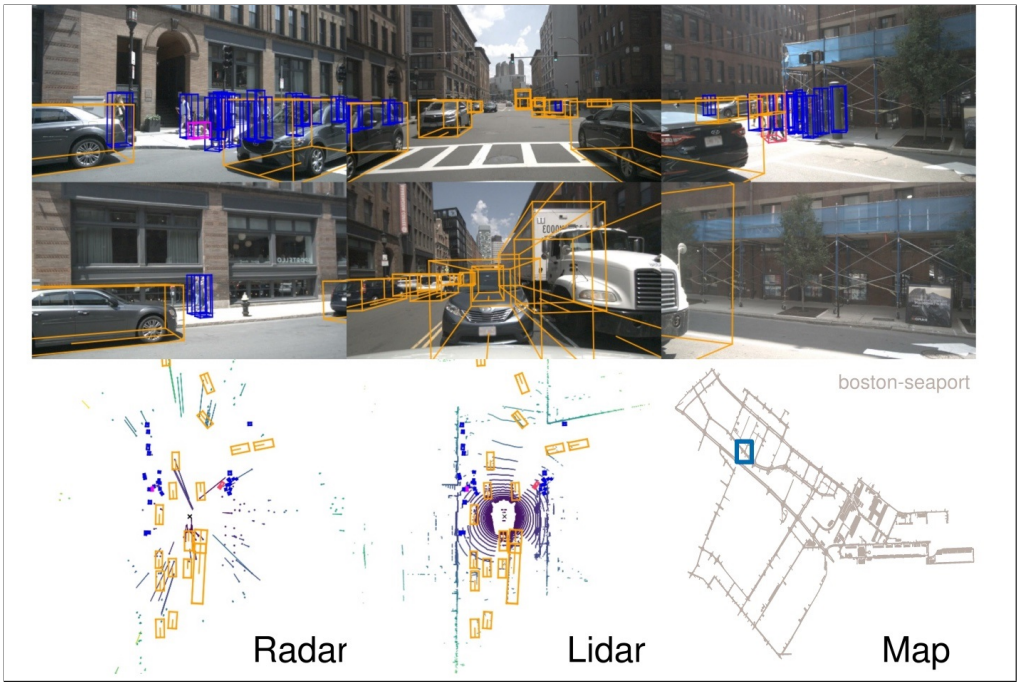


Figure 2.13: nuScenes [13] dataset

3

Interaction Aware Trajectory Prediction on Highways

Contents

3.1 Problem Definition 41
3.2 Overview 41
3.2.1 LSTM Encoder-Decoder 42
3.2.2 Multi-Head Attention Mechanism 44
3.2.3 Brief Description of our Proposed Methods 47
3.2.4 Related Work 47
3.3 Relational Recurrent Neural Networks based Methods (L-RRNN and Sc-RRNN) 50
3.3.1 Overall Model 50
3.3.2 Inputs Representation 51
3.3.3 Inputs Embedding 52
3.3.4 Relational Recurrent Encoder-Decoder 53
3.3.5 Multi-Head Dot Product Attention (MHDPA): 53
3.3.6 Introducing MHA into an LSTM 54
3.3.7 Implementation Details 55
3.4 Multi-head Attention based Method (MHA-LSTM) 55
3.4.1 Overall Model 56
3.4.2 Trajectory Encoder 57
3.4.3 Vehicle Interaction Modules 57
3.4.4 High Order Interaction 59
3.4.5 Trajectory Generation 60
3.4.6 Another Variant: Local and Non Local Social Pooling 60
3.4.7 Implementation Details 62
3.5 Experimental Evaluations 62
3.5.1 Evaluation Metric 62

3.5.2 Models Compared 63
3.5.3 Quantitative evaluation 64
3.5.4 Qualitative Analysis of Predictions 68
3.6 Conclusion 71

In this chapter, we tackle the task of trajectory prediction in a highway. For this purpose, we consider the features that characterise the driver motion. Vehicles' past states give important information about its motion dynamics since the future trajectory is a continuation of the past one. However, the trajectory taken by each vehicle in the future is not only dependent on its own state history, the presence and actions of the neighboring vehicles have a great influence on a vehicle's behavior as well. Therefore, we propose to model the interactions between the neighboring vehicles to represent the most relevant information about the social context with a capability of learning long-range relations. In our approach, we attempt to mimic human reasoning, which pays a selective attention to a subset of surrounding vehicles in order to extract the elements that most influence the target vehicle's future trajectories while paying less attention to other vehicles. For example, a vehicle performing a lane change maneuver will pay more attention to the vehicles in the target lane than those in the other lanes. Consequently, its future behavior could be more dependent on distant vehicles in the target lane than the close ones in the other lanes.
As mentioned in Section 2.3, most of interaction aware trajectory prediction frameworks are generally composed of three main modules: an input feature encoder, an interaction modeling block and a trajectory generator. This is the case with our proposed methods. These three modules can be deployed separately or combined. In this chapter, we present two different methods to tackle the task of trajectory prediction:

- The first method, presented in Section 3.3, combines feature extracting and interaction modeling. It is based on applying simultaneous spatial and temporal dependencies modeling at each time step based on relational recurrent neural networks (RRNNs). In addition, we form two variants of this method; scene RRNN (Sc-RNNN) and lane RRNN (L-RRNN) using two different input representations. Then, we evaluate the performance of each one.

- The second method, denoted (MHA-LSTM) and described in Section 3.4, has the same building blocks than the first one. But, they are mounted in a more optimal way so that the prediction performance is improved. Here, we use two separate modules for feature extraction and interaction modeling.

Before presenting these method, we will describe the problem we want to solve and the main NN building blocks used in our methods.

3.1 Problem Definition

The goal of this chapter is to predict the future trajectory of a target vehicle T, knowing its past states and the past states of its neighboring vehicles at observation time t_{obs}.
We have as input the past tracks of the target and a set of its neighboring vehicles. The input tracks of a vehicle i are defined as $\mathbf{X}_i = [\mathbf{x}_i^1, \ldots, \mathbf{x}_i^{t_{obs}}]$ where $\mathbf{x}_i^t = (x_i^t, y_i^t)$ is the state vector. We note $\mathbf{X}_T$ the state of the target vehicle T.
The coordinates of all the considered vehicles, are expressed in a stationary frame of reference where the origin is the position of the target vehicle at time t_{obs}. The $y\text{-}axis$ and $x\text{-}axis$ point respectively to one direction of motion on the highway and to the direction perpendicular to it.
We also consider additional information about the considered vehicles such as velocity, acceleration and type ($\mathbf{x}_i^t = (x_i^t, y_i^t, v_i^t, a_i^t, type)$) in some experiments.
In order to model the uncertainty in the trajectory prediction, our model outputs the parameters characterizing a probability distribution over the predicted positions of the target vehicle.

$$\mathbf{Y}_{pred} = [\mathbf{y}_{pred}^{t_{obs}+1}, \ldots, \mathbf{y}_{pred}^{t_{obs}+t_f}]$$

Where $\mathbf{y}^t = (x^t, y^t)$ is the predicted coordinates of the target vehicle.
Our model infers the conditional probability distribution $\mathbf{P}(\mathbf{Y}|\mathbf{X})$.
The distribution over the possible positions at time $t \in \{t_{obs}+1, \ldots, t_{obs}+t_f\}$ can be presented as a bivariate Gaussian distribution with the parameters $\Theta^t = (\mu^t, \Sigma^t)$ of the form:

$$\mathbf{y}^t \sim \mathcal{N}(\mu^t, \Sigma^t)$$

Where μ^t is the mean vector and Σ^t is the covariance matrix:

$$\mu^t = \begin{pmatrix} \mu_x^t \\ \mu_y^t \end{pmatrix}, \Sigma^t = \begin{pmatrix} (\sigma_x^t)^2 & \sigma_x^t \sigma_y^t \rho^t \\ \sigma_x^t \sigma_y^t \rho^t & (\sigma_y^t)^2 \end{pmatrix}$$

Before presenting our proposed methods, described in Sections 3.3 and 3.4 and deployed to resolve the above problem, we introduce the main state-of-the-art methods that inspired our work. In particular, we describe the main NN building blocks that we used in our models: LSTMs and MHA mechanism.

3.2 Overview

Knowing the performance of LSTMs in sequence modeling and the power of attention mechanism to capture long range dependencies, we propose methods composed of two main building architectures. The first is the LSTM encoder decoder deployed for temporal dependency modeling. The second is the multi-head attention (MHA) mechanism. In the following, first, we describe the LSTM encoder decoder architecture as well as the MHA mechanism. Then, we give the intuition behind each of our proposed methods.

3.2.1 LSTM Encoder-Decoder

The long short-term memory (LSTM) is a special variant of RNN widely used for time series modeling. It resolves the vanishing gradient problem of RNNs and therefore it is characterized by an increased ability of learning long-term dependencies. The LSTM is composed of a chain of neural networks repeating modules. The key element of an LSTM is the cell memory that passes through the entire LSTM chain and stores the salient information about the past input sequence. Gating mechanisms are deployed to control the flow of information between the input, output, and cell memory (cf. Figure 3.1).

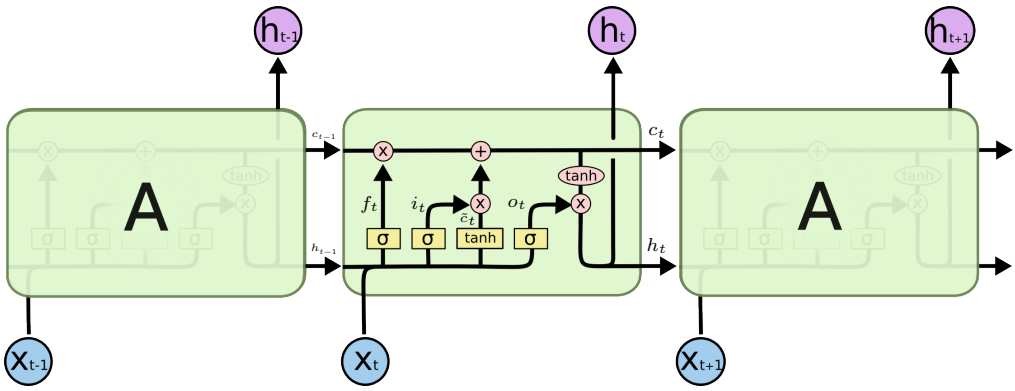


Figure 3.1: Regular LSTM

$$f_t = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + b_f) \tag{3.1}$$

$$i_t = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + b_i) \tag{3.2}$$

$$\tilde{c}_t = \tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c) \tag{3.3}$$

$$o_t = \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + b_o) \tag{3.4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.6}$$

Where:

- $\sigma(x) = \frac{1}{1+exp(x)}$: sigmoid function.
- $x \odot y$: element wise product.
- x_t: input vector.
- $W_{xf}, W_{hf}, W_{xi}, W_{hi}, W_{xo}, W_{ho}, W_{xc}, W_{hc}$: linear transformation matrices.

- b_f, b_i, b_o, b_c: bias vectors.
- i_t, f_t, o_t: gating vectors.
- c_t: cell memory state vector.
- h_t: hidden state vector.

The LSTM operates by updating the cell memory c_t based on the input vector x_t, the previous cell memory c_{t-1} and hidden state vector h_{t-1} and then generating the hidden state h_t using the resulting cell memory c_t. To do so, gates composed of a sigmoid NN layer and a pointwise multiplication are applied on the input vector x_t and the previous hidden state vector h_{t-1} to generate the input i_t, the forget f_t and the output o_t gating vectors (cf. Equations 3.1, 3.2 and 3.4). Then, the input and the forget gating vectors update the cell memory by removing unnecessary information from the previous cell memory c_{t-1} or adding information based on the input vector x_t and the previous hidden state vector h_{t-1} (cf. Equation 3.5). Finally, the output gating vectors filters the cell state c_t in order to output the current hidden state h_t (cf. Equation 3.6).
Sutskever et al. [99] extend the use of the LSTM and introduce the sequence to sequence concept (named also sequential encoder decoder or auto-encoder architecture). It consists in mapping a fixed-length input with a fixed-length output.
Trajectory prediction can be considered as a sequence to sequence learning task that maps past trajectory to the future one. Recent studies [24, 72, 76] use an LSTM based encoder decoder architecture to model the spatial interactions between neighboring vehicles. Indeed, Park et al. [76] employ the encoder-decoder architecture in order to extract salient information about the past trajectory pattern and generate the future trajectory sequence (cf. Figure 3.2).

In their work [76], the LSTM encoder receives the trajectory of a neighboring vehicle and generates a compact vector which represents the salient information of its past motion dynamics. The LSTM decoder receives the encoding vector and recursively produces its future trajectory.

LSTMs perform well in sequence modeling. However, they lack the spatio-temporal structure to capture both, the temporal evolution and the spatial interactions between vehicles in the driving scene. As a remedy, we propose the use of a new architecture based on human like reasoning which selectively focuses attention on a subset of surrounding vehicles motion features and efficiently retain pieces of information that probably influence his future trajectory.

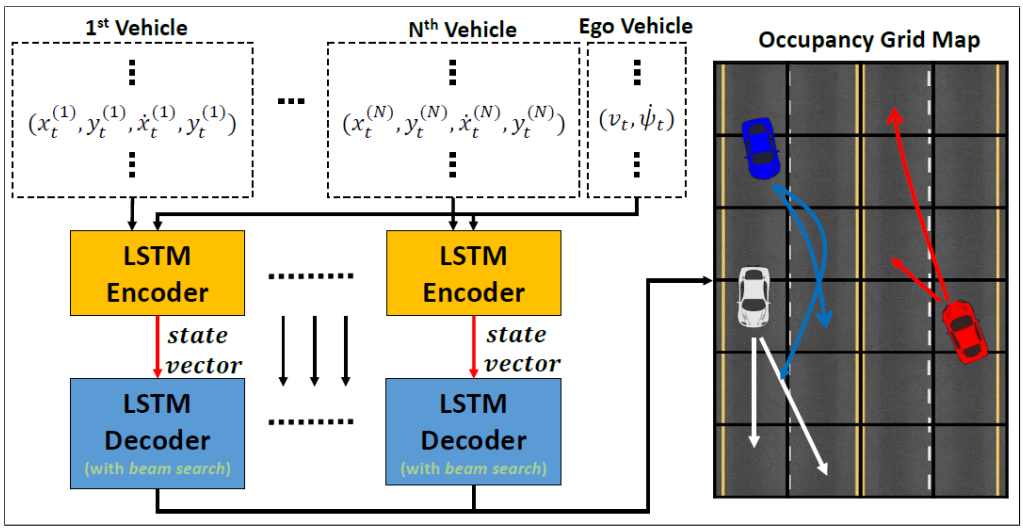


Figure 3.2: LSTM encoder decoder [76]

3.2.2 Multi-Head Attention Mechanism

MHA was introduced by Vaswani et al. [107] for NLP purposes. They completely replaced the recurrent blocks with MHA to model the dependencies between sequence elements. MHA is currently integrated with different NN architectures such as CNNs [112] and GANs [119] and deployed for different computer vision models. MHA is also integrated into the LSTM cell [92] in order to enhance its ability to perform relational reasoning on the input sequence.

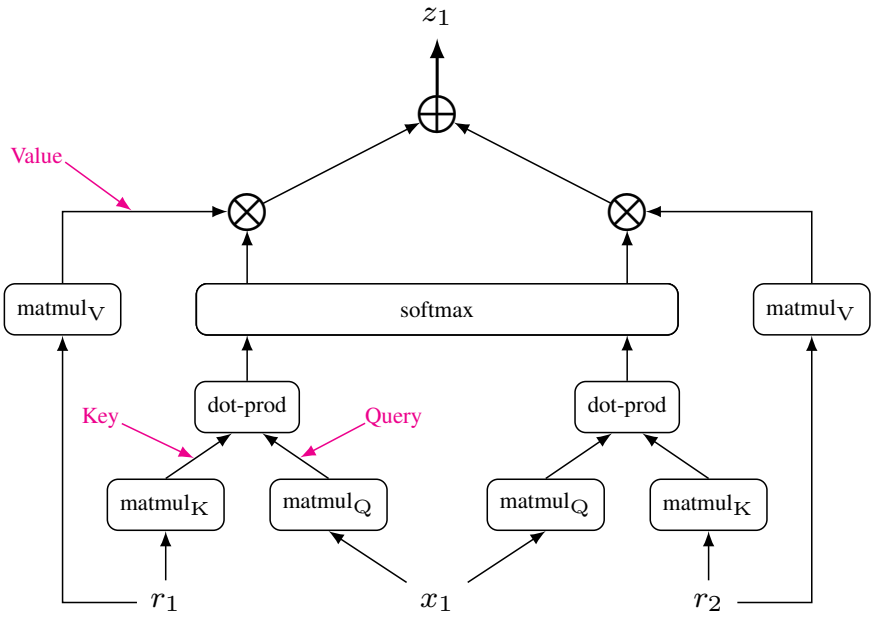


Figure 3.3: Computation of One Attention Head Explained: Attention over the vectors r_1 and r_2 conditioned on x_1.

The attention mechanism's main advantage lies in its capability of learning high quality features by considering the whole context and directly performing a relational reasoning between both local and distant features. This relational reasoning enables to evaluate the importance of each feature vector. Then, attention mechanism aggregates features based on their importance. Figure 3.3 summarizes the main steps and operations of computing the dot product attention over the vectors r_1 and r_2 conditioned on the vector x_1. First, x_1 is projected to form the Query vector, while r_1 and r_2 are projected in two different spaces to form the Keys and Values vectors. The dot product of the Query and Keys followed by the $Softmax$ operation determine the importance or the weight of each of the Values corresponding to r_1 and r_2. The attention over r_1 and r_2 conditioned on x_1 is the sum of the Value vectors weighted by their corresponding weights. Further details about MHA computation are illustrated in Figure 3.4.

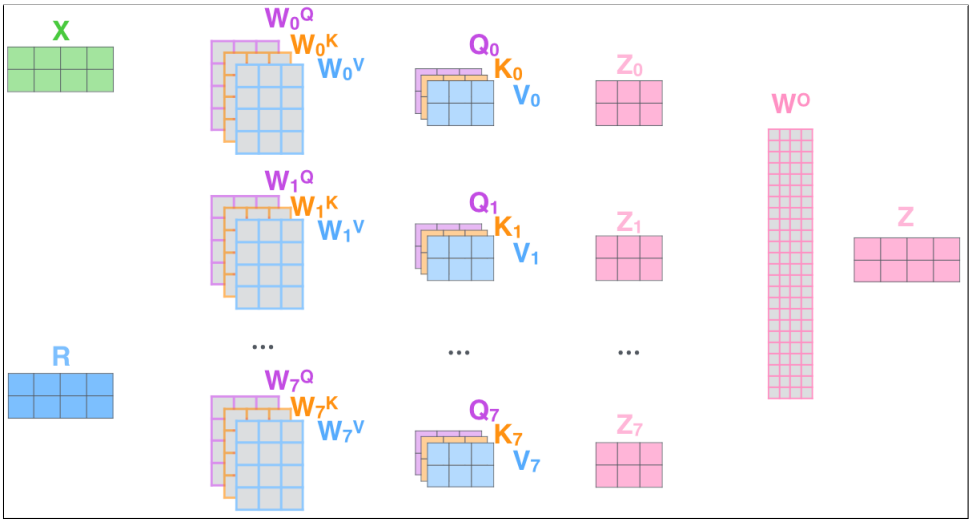


Figure 3.4: An example of computing MHA using two inputs X and R

Figure 3.4 illustrates an example of computing MHA using two inputs X and R. In this example, we want to aggregate features from the matrix R based on a relational reasoning between the matrices X and R. We denote $X = [x_1; x_2]$ and $R = [r_1; r_2]$ where x_i and r_i $(i = 1, 2)$ represent the row vectors of the matrices X and R respectively.

The first step in calculating attention is to create three matrices; Query matrix $Q_l = \left[q_1^l; q_2^l\right]$, Key matrix $K_l = \left[k_1^l; k_2^l\right]$, and Value matrix $V_l = \left[v_1^l; v_2^l\right]$. The Query vectors q_i^l represent the projection of input feature vectors x_i using the matrix W_l^Q while k_i^l and v_i^l are the projections of input feature vectors r_i using the matrices W_l^K and W_l^V respectively. The matrices W_l^Q, W_l^K and W_l^V are network parameters that we learn during the training process. Then, for each selected Query vector q_i^l, the attention mechanism generates attention weights α_{ij}^l, $(\alpha_{ij}^l \in \mathbb{R})$ that determine the relative importance of the Value

vectors $v_j^l,\ (j = 1, 2)$. These weights are computed by applying a Softmax function to the dot product operation on that Query vector and each of the Key vectors $k_j^l,\ (j = 1, 2)$ $i.e.\ \alpha_i^l = Softmax(\frac{q_i^l\ transpose(K^l)}{\sqrt{d}})$. Therefore, the correlation between the Query vector and a Key vector determine the relative importance of the corresponding Value vector. An attention vector z_i^l is generated for each Query vector q_i^l as a weighted sum of the Value vectors weighted by their corresponding attention weights $z_i^l = \sum_j \alpha_{ij}^l v_j^l$. Consequently, we construct the resulting matrix $Z_l = \left[z_1^l; z_2^l\right]$.

The key idea behind MHA is that instead of using only one attention mechanism to perform a relational reasoning, multiple attention heads are used (We denote n_h the number of attention heads, in this example $n_h = 8$). Indeed, multiple sets of Query, Key and Value matrices (Q_l, K_l and V_l respectively, $l = 0, \ldots, n_h - 1$) are generated by projecting the input features into multiple different spaces using different linear transformations (with parameters W_l^Q, W_l^K and W_l^V respectively, $l = 0, \ldots, n_h - 1$). The resulting matrices $Z_l = \left[z_1^l; z_2^l\right]$ are concatenated and projected using the matrix of learnt parameters W^O to generate a final attention matrix Z. The use of multiple attention heads enables to boost the performance of the relational reasoning by capturing a different aspects of dependencies between X and R with each attention head.
The three set of vectors Query Q_l, Key K_l and Value V_l can be generated from the same input feature matrices ($i.e.\ X = R$). In this case, we talk about self-attention.

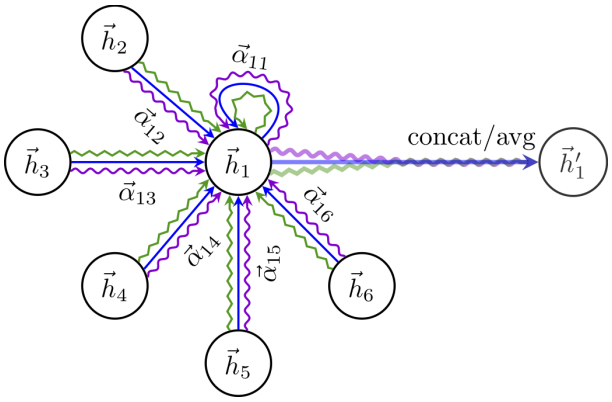


Figure 3.5: Graph Attention Network: The graph illustrates MHA (with 3 heads) applied by a node 1 on its neighborhood. The different arrow styles and colors correspond to the attention heads. Then, the resultant features from each head are concatenated or averaged to form the node 1 representation.

The dependencies and the interactions between the neighboring vehicles in the scene can be presented as a structured graph. In addition, Velickovic et al. [109] propose a Graph ATtention network (GAT) that applies MHA on graph-structured data to specify weights to nodes in a neighborhood. The GAT slightly modifies the MHA computation by using the concatenation attention instead of the dot product for example. Our utilisation

of MHA to model the interactions between vehicles (cf. Section 3.4) is comparable to the GAT applied to a target vehicle.

3.2.3 Brief Description of our Proposed Methods

Here, we briefly describe our proposed methods by presenting their main building blocks and the ways of connecting them. Our first method extends the basic LSTM encoder decoder architecture (cf. Figure 3.6).

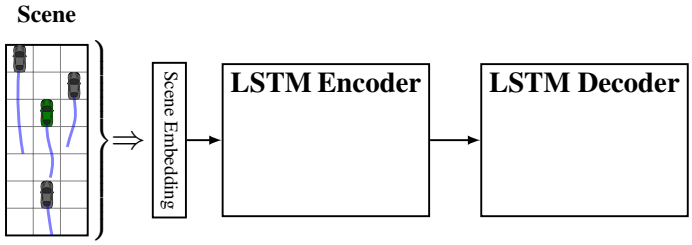


Figure 3.6: Basic LSTM Encoder Decoder Architecture

Our first method proceeds by simultaneously modeling spacial and temporal dependencies between agents motions at each past time step by integrating MHA inside of the LSTM block. However, it requires important computational resources since it performs relational reasoning at each past time step (cf. Figure 3.7).

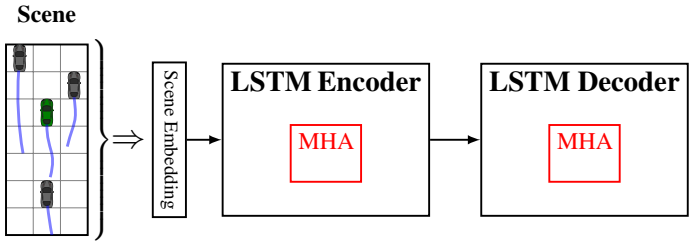


Figure 3.7: RRNN Architecture

In order to further optimize our first method, we maintain the same building blocks (LSTM and MHA) and mount them differently to form our second model (cf. Figure 3.8); First, we use the LSTM to encode the trajectories of the target vehicle and its surroundings. Then, we deploy the attention mechanism to derive the relative importance of surrounding vehicles with respect to their whole past motion only at the prediction time and generate a whole context vector.

3.2.4 Related Work

Our second method is inspired by two recent studies: social LSTM [1] and convolutional social pooling [23] which will be described more in depth in the next subsections 3.2.4.1 and 3.2.4.2.

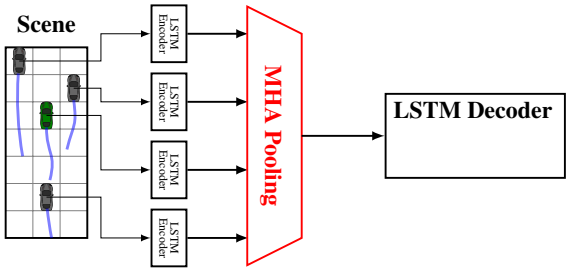


Figure 3.8: MHA Pooling based Architecture

3.2.4.1 Social LSTM

Alahi et al. [1] introduced the social LSTM concept for pedestrian trajectory prediction task. They simultaneously capture the spatial and temporal dependencies in the neighboring agents recent motion at each time step using a regular LSTM and a social pooling technique (cf. Figure 3.9).

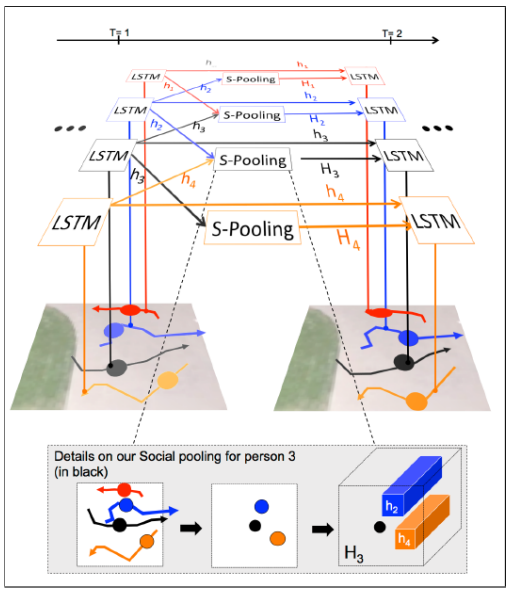


Figure 3.9: Social LSTM [1]

In contrast to the work of Park at al. [76] that uses a single LSTM and feed it with all the agents' states, Alahi et al. [1] attribute an LSTM to each agent to model its motion dynamics. The hidden state of each LSTM presents its past motion dynamics. In order to model the interactions between the neighboring agents, they combine the hidden states of the LSTMs corresponding to a set of neighboring pedestrians. To do so, they compute the position of the neighboring agents with reference to the target agent at each of the past time steps considered and place their hidden state on a grid based on their positions. The combination of hidden states information from the LSTM cells of neighboring agents forms a compact representation of the interactions between agents

named also a context vector. The social pooling consists in feeding each target agent LSTM cell with its corresponding context vector. Thus, at each time step, the target agent LSTM cell is fed with not only the target agent position and its previous hidden state, but also with the context vector. This enables the network to better model the interactions between the neighboring agents.

3.2.4.2 Convolutional Social Pooling

Deo et al. [23] extend the social pooling technique and deploy it for vehicle trajectory prediction task. In contrast to social LSTM [1] where a regular LSTM is deployed, Deo et al. [23] use an LSTM encoder to generate a representation of each vehicle trajectory (motion encoding). They also use a grid representation of the scene; Indeed, they place the motion encoding of each agent based on its position in the grid at the prediction time. Then, they use convolutional layers applying successive local operations on the outputs from the encoders followed by a maxpool layer (cf. Figure 3.10). In this way, they generate a context vector that consists in a compact representation of the vehicles interactions.

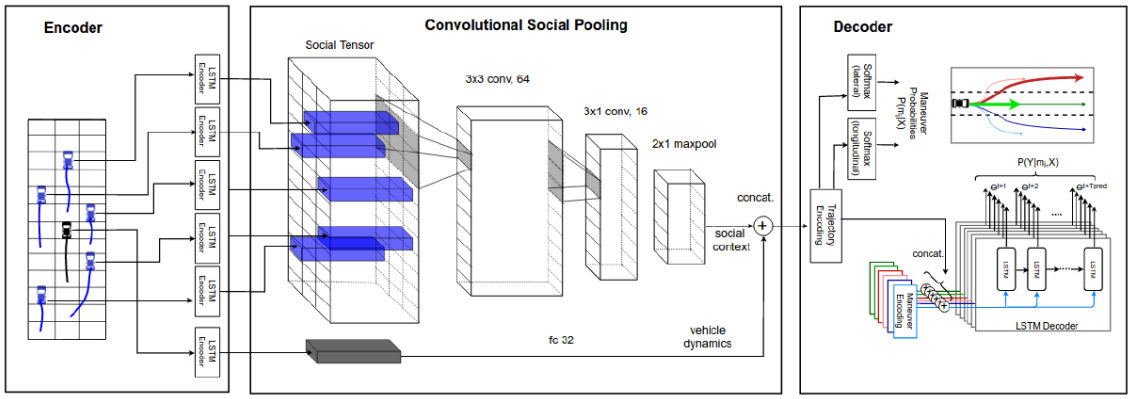


Figure 3.10: Convolutional Social Pooling [23]

Convolutional Social pooling improves the performance of Social LSTM. However, it has some limitations. In fact, successive local operations are not always efficient at capturing long range dependencies because of limited receptive fields. Furthermore, the generated context vector is independent of the target vehicle's state. This infers that two agents in the same scene would have the same representation of the context independently of their recent motion and their intention. This representation does not properly follow human reasoning. Indeed, a human driver focuses more of the elements of the scene that would influence his future motion. Therefore, his representation of his context depends on his recent motion and his goal. In our approach, we take the advantages of the convolutional social pooling and extend it using a different pooling method that better imitates human reasoning.

Further detail about our proposed methods are presented in the following.

3.3 Relational Recurrent Neural Networks based Methods (L-RRNN and Sc-RRNN)

In our first approach, we predict the future trajectory of a target vehicle by combining the advantages of LSTMs in sequence modeling and the power of attention mechanism to capture the spatial inter-vehicles dependencies. To that end, we bring Relational Networks (RNs) based methods to the problem of interaction aware vehicle motion prediction. RNs operates by extracting the dependencies and the relations between features, among them, we list Relational Recurrent Neural Networks (RRNNs) [92]. Specifically, we propose two variants of RRNN-based methods (Scene representation **Sc-RRNN** and per Lane representation **L-RRNN**) based on two different ways of considering the input data and therefore modeling the interactions between the surrounding agents.

3.3.1 Overall Model

RRNNs extend the LSTM architecture by introducing interactive memory blocks using Multi-Head Dot Product Attention inside of the LSTM block. Our first approach to tackle the task of trajectory prediction is based on RRNNs encoder decoder. It is characterized by:

- **Per block information storing:** Instead of using a memory vector (like in regular LSTM), input vectors are selectively stored into separate interacting memory slots based on their contents. Indeed, each memory attends to all of the other memories. Then, it updates its content based on the attended information.

- **Relational reasoning:** The RRNN performs a complex relational reasoning having two aspects: temporal and spatial. The relational reasoning across time consists in comparing and contrasting features at different times steps. In addition, the RRNN relates salient information about vehicles motions and therefore models their spatial interactions. The attention mechanism with per slot information storing enables to capture the dependencies between vehicles features by relating their current states representations to different representations of their past tracks stored in the memory blocks using multiple attention heads.

- **No distance constrained analysis:** The dependencies between input features is not always tied to proximity in time or space. This is modeled in our approach by attention mechanism since it directly relates distant features based on their content rather than proximity.

- **Different focusing:** The RRNN does not only extract information based on one relational aspect between the input features. Different relations are encoded using multiple selective attentions.

RRNNs are memory based recurrent neural networks able to perform relational reasoning between input entities over time. They operate by slicing the memory and the input into slots and heads and provoking interactions between them. Our model consists of a scene embedding (cf. Section 3.3.3) and RRNNs based encoder and decoder (cf. Section 3.3.4). Figure 3.11 illustrates the per lane representation **L-RRNN** described in 3.3.3.2. After the scene embedding, the encoder learns the vehicle motion and captures the dependencies in the input data using the Relational Memory Core (RMC) block. At each iteration, the RMC is fed with the previous memory matrix M^t and the current scene embedding Emb^t. It applies the Multi-Head Dot Product Attention (MHDPA) (cf. 3.3.5) to provoke the interaction between memory and input slots. MHDPA operates by projecting each memory and input slot using row-wise shared weights W_Q^l, W_K^l and W_V^l to generate the queries Q_l^t, keys K_l^t and values V_l^t respectively. The MHDPA module is followed by row-wise multilayer perceptron MLP, then, the resulting memory is gated to form the next memory state and an output vector which are fed to the decoder at t_{obs}. The decoder, composed of RRNNs, outputs the predicted future trajectory of the target vehicle. Further details about our RRNN based method will be presented in the following.

3.3.2 Inputs Representation

The first step to form our model is to determine the input cues and to embed them into an adequate representation. In this case, we input to our model a sequence of the states of the target vehicle and its surrounding vehicles in order to be able to model the interactions between them. To do so, we define a $3D$ spatial grid G^t composed of the coordinates of the target and its surrounding vehicles at time t based on their positions at time t_{obs}.

$$G^t(m,n,:) = \delta_{mn}(x_i^{t_{obs}}, y_i^{t_{obs}})(x_i^t, y_i^t) \quad \forall i \in \mathcal{A}_T \tag{3.7}$$

$\delta_{mn}(x,y)$ is an indicator function equal to 1 if and only if (x,y) is in the cell (m,n), $\mathcal{A}_T$ is the set of neighboring vehicles in addition to the target one.
As we make trajectory prediction in a highway environment, we exploit the lane markings to define the columns of our grid and associating them to the three lanes (cf. Figure 3.11). We consider a grid size of $(13,3)$ centered on the target vehicle position and covering a longitudinal distance of 58.5 meters (Grid cell size $= 4.5m$).
Unlike most of the state of the art works that consider the vehicles immediately around the target vehicle, we adopt a grid over the neighboring area. This representation of the scene has the following advantages:

- It models the spatial distances between the vehicles in the scene and represents the drivable areas.
- It enables us to consider different scenarios with different numbers of traffic participants.
- It preserves the lane structure of the highway.

3.3.3 Inputs Embedding

Trajectory prediction depends on the input data format and their latent representation generated using embedding. We feed our network with two different ways of embedding the input data, and then we compare the results of the different methods.

3.3.3.1 Scene embedding (Sc-RRNN)

Our first method consists in considering the states of all the agents in the scene as an input vector at each past time step. We embed each vector of states using a fully connected layer to generate an embedding vector. These embedding vectors for the time steps $t = 1, \ldots, t_{obs}$ are sequentially fed to the RRNN encoder:

$$Emb^t = \Psi(G^t; W_{emb})$$

where $\Psi()$ is a fully connected function with LeakyReLU non linearity, W_{emb} is the learnt embedding weights.
The RRNN with per slot memory storing using multiple attention heads infers the interactions and the dependencies between the input vehicles.

3.3.3.2 Per lane representation (L-RRNN)

Here, we divide the scene based on lanes to generate an input matrix. We embed the states of the agents at each lane using a fully connected layer Ψ to generate an embedding matrix of size $(3, size\ of\ embedding)$. The matrix embedding the scene for time steps $t = 1, \ldots, t_{obs}$ are sequentially fed to the RRNN encoder:

$$Emb^t(n, :) = \Psi(G^t(:, n, :); W_{emb}),\ n = 1, 2, 3$$

This model conserves the lane-wise structure of the road. It captures the spatio-temporal interactions between vehicles in the same and adjacent lanes. It performs a lane-based attention to focus on the lane-changing behavior. In this model, we consider three memory slots to store lane-level information.

3.3.4 Relational Recurrent Encoder-Decoder

We deploy an encoder decoder architecture in the task of trajectory prediction:

- **RRNN encoder:** receives the input sequence embedding, extracts the properties of the target vehicle past trajectory and interaction information, compresses them in an encoding vector and feeds this vector together with the memory block to the decoder.

- **RRNN decoder:** learns to generate the predicted trajectory based on the received information: At t_{obs} time step, the decoder has as input the encoding vector and the memory block. It makes prediction for the next time steps and generates the next memory block. Then, we proceed by forming and passing the memory blocks and reinjecting the decoder's predictions into the decoder's input of the next time step to sequentially generate the predicted target vehicle positions.

The encoder and decoder are composed of RRNNs. RRNNs are memory based recurrent neural networks able to perform relational reasoning between input entities over time. They are based on iterative information selective storing into blocks and computing interactions between them. In fact, each RRNN block contains a number of memory slots where the pertinent information is stored.
RRNNs operate by slicing the memory and the input into slots and heads and provoking interactions between them. Indeed, each memory slot is updated each time step based on:

- **Memory-Memory attention:** each memory slot attends over the other memory slots. This captures the interactions and dependencies in the stored information. Since the stored information here present the recent motions of a set of neighboring vehicles, the MHA captures their interactions over time.

- **Memory-Input attention:** each memory slot attends over the input embedding slots. Attention enables to decide which information from the input would be stored in adequate memory slots based on its relation to what is already contained in the memory. This infer inter-vehicles interactions as well.

3.3.5 Multi-Head Dot Product Attention (MHDPA):

In each RRNN block, we use linear projections of the previous memory M^t and the input embedding Emb^t at each time step t to generate the queries $Q_l^t = M^t\, W_Q^l$, keys $K_l^t = [M^t; Emb^t]\, W_K^l$ and values $V_l^t = [M^t; Emb^t]\, W_V^l$. While $[M^t; Emb^t]$ denotes the row-wise concatenation of M^t and Emb^t.

In order to enable the memory slots to share different information and represent different spatio-temporal interactions, we use multiple attention heads. Therefore, we generate n_h sets of queries, keys, and values for $l = 1, \ldots, n_h$ using different projection matrices. The memory is updated using multi-head dot product attention over the other memory slots and the current input embedding:

$$\tilde{M_l}^{t+1} = A(Q_l^t, K_l^t, V_l^t) = \underbrace{softmax\left(\frac{Q_l^t \; transpose(K_l^t)}{\sqrt{d^k}}\right)}_{attention \; weights} V_l^t \tag{3.8}$$

$\tilde{M}_l^{t+1}$ is an update of the memory where each slot is a weighted sum of the projections of the previous memory slots and the projections of the current embedding input.
d^k is a a scaling factor that corresponds to the dimensionality of the key vectors.
We apply the attention operation described above for each head. The resulting memory $\tilde{M}^{t+1}$ is column-wise concatenation of the memories $\tilde{M}_l^{t+1}$ for $l = 1, \ldots, n_h$.
We employ a residual connection [36] around the MHDPA followed by an MLP then a second residual connection. These operations are encapsulated into an LSTM cell as described in [92]. Therefore, the resulting memory block is gated and used as next memory state M^{t+1}.

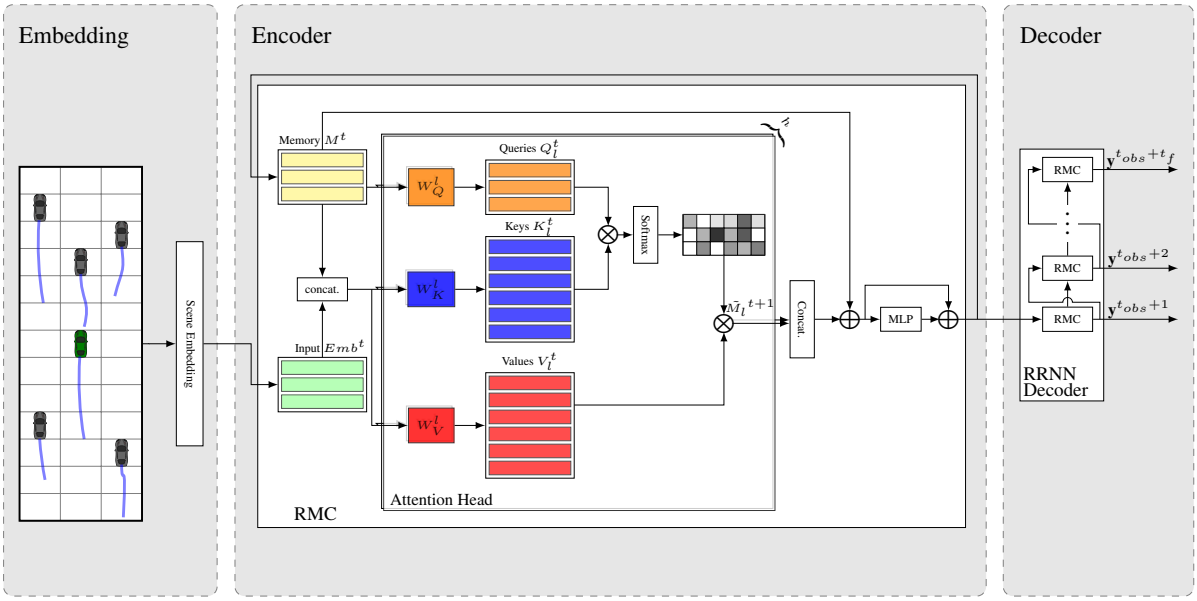


Figure 3.11: Per Lane Scene Representation RNN (L-RRNN) Model

3.3.6 Introducing MHA into an LSTM

In order to model the spatio-temporal evolution of the trajectories, the MHA is integrated inside of an LSTM. To do so, the memory matrix M is randomly initialised in the LSTM,

then, it is updated with $\tilde{M}$ at each time step. M can be considered as a matrix of cell states c for a regular LSTM (cf. Section 3.2.1). The operations of an individual memory m_r (a row from the matrix M) are nearly equivalent to those in a regular LSTM cell state.

$$f_{r,t} = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{r,t-1} + b_f) \tag{3.9}$$

$$i_{r,t} = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{r,t-1} + b_i) \tag{3.10}$$

$$\tilde{c}_{r,t} = \tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{r,t-1} + b_c) \tag{3.11}$$

$$o_{r,t} = \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{r,t-1} + b_o) \tag{3.12}$$

$$m_{r,t} = f_{r,t} \odot m_{r,t-1} + i_{r,t} \odot g_\Psi(\tilde{m}_{r,t}) \tag{3.13}$$

$$h_{r,t} = o_{r,t} \odot \tanh(m_{r,t}) \tag{3.14}$$

Same notations as in regular LSTM equations (cf. Section 3.2.1) are deployed (r is added as a subscript to denote the row from a matrix) and g_Ψ is a memory-wise MLP with layer normalisation.

We notice here that the main difference with a regular LSTM is that the gates are applied for each memory row. In addition, a different kind of gating denoted 'memory' gating is applied; W_{xf}, W_{hf}, W_{xi}, W_{hi}, W_{xc}, W_{hc}, W_{xo}, W_{ho} are converted from weight matrices (in regular LSTM) into weight vectors and element-wise product in the gating equations (in regular LSTM) are replaced with scalar-vector multiplications.

We note that since all the parameters are shared for each m_i, different number of memories can be used without affecting the number of parameters.

3.3.7 Implementation Details

The input grid is embedded into an embedding vector or matrix of sizes 64 and (3,64) depending on the input embedding type. Then, we use the Leaky ReLU activation function with $\alpha = 0.1$.

We deploy RRNNs encoder decoder with two memory slots for Sc-RRNN and three for L-RRNN. Each memory slot is 64 in size. We employ $h = 2$ parallel attention heads over projected vectors of size 32. We use a batch size of 128. We adopt the Adam optimizer [52]. The model is implemented using PyTorch [78].

3.4 Multi-head Attention based Method (MHA-LSTM)

In this section, we describe our second method based on LSTM encoder decoder architecture and the MHA block. In this method, instead of computing the attention at each time step to capture the dependencies between the input features, we adopt the attention mechanism to derive the relative importance of surrounding vehicles with

respect to their overall motion representation. In other words, we apply the MHA on the whole past trajectories encodings only at the prediction time. The attention mechanism selectively aggregates the features that model the interaction between the vehicles by a weighted sum of the features representing all the surrounding vehicles' trajectories and thus directly relates vehicles based on their correlation without constraints on their distance. In addition, using multiple attention heads enables to extract different types of interactions and combine them to capture higher order relationships. This provides a better understanding of the interactions in scene.

In the following, we start by presenting the overall model. Then, we describe each building block of our model: the trajectory encoder, the vehicles interaction module and the trajectory generator. We also propose another MHA-based method of interaction modeling.

3.4.1 Overall Model

In this proposed method, we use an LSTM encoder decoder architecture with an attention based pooling technique. In fact, instead of using an RRNN that performs relational reasoning only at each past time step, we model interactions using the generated motion encoding vectors at the prediction time which helps to reduce the model complexity.
Our model architecture is made up of three main components (cf. Figure 3.12):

- **Encoding layer,** where the temporal evolution of each vehicle trajectory and its motion properties are encoded by an LSTM encoder.

- **Attention Module**, which links the hidden states of the encoder and decoder. It explicitly evaluate the importance of the surrounding vehicles in determining the future motion of the target vehicle using different operations based on their motion encoding. Then, it forms a vector representing the context.

- **Decoding layer**, which receives the context vector containing the selected information about the neighboring vehicles and the target vehicle motion encoding and generates the parameters of the distribution over the target vehicle's predicted future positions.

3.4.2 Trajectory Encoder

This encoding layer encodes the trajectories of the vehicles belonging to a neighborhood of the target vehicle at time $t = t_{obs}$. Each state vector $\mathbf{x}_i^t$ of each vehicle i of the neighboring area is embedded using a fully connected layer to form an embedding vector e_i^t.

$$e_i^t = \Psi(\mathbf{x}_i^t; W_{emb})$$

where $\Psi()$ is a fully connected function with LeakyReLU non linearity, W_{emb} is the learnt embedding weights.
The LSTM encoder is fed by the embedding vectors of each vehicle i for time steps $t = 1, \ldots, t_{obs}$:

$$h_i^t = LSTM(h_i^{t-1}, e_i^t; W_{encoder})$$

h_i^t is the hidden state vector of the i^{th} neighboring vehicle at time t. We note h_T^t the hidden state vector of the target vehicle at time t. $W_{encoder}$ are the LSTM encoders weights. Each LSTM encoder share the same weights $W_{encoder}$.
We built a $3D$ spatial grid H composed of the neighboring vehicles' hidden states at time t_{obs} based on their positions at time t_{obs}.

$$H(n, m, :) = \delta_{nm}(x_i^{t_{obs}}, y_i^{t_{obs}}) h_i^{t_{obs}} \quad \forall i \in \mathcal{A}_T$$

$\delta_{nm}(x, y)$ is an indicator function that equals 1 if (x, y) is in the cell (n, m) and 0 otherwise. $\mathcal{A}_T$ consists of the set of surrounding vehicles present in the considered area. The columns correspond to the three lanes $(M = 3)$. The considered spacial area corresponding to the grid is centered on the target vehicle position and sized of (N, M). It covers a longitudinal distance of $90\ m$ with a grid cell size of $4.5\ m$. We note C the dimension of the trajectory encoding vectors $h_i^{t_{obs}}$ and we reshape the grid H to (NM, C).

3.4.3 Vehicle Interaction Modules

As the behavior of vehicles on a highway could be highly correlated, it is important to consider the interactions between the vehicles when predicting their future motion. Attention is used to capture long-range spatio-temporal dependencies. The attention module explicitly models the interactions between the target vehicle and the other vehicles in the grid H and selects the surrounding vehicles to pay attention to when computing the future trajectory of the target vehicle.
Instead of computing vehicle relationships at each time step, which is computationally expensive, we use the hidden states of the encoder LSTM computed at the observation time as inputs to the attention module. These hidden states are projected into a high-dimensional space, if we consider all the attention heads. The vehicles interactions can be exploited as follows:

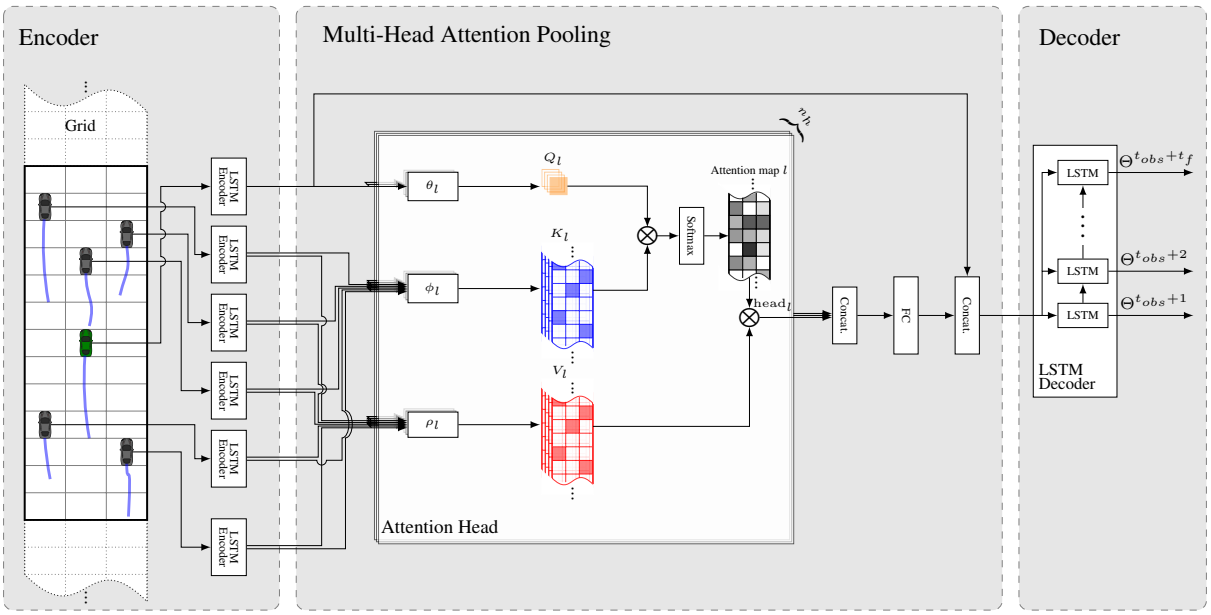


Figure 3.12: Description of the MHA Pooling based Model: The LSTM encoders, with shared weights, generate a vector encoding of each vehicle motion. The multi-head attention module models the interactions between the target (green car) and the neighboring vehicles based on their importance. The decoder receives the interaction vector and the target vehicle encoding and generates a distribution for the predicted trajectory.

- The hidden state of the target vehicle is mapped to a query $Q_l = \theta_l(h_T^{t_{obs}}, W_{\theta_l})$
- The grid is mapped to form the keys $K_l = \phi_l(H, W_{\phi_l})$ and the values $V_l = \rho_l(H, W_{\rho_l})$.

W_{θ_l}, W_{ϕ_l} and W_{ρ_l} are the weight matrices that will be learned in each attention head l. An attention feature $head_l$ is then calculated as a weighted sum of values v_{lj}, where the attention weights, α_{lj}, weight the effect of surrounding vehicles on the target vehicle future motions, based on their relative dynamics.

We investigate three possible ways to compute the attention weights α_l:

$$head_l = \sum_{j=1}^{NM} \alpha_{l_j} v_{l_j}$$

3.4.3.1 α-Attention

Attention weights are computed from the encoding vectors of the surrounding vehicles independently of the target vehicle state. They are computed using a $tanh$ function and a fully-connected layer.

$$\alpha_l = softmax(w_l^T \tanh(K_l))$$

w_l is a learned weight, and $\alpha_l \in R^{1 \times NM}$ is the l^{th} attention.

3.4.3.2 Dot-Product Attention

The weights represent the effect of an interaction between a pair of vehicles based on their relative dynamics. They are the product of the query Q with keys K.

$$\alpha_l = softmax\left(\frac{Q_l K_l^T}{\sqrt{d}}\right)$$

$Q_l K_l^T$ is matrix multiplication used to calculate dot product similarities. d is a scaling factor that equals to the dimensionality of the projection space.

3.4.3.3 Concatenation Attention

The pairwise relation can be also represented by a concatenation operation, as in [93, 112].

$$\alpha_l = softmax(w_l^T concat(repeat(Q_l), K_l))$$

One can notice that dot-product and concatenation attentions consider pairwise inter-relationships, whereas α-attention does not.

3.4.4 High Order Interaction

We deploy a higher order interaction extractor based on multi-head attention to retain different types of spatio-temporal relationships. The use of multi-head is inspired by the Transformer [107] architecture. In fact, a single learned attention feature mainly focuses on one inter-related subgroup of vehicles that may represent a single aspect of the possible spatio-temporal relationships occurring in the neighborhood of the target vehicle. In order to extend the attention to higher order interactions, different queries, keys and values are generated n_h times in parallel, in n_h attention heads, with different learned linear projections Q_l, K_l and V_l, $l \in \{1, \ldots, n_h\}$.
The n_h generated attention features represent n_h subgroups of vehicles inter-related with the target vehicle. These representations are concatenated and dynamically weighted to extract complex interactions between the different subgroups.

$$z = Concat(head_1, ..., head_{n_h})\, W^O$$

z is the compact context vector that combines interaction information of all the vehicles.

3.4.5 Trajectory Generation

LSTM Decoder is fed by the context vector z, which contains the selected information about the vehicles interactions, and the motion encoding of the target vehicle: $h_{dec} = Concat(h_T^{t_{obs}}, z)$. It generates the predicted parameters of the distributions over the target vehicle's estimated future positions for time steps $t = t_{obs} + 1,\ldots, t_{obs} + t_f$.

$$\Theta^t = \Lambda(LSTM(h_{dec}^{t-1}; W_{dec}))$$

where Θ^t is the predicted parameters of the positions distribution at time t, $\Lambda()$ is a fully connected function followed by a LeakyReLU non linearity, W_{dec} are the learnt weights of the LSTM decoder and h_{dec}^{t-1} is the hidden state vector of the decoder at time $t-1$.
Our model is trained by minimizing the following negative log-likelihood loss function:

$$L_{nll}(\mathbf{Y}_{pred}) = -\sum_{t_{obs}+1\leq t\leq t_{obs}+t_f} \Big\{log(P_{\Theta^t}(\mathbf{y}^t|\mathbf{X}))\Big\}$$

3.4.6 Another Variant: Local and Non Local Social Pooling

We propose to extend the non-local multi-head attention mechanism previously described and to combine it with convolution blocks to model both local and distant influences of vehicles, based on their relation with the autonomous vehicle.
This pooling mechanism builds a context vector which encodes the cues of surrounding vehicles that most influence the future trajectory by combining the advantages of two individual architectures:

- Convolution Layer captures the interactions of nearby vehicles and produces the local context over which we use attention operations.
- Multiple head attentions learn different local and distant relationships and combine them according to their importance.

Interaction between traffic participants can be complex. It is composed not only of local but also non-local influences. In the proposed pooling module, convolutions and attention complement each other to model local and distant dependencies. We define a spatial grid H_α composed of the transformation of the encoders last hidden state of each of the surrounding vehicles by a function α (defined later) based on their positions at time t_{obs}.

$$H_\alpha(m, n, :) = \sum_{\forall i\in\mathcal{A}_T} \delta_{mn}(x_i^{t_{obs}}, y_i^{t_{obs}})\alpha(h_i^{t_{obs}}) \tag{1}$$

$\delta_{mn}(x, y)$ is an indicator function equal to 1 if and only if (x, y) is in the cell (m, n), $\mathcal{A}_T$ is the set of neighboring vehicles.

This representation preserves the spatial relationships between vehicles and the lane structure. The columns correspond to the three lanes. Depending on the used dataset, we consider two different grid sizes $(13, 3)$ and $(41, 3)$ covering a longitudinal distance of respectively 58.5 and 184.5 meters. The greater size of the second grid is justified by the relatively high inter-vehicles distances caused by high velocities in the HighD dataset.

3.4.6.1 Convolution Layer

Convolutions enable the model to learn local dependencies. We use a (3×3) depthwise convolution kernel [19] to capture the dependencies between nearby vehicles. We use horizontal zero padding to maintain the three-lane road structure.

3.4.6.2 Non-local Multi-head Attention Mechanism

The attention mechanism enables us to get a compact representation which combines information from all the surrounding vehicles based on their relationship with the target vehicle. We build n_h interaction vectors a_j, $j \in \{1, \ldots, n_h\}$ using the generic non-local operation defined in [112]:

$$a_j = \frac{1}{\mathcal{C}_j(h_T^{t_{obs}})} \sum_{\forall (m,n) \in grid} f_j(h_T^{t_{obs}}, h_i^{t_{obs}}) \otimes Conv(H_{g_j}, W_L^j)$$

$\otimes$ represents an elementwise multiplication operation applied to the two grids $f_j(h_T^{t_{obs}}, h_i^{t_{obs}})$ and $Conv(H_{g_j}, W_L^j)$.

As defined in equation (1), H_{g_j} is the grid composed of the projections of the encoders last hidden states $h_i^{t_{obs}}$ by the linear transformation $g_j(h_i^{t_{obs}}) = W_g^j h_i^{t_{obs}}$. W_L^j is the convolution kernel and $\mathcal{C}_j(h_T^{t_{obs}})$ is a normalization factor.

The vector describing the target vehicle motion $h_T^{t_{obs}}$ is transformed into a new feature space by a linear function $\theta_j(h_T^{t_{obs}}) = W_\theta^j h_T^{t_{obs}}$. We build a second grid H_{Φ_j} composed of the projections of the encoders last hidden states $h_i^{t_{obs}}$ by the linear transformation $\Phi_j(h_i^{t_{obs}}) = W_\Phi^j h_i^{t_{obs}}$.

The Gaussian function computes the influence of the position (m, n) in the grid H_{Φ_j} on the target vehicle's motion in the projection space.

$$f_j(h_T^{t_{obs}}, h_i^{t_{obs}}) = e^{transpose(\theta_j(h_T^{t_{obs}}))\ Conv(H_{\Phi_j}, W_L^j)}$$

The normalization factor is

$$\mathcal{C}_j(h_T^{t_{obs}}) = \sum_{\forall (m,n) \in grid} f_j(h_T^{t_{obs}}, h_i^{t_{obs}})$$

The attention weight $\frac{1}{\mathcal{C}_j(h_T^{t_{obs}})} f_j(h_T^{t_{obs}}, h_i^{t_{obs}})$ computes how much attention is put on the (m, n) position in the grid when predicting the target vehicle's motion.
The $head_j$ is a weighted sum of the features at all positions of the grid in the projection space. We consider $head_j$ also as the j^{th} attention head. We use a multi-head attention approach to enable the model to differently focus on different positions in different projection spaces.
The attention heads are concatenated and projected to form the output z of the attention block:

$$z = W_z Concat(head_1, .., head_{n_h})$$

where n_h is the number of the attention heads.
We employ a residual connection [36] around the attention block, followed by a normalization layer.

3.4.7 Implementation Details

We deploy LSTM encoder with 64 units (C=64) and decoder with 128 units. The dimension of the embedding space is 32. We use different numbers of parallel attention operations applied on the projected vectors of size d=32. The batch size is 128 and the adopted optimizer is Adam [52]. The model is implemented using PyTorch [78].

3.5 Experimental Evaluations

Evaluations have been performed on public driving datasets (NGSIM [44, 45] and highD [55]) that are described in Section 2.5. Our proposed approaches Sc-RRNN, L-RRNN and MHA-LSTM are compared with state-of-the-art quantitatively. Qualitative results are also presented for further analysis.

3.5.1 Evaluation Metric

In our evaluation, we use Root of the Mean Squared Error (RMSE) metric since it averages the distance between predicted trajectories and the ground truth.

$$L_{RMSE} = \sqrt{\frac{1}{t_f} \sum_{t=t_{obs}+1}^{t_{obs}+t_f} (x_T^t - x_{pred}^t)^2 + (y_T^t - y_{pred}^t)^2}$$

We use the means of the predicted distributions over the future trajectories to calculate the RMSE.

3.5.2 Models Compared

Evaluations have been performed by comparing our proposed methods with some recent state of the art methods. To do so, we consider two types of approaches:

- Models without the social pooling technique: Vanilla LSTM (*V-LSTM*), Scene LSTM Encoder Decoder (*Sc-LSTM*), our RRNN with scene representation (*Sc-RRNN*) and our RRNN with per lane representation (*L-RRNN*).
- Models considering the interactions between surrounding vehicles with a variant of social pooling: Social LSTM (*S-LSTM*) [1], Convolutional Social Pooling (*CS-LSTM*) [23], Multi-Agent Tensor Fusion (*MATF GAN*) [120], our Non-local Social Pooling (*NLS-LSTM*) and our Multi-head Attention Social Pooling (*MHA-LSTM*).

All models are fed with the track history of the target and the surrounding vehicles and output distributions over the future trajectory of the target vehicle. In the following, we compare each type of methods separately.

- **Vanilla LSTM** (*V-LSTM*): an encoder decoder LSTM based model. It uses the track history of the target vehicle in the encoder LSTM and generates the output trajectory with the LSTM decoder. This represents an independent trajectory prediction model.
- **Scene LSTM Encoder Decoder** (*Sc-LSTM*): an encoder decoder based model where the encoder encodes the trajectories of the target and surrounding vehicles. The encoding vector is fed to the decoder which generates trajectory predictions.
- **Social LSTM** (*S-LSTM*) [1]: An LSTM encoder-decoder using a fully connected social pooling.
- **Convolutional Social Pooling** (*CS-LSTM*) [23]: An LSTM encoder-decoder using a convolutional social pooling.
  (*CS-LSTM(M)*) generates multi-modal trajectory predictions based on six maneuvers (2 longitudinal and 3 lateral).
- **Multi-Agent Tensor Fusion** (*MATF GAN*) [120]: the model encodes the scene context and vehicles' past trajectories, then, deploys convolutional layers to capture interactions. Finally, the decoder generates the predicted trajectories, using adversarial loss.

Our proposed methods:

- **Relational Recurrent Neural Network with scene representation** (*Sc-RRNN*) model described in section 3.3.3.1.
- **Relational Recurrent Neural Network with per lane representation** (*L-RRNN*) model described in section 3.3.3.2.
- **Non-local Social Pooling** (*NLS-LSTM*) [68]: combines local and non local operations to generate an adapted context vector for social pooling. Five attention heads are used in this approach. (cf. Section 3.4.6)
- **Multi-head Attention Social Pooling** (*MHA-LSTM*): This is the model described previously using multi-head dot product attention with $\mathbf{x}_i^t = (x_i^t, y_i^t)$, i.e. without using velocity, acceleration and type information for each vehicle and with four attention heads.
- **Multi-head Attention Social Pooling** (*MHA-LSTM(+f)*): *MHA-LSTM* with additional input features (velocity, acceleration and type) and with three attention heads.

3.5.3 Quantitative evaluation

3.5.3.1 Overall evaluation of RRNN

Table 3.1: Root mean squared prediction error (RMSE) in meters over a 5 second prediction horizon for the models using highD dataset

Error	Prediction Horizon (s)	V-LSTM	Sc-LSTM	Sc-RRNN	L-RRNN
Total	1	0.31	0.32	0.29	**0.22**
	2	0.81	0.82	0.69	**0.65**
	3	1.51	1.60	1.33	**1.31**
	4	2.48	2.63	2.22	**2.22**
	5	3.71	3.87	**3.33**	3.38
Lateral	1	0.10	0.10	0.08	**0.05**
	2	0.32	0.20	0.18	**0.14**
	3	0.46	0.33	0.30	**0.26**
	4	0.57	0.45	0.43	**0.37**
	5	0.65	0.56	0.53	**0.48**
Longitudinal	1	0.27	0.31	0.27	**0.22**
	2	0.74	0.79	0.66	**0.63**
	3	1.44	1.57	1.30	**1.29**
	4	2.42	2.59	**2.16**	2.19
	5	3.65	3.83	**3.27**	3.34

Let us first compare LSTM based methods that don't deploy a social pooling technique even if they may model agents interactions. This includes Sc-LSTM, V-LSTM and two variants of our RRNN based proposed method: Sc-RRNN and L-RRNN. Table 3.1 shows the RMSE values for four of the models being compared over a prediction horizon of five seconds on highD dataset. First, we observe that Sc-LSTM and V-LSTM have comparable total RMSE errors. While Sc-LSTM produces better lateral error, it has larger longitudinal error than V-LSTM. This may be due to the fact that the LSTM has limited capability to capture the effects of surrounding vehicles on predicting the future motion of the target vehicle. This also proves the effectiveness of considering neighboring vehicles in the prediction of the lateral motion of the target vehicle.
Both proposed methods, Sc-RRNN and L-RRNN, lead to further improvement in prediction error, suggesting the importance of the use of the multiple memory slots and the attention across these memories in the task of motion prediction. We also note that the improvement produced by the use of RRNNs seems to be more remarkable for longer prediction horizons. This shows that LSTM without the MHA has limited capacity to perform long term relational reasoning.
Additionally, the per-lane embedding of the scene has produced lower lateral error. This can infer that explicit lane-wise division of the scene and the memory-input slots interactions via MHDPA gives additional information about inter-lane dependencies.
This first step of evaluation highlights the importance of considering surrounding agents in the trajectory prediction task. Furthermore, it infers that the input representation influences the prediction performance. In addition to our proposed RRNNs based variants, we presented a second approach that aims to improve vehicle motion prediction using a social pooling technique.

3.5.3.2 Overall evaluation of MHA-LSTM

Let us now evaluate the set of methods that use a variant of social pooling. To compare our MHA pooling based models, we consider the results reported in recent studies [23, 24] on the NGSIM dataset and we train S-LSTM and CS-LSTM on highD dataset as well. We train and test the approaches on the NGSIM and highD datasets separately and we notice that the RMSE values obtained on the NGSIM dataset are higher than the ones computed on the highD dataset. This may be due to the difference in size of the two datasets: highD contains about 12 times more vehicles than NGSIM. It can be also caused by annotation inaccuracies resulting in physically unrealistic vehicle behaviors in the NGSIM dataset, as observed by Coifman et al. [20].
Anyway, examining the RMSE values for either NGSIM or highD datasets leads to the same order for the proposed methods. First, we compare the results reported in table 3.1

Table 3.2: RMSE in meters over a 5-second prediction horizon for the models

Dataset	Prediction Horizon (s)	S-LSTM	CS-LSTM	CS-LSTM(M)	MATF GAN	NLS-LSTM	MHA-LSTM	MHA-LSTM(+f)
highD	1	0.22	0.22	0.23	-	0.20	0.19	**0.06**
	2	0.62	0.61	0.65	-	0.57	0.55	**0.09**
	3	1.27	1.24	1.29	-	1.14	1.10	**0.24**
	4	2.15	2.10	2.18	-	1.90	1.84	**0.59**
	5	3.41	3.27	3.37	-	2.91	2.78	**1.18**
NGSIM	1	0.65	0.61	0.62	0.66	0.56	0.56	**0.41**
	2	1.31	1.27	1.29	1.34	1.22	1.22	**1.01**
	3	2.16	2.09	2.13	2.08	2.02	2.01	**1.74**
	4	3.25	3.10	3.20	2.97	3.03	3.00	**2.67**
	5	4.55	4.37	4.52	4.13	4.30	4.25	**3.83**

and table 3.2 on highD dataset. We notice that, in general, social pooling based methods perform better in the task of trajectory prediction. We also observe that our attention-based approaches (*NLS-LSTM*, *MHA-LSTM* and *MHA-LSTM(+f)*) perform better than the others. *MHA-LSTM* reduces the prediction error by about 10% compared to the CS-LSTM while having comparable execution time. With *MHA-LSTM(+f)*, we investigate the use of additional features like the speed and acceleration. We notice that this leads to significant improvements in the motion prediction accuracy, as *MHA-LSTM(+f)* outperforms all the methods. This consolidates our assumption that the relation between vehicles is not only related to their positions but also to their dynamics. The type of the transportation (truck or car) also characterizes the speed and pattern of the motion. Table 3.2 shows the RMSE values for the models using different social pooling techniques on the NGSIM and highD datasets. Therefore, these results indicate that multi-head attention better models the interdependencies of vehicle motion than convolutional social pooling. Moreover, this suggests that considering the relative importance of surrounding vehicles using both positions and dynamics when encoding the context is better than focusing on local dependencies.

In order to further investigate the performance of our elected method *MHA-LSTM*, we conduct further analysis in the following.

3.5.3.3 Effects of using multiple attention heads

In order to evaluate the influence of the number of attention heads on the prediction accuracy, let us examine the RMSE values obtained by *MHA-LSTM* on the highD dataset with 2, 3 4, 5 and 6 attention heads on Table 3.3. We notice that using several attention heads improves the prediction accuracy since each attention head represents a set of weights capturing one aspect of the effect of surrounding vehicles on the target vehicle. In addition, combining the attention vectors helps extract higher order relations. The best performance is reached with four attention heads.

We have conducted further experiments to evaluate the benefits of adding extra features,

Table 3.3: RMSE in meters over a 5-second prediction horizon for different numbers of attention heads on the highD dataset

Time(s) \ Heads	2	3	4	5	6
1	0.21	0.20	**0.19**	0.20	0.21
2	0.61	0.61	**0.55**	0.57	0.59
3	1.20	1.19	**1.10**	1.13	1.16
4	1.96	1.99	**1.84**	1.87	1.92
5	2.95	3.01	**2.78**	2.83	2.93

including explicit vehicle dynamics and type (*MHA-LSTM(+f)*). We observe that we outperform previous results when using different numbers of attention heads.
Considering the trade-off between the complexity of calculation and the *MHA-LSTM(+f)* RMSE corresponding to different numbers of attention heads, we choose to deploy three attention heads in the experiments that follow.

3.5.3.4 Comparison of attention methods

In Table 3.4, we show the performances of the three possible ways to compute the attention weights in *MHA-LSTM(+f)*, named α-attention, dot product attention, and concatenation attention presented in Section 3.4.3. One can note that dot product and concatenation attentions outperform the α-attention. Therefore, we conclude that both the dynamics of the surrounding vehicles and their relationships with the target vehicle are of great importance for trajectory prediction.

Table 3.4: RMSE in meters over a 5-second prediction horizon for different attention operations on the highD dataset

Time(s) \ Methods	$\alpha-$attention	Dot product	Concatenation
1	0.06	**0.06**	0.07
2	0.10	**0.09**	0.11
3	0.26	**0.24**	0.25
4	0.62	**0.59**	0.61
5	1.25	**1.18**	1.20

3.5.3.5 Error evaluation per lane change

In order to complete the evaluation of our approach, we use the trained model *MHA-LSTM(+f)* to estimate the lateral and longitudinal errors obtained while carrying out right (RLC), left (LLC) lane change maneuvers or lane Following (LF) in the test set of the

highD dataset (cf. Table 3.5).

One can notice that the observed lateral error is low even during lane changes maneuvers

Table 3.5: Longitudinal and lateral errors in meters over a 5-second prediction horizon for different maneuvers on HighD dataset

Maneuver	RLC		LLC		LF	
Error	Long	Lat	Long	Lat	Long	Lat
1	0.07	0.03	0.20	0.03	0.05	0.01
2	0.12	0.06	0.32	0.07	0.07	0.02
3	0.34	0.18	0.42	0.19	0.22	0.06
4	0.79	0.43	0.88	0.45	0.54	0.14
5	1.43	0.76	1.74	0.78	1.10	0.22

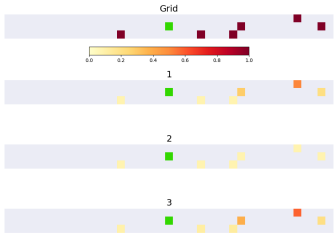
(5% of the test data). This demonstrates the effectiveness of our method in predicting lane changes. It may also be observed that the longitudinal and lateral errors are greater during the LLC maneuvers. This may be due to the fact that the vehicle often speeds up when it performs LLC, which is not the case for the other maneuvers (LF or RLC).

3.5.4 Qualitative Analysis of Predictions

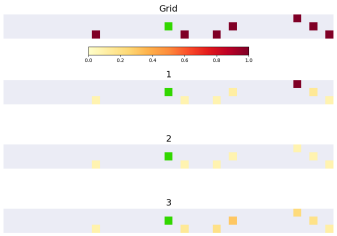
To understand which vehicles are taken into account by each attention head in our method, in Figure 3.13 we present the attention maps corresponding to two lane change maneuvers carried out by the target vehicle. More precisely, a left lane change and a right lane change are shown and the attention maps are computed at times $t_{obs} = t_{lc} - 2s$, $t_{obs} = t_{lc} - 1s$ and $t_{obs} = t_{lc}$ where t_{lc} is the time of crossing the lane mark during the lane change maneuver. Each attention map corresponds to an attention head. The target vehicle is shown in green in the center of the attention map, the grey rectangular region corresponds to the 2D drivable area described by the grid H and the colors of the other vehicles indicate the attention weight associated to them in the attention head (they are darker when their attention weight increases).

We can notice that each attention head focuses on a subset of vehicles in the grid that are crucial to determine the future trajectory of the target vehicle. Moreover, like a human driver, most of the attention is directed to vehicles positioned in front of the target vehicle, the vehicles behind it being less considered.

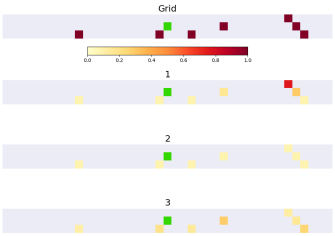
We also notice that, in each example, one attention head considers all the vehicles in the grid equally (attention head 2 for the left lane change and attention head 1 for the right lane change). Moreover, at time $t_{lc} - 2s$, attention map 3 is such that the most important vehicles belong to the target lane even though some other vehicles are closer to the target vehicle in another lane. This consolidates our assumption that the closest neighbors do



(a) Left lane change at $t_{obs} = t_{lc} - 2s$

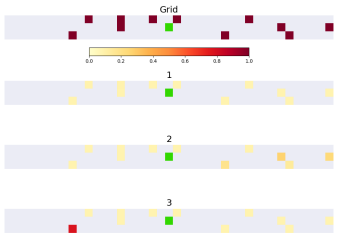


(b) Left lane change at $t_{obs} = t_{lc} - 1s$

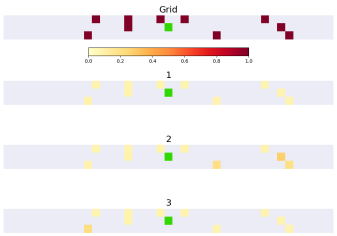


(c) Left lane change at $t_{obs} = t_{lc}$

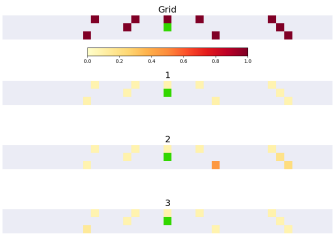
(d) Left lane change at $t_{obs} = t_{lc} - 2s$: blue, red and yellow tracks represent respectively past, future and predicted trajectories



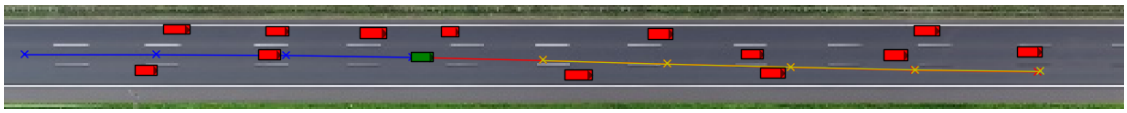
(e) Right lane change at $t_{obs} = t_{lc} - 2s$



(f) Right lane change at $t_{obs} = t_{lc} - 1s$



(g) Right lane change at $t_{obs} = t_{lc}$



(h) Right lane change at $t_{obs} = t_{lc} - 2s$: blue, red and yellow tracks represent respectively past, future and predicted trajectories

Figure 3.13: Three heads attention maps for two different lane change maneuvers. For visualisation, the target vehicle is added in green in the center of all the maps. The driving direction is from left to right.

not always have the strongest influence on the motion of the target vehicle.
Some other factors such as the speed and the vehicle's lane are also essential for correctly estimating the importance of a neighbor. To emphasise that aspect, we consider the relative speeds of the vehicles surrounding the target vehicle and belonging either to the same lane as the target vehicle or to the target lane in examples 1 and 2. Table 3.6 summarizes the states of the considered interacting vehicles.

Table 3.6: Neighbor vehicles states.

Example 1	Vehicle	Preceding	Lead		
	State	Sl	S +		
Example 2	Vehicle	Preceding	Following	Lead	Rear
	State	S +	F	Sl +	F -

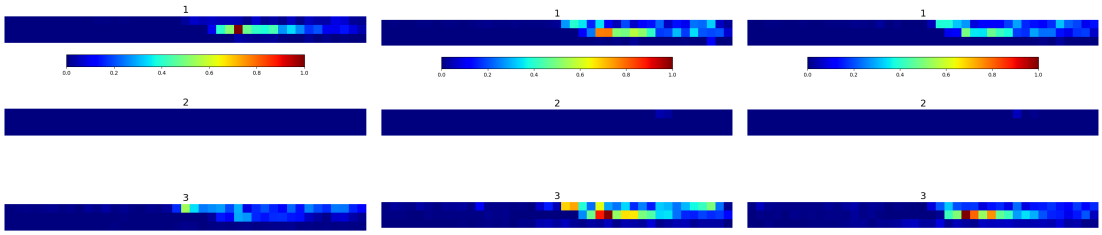
- Preceding, following: a vehicle belonging to the same lane as the target vehicle and preceding or following it.

- Lead, rear: a vehicle belonging to the target lane and positioned ahead or behind the target vehicle.
- S, Sl, F: same speed, slower, faster than the target vehicle respectively.
- -, +: decelerating, accelerating respectively.

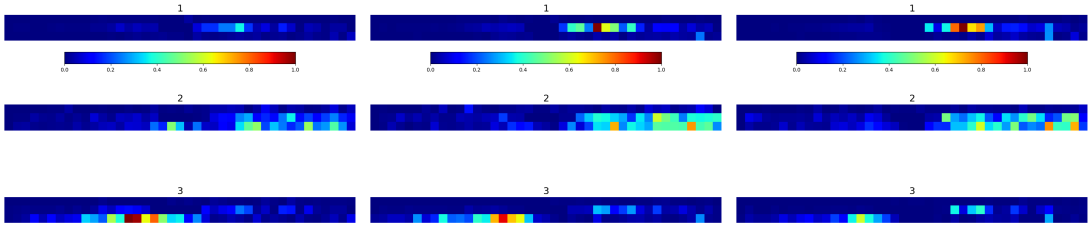
In example 1, the preceding vehicle is slower than the target vehicle. The latter has two possible maneuvers: either to continue in the same lane and decelerate, or to accelerate and make a left lane change. In the left lane, the lead vehicle is far away from the target vehicle and has a similar velocity. This makes the lane change maneuver more likely.
In example 2, the preceding vehicle is accelerating and the following one is faster than the target vehicle. Therefore, the target vehicle has two options, either to accelerate or to make a right lane change.
In these two examples, we notice that even 2 seconds before performing a lane change, the target vehicle focuses mainly on the vehicles that belong to the target lane and which may have an influence on its future speed. Indeed, in both cases, the target vehicle performs the lane change while accelerating or decelerating according to the situation.
In order to go beyond specific examples and visualize the general aspect of attention maps, we also present the mean over each of the three attention heads of 100 vehicles performing lane changes. We notice that there is always a focus on the preceding vehicles in the current lane. Moreover, even $3s$ before the lane change the network detects that the vehicles on the target lane are more important (have higher attention weights) than vehicles on the other lane.

In order to further analyse the degree of the attention accorded to the surrounding agents, we randomly sample vehicles traveling in the center lane; 100 performing a left lane change and 100 performing right lane change. The samples correspond to different scenes with different surrounding agents configurations. we present in figure 3.14 the means of the three attention heads maps of our method *MHA-LSTM(+f)* over the 100 samples of target vehicles traveling in the center lane performing a left lane change (cf. Figures 3.14a, 3.14b and 3.14c) and a right lane change (cf. Figures 3.14d, 3.14e and 3.14f). The attention maps are centred on the target vehicle position.

Considering all figures, we notice that, each attention map concentrates attention on vehicles present in various regions of the grid depending on the performed maneuver; They cooperate to capture different dependencies. In general, there is a focus on vehicles in the same lane of the target vehicle and the target lane. In addition, we observe that, the relatively high attention weights are not limited to nearby vehicles. Even the distant vehicles are important in determining the target vehicle future trajectory. We also note



(a) Left lane change at $t_{obs} = t_{lc} - 3s$ **(b)** Left lane change at $t_{obs} = t_{lc} - 2s$ **(c)** Left lane change at $t_{obs} = t_{lc} - 1s$



(d) Right lane change at $t_{obs} = t_{lc} - 3s$ **(e)** Right lane change at $t_{obs} = t_{lc} - 2s$ **(f)** Right lane change at $t_{obs} = t_{lc} - 1s$

Figure 3.14: Three heads of average attention maps over 100 vehicles with two different lane change maneuvers (the attention maps are centered on the target vehicle position)

that the attention head 2 in Figures 3.14a, 3.14b and 3.14c is blue which means that the surrounding vehicles have, on average, low and almost equal impact on the target vehicle future trajectory. This is coherent with the observed attention head 2 in Figures 3.13a, 3.13b and 3.13c that presents a sample of a target vehicle performing a left lane change.

Figures 3.14a, 3.14b and 3.14c show that, on average, the vehicles intending to perform a left lane change focus more attention to the motion of vehicles ahead of the target one in the current and target lanes even 3 seconds before performing the maneuver. Figures 3.14d, 3.14e and 3.14f illustrate the focus on vehicles present in three regions in the case of a right lane change; ahead of the target vehicle in the current lane, ahead of the target vehicle in target lane and behind of the target vehicle in target lane. This consolidates our analysis of Figure 3.13 that presents a study considering two examples of a target vehicle performing a lane change.

3.6 Conclusion

The two approaches presented in this chapter tackle the task of long-term (5s) trajectory prediction on highway using RRNNs and MHA based pooling technique. They combine the advantages of multi-head dot product attention mechanism and LSTMs to capture the spatio-temporal dependencies between the input tracks.

The first model variants provided competitive results with the state-of-the-art on the naturalistic driving large scale highD dataset based on the RMSE metric for both

longitudinal and lateral position prediction. Therefore, The RRNN based architecture represents a promising way for motion prediction of surrounding vehicles for autonomous vehicle.

We also proposed an adapted MHA pooling based method for modeling vehicle interactions during the tasks of vehicle trajectory prediction on highways. Experiments showed that our approach *MHA-LSTM(+f)* significantly outperforms the state-of-the-art on two naturalistic large-scale driving datasets based on the RMSE metric. Furthermore, the presented visualisation of the attention maps enabled us to recognize the importance and the dependencies between vehicles. It confirmed that the attention is directed based on the future maneuver.

Our evaluation results confirmed our intuitions: the importance of the relative dynamics and the efficiency of multi-head attention mechanism in modeling interactions between vehicles to predict vehicle trajectories in a highway scenario.

Our proposed approach can be extended to consider heterogeneous and mixed traffic scenarios with different road users, such as buses, trucks, cars, scooters, bicycles, or pedestrians. However, further information about the road structure should be integrated in our model for better representation of different driving scenes. This is described in the next chapter.

4

Scene Aware Trajectory Prediction in an Urban Environment: A Recurrent Approach

Contents

4.1 Multi-head Attention with Joint Agent Map Representation (MHA-JAM) 75
4.1.1 Input Representation 76
4.1.2 Multimodal Output 76
4.1.3 Encoding Layer 78
4.1.4 Separate Agent-Map plus Joint Conditioning MHA 78
4.1.5 Joint Agent-Map Attention 80
4.1.6 Decoding Layer 82
4.1.7 Intention-based Prediction 82
4.1.8 Dynamically-feasible Prediction 85
4.1.9 Loss Functions 87
4.2 Double Attention Based Model 89
4.2.1 Input and Output Representation 89
4.2.2 Encoding and Decoding layers 89
4.2.3 Multi-head Double Attention 89
4.2.4 Double Attention for Scene Interaction (DA-JAM) 90
4.3 Simultaneous multi-vehicle trajectory prediction 92
4.3.1 Problem Definition 92
4.3.2 Multi-Head Attention for simultaneous multi-vehicle trajectory prediction 93
4.3.3 Multi-Head Double Attention for simultaneous multi-vehicle trajectory prediction 93
4.4 Experimental Analysis and Evaluations 94

4.4.1 Training and Implementation Details 95
4.4.2 Evaluation Metrics 95
4.4.3 Models Compared 96
4.4.4 Quantitative Evaluation 99
4.4.5 Ablation Experiments 100
4.4.6 Qualitative Evaluation 103
4.5 Conclusion 108

In this chapter, we tackle the task of trajectory prediction in an urban environment where an autonomous vehicle is more likely to encounter complex traffic scenes. In this context, information about the target vehicle past motion and its surrounding agents are not sufficient to predict the future motion of a target vehicle. Indeed, details about the static scene structure are required in order to be able to infer the possible motions. Due to the high variability in the scene structure and agent configurations, prior work has employed the basic form of attention mechanism, applied separately to the scene and agent configuration to learn the most salient parts of both cues. However, the two cues are tightly linked. The agent configuration can inform what part of the scene is most relevant to prediction. The static scene in turn can help to determine the relative influence of agents on each other's motion. For example, the presence of a nearby pedestrian could make a crosswalk in the path of the vehicle of interest a salient part of the static scene, as opposed to if there were no nearby pedestrians. On the other hand, the presence of a crosswalk would make a nearby pedestrian on the sidewalk a more salient part of the input context as opposed to if there was no crosswalk. Therefore, information from the two modalities should be fused before extracting the salient features for trajectory prediction. Moreover, the distribution of future trajectories is multimodal, with modes corresponding to different agent's intentions. For each agent's intention, there are elements of the scene and a set of surrounding agents that influence the execution of the corresponding motion.

Here, we propose a novel approach applying multi-head attention (MHA) by considering a joint representation of the static scene and the surrounding agents. Our method attends to the shared representation of the agent trajectories and map while projecting them into a joint space, where cross-modal correlation is computed by dot product operation. We also use each attention head to generate a distinct future trajectory to address multimodality of future trajectories. Then, we augment our approach by different methods defining drivers intentions and attributing an intention to each mode in order to enhance the trajectory prediction diversity. In addition, we propose to replace the MHA with alternative attention based modules such as the Double Attention (DA) network in order to reduce the complexity of the model while generating competitive predictions.

In the following, we start by presenting the main building blocks of our main model Multi-head Attention with Joint Agent Map Representation in Section 4.1. Specifically,

we define the inputs and the outputs of our model. Then, we describe the encoding layers of our input features, different methods of modeling interaction with the context and the decoding layer. In Section 4.2, we present an alternative method of interaction modeling and generating a multimodal trajectory predictor using a double attention module. We expand our proposed methods to deal with the problem of simultaneous multi-vehicle trajectory prediction in Section 4.3. Section 4.4 presents the experiments done on the nuScenes dataset to evaluate the performance of our proposed methods.
The first part of this work (our MHA-JAM [70]) has been done during a research visit to the University of California San Diego (UCSD) in collaboration with the Laboratory for Intelligent and Safe Automobiles (LISA) team.

4.1 Multi-head Attention with Joint Agent Map Representation (MHA-JAM)

The attention mechanism has been used for trajectory prediction with approaches using a single attention head [77, 89, 110]. In our methods Sc-RRNN, L-RRNN and MHA-LSTM presented in Chapter 3, we used multiple attention heads to extract multiple aspects of interactions between vehicles. In this part, we use MHA differently by generating attention keys and values using a joint representation of the HD map and surrounding agent motion, to model the existing spatio-temporal context interactions. Moreover, our model incorporates multimodality by using each attention head to extract a different context representation and generates a plausible trajectory conditioned on each context. We also predict the probability of each generated trajectory being the best fit to the ground truth.

In the following, we describe the inputs of our proposed methods in Section 4.1.1. Then, we present our multimodal outputs that form multiple plausible trajectories based on distinct latent representations in Section 4.1.2. After that, we introduce two MHA-based methods that model the interactions between the target vehicle and the static and dynamic scene elements in Sections 4.1.4 and 4.1.5. In Section 4.1.7, we augment our MHA-JAM with different intention definition techniques in order to enhance the diversity of the generated trajectories. We also enforce dynamic constraints on our MHA-JAM to enable it to generate dynamically-feasible trajectories in Section 4.1.8. Finally, Section 4.1.9 presents the deployed loss functions: regression loss, classification loss, off-road loss and diversity loss.

4.1.1 Input Representation

Our goal is to predict the future trajectory of a target vehicle. For this purpose, we exploit two main cues:

- The past trajectories of the target and its surrounding agents.
- The scene map presenting the road structure oriented toward the driving direction.

We define the interaction space of a target vehicle T as the area centered on its position at the prediction time t_{pred} and oriented toward its direction of motion. In the following, we consider the set of agents present in this area $\mathcal{A}_T$ and the map covering it. This representation enables us to consider different numbers of interacting agents based on the occupancy of this area.

Trajectory representation: Each agent i in the interaction space is represented by a sequence of its states, for t_h past time steps between $t_{pred} - t_h$ and t_{pred},

$$S_i = [S_i^{t_{pred}-t_h}, \ldots, S_i^{t_{pred}}]. \tag{4.1}$$

Each state is composed of a sequence of the agent relative coordinates x_i^t and y_i^t, velocity vel_i^t, acceleration acc_i^t and yaw rate ω_i^t,

$$S_i^t = (x_i^t, y_i^t, vel_i^t, acc_i^t, \omega_i^t),\ t \in t_{pred} - t_h, \ldots, t_{pred}. \tag{4.2}$$

We note S_T the state of the target vehicle T. The positions are expressed in a stationary frame of reference where the origin is the position of the target vehicle at the prediction time t_{pred}, The y-axis is oriented toward the target vehicle's direction of motion and x-axis points to the direction perpendicular to it.

Map representation: The road geometry, driving area and lane divisions of the interaction space map are fed to the model as a rasterized RGB image [21] denoted $\mathcal{M}$ (cf. Figure 4.6a).

4.1.2 Multimodal Output

The inherent uncertainty of the future imposes that there is not one single correct possible future trajectory prediction. Given a history of an agent's motion and a similar context, there are multiple possible and plausible motions of this agent in the future. In fact, the future trajectory of an agent in the scene depends on the agent's goal and behavioral characteristics or driving style and on the interaction with the surrounding road users. However, the goals and behaviors are not externally observable. Therefore, distinct possible goals could be defined and for each potential goal or behavior, there could be

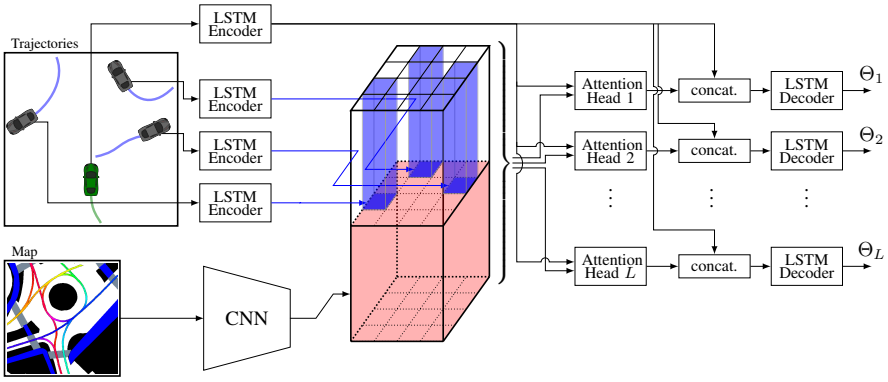


Figure 4.1: MHA-JAM (MHA with Joint Agent Map representation): Each LSTM encoder generates an encoding vector of one of the surrounding agent recent motion. The CNN backbone transforms the input map image to a 3D tensor of scene features. A combined representation of the context is build by concatenating the surrounding agents motion encodings and the scene features. Each attention head models a possible way of interaction between the target (green car) and the combined context features. Each LSTM decoder receives an context vector and the target vehicle encoding and generates a possible distribution over a possible predicted trajectory conditioned on each context.

a distinct possible future trajectory or mode. Even humans reason by accounting for multiple possibilities of future motions and put more focus on the most probable one.
To address multimodality most existing approaches build one latent representation of the context [23, 24, 35, 120] and then generate multiple possible trajectories based on this representation. However, we believe that each possible future trajectory is conditioned on a specific subset of surrounding agents' behaviors and scene context features. For each possible intention, a different partial context is important to understand the future behavior.
We wish to estimate the probability distribution $P(Y|S, \mathcal{M})$ over the future locations Y of the target vehicle, conditioned on the past trajectories S of agents in $\mathcal{A}_T$, and the map $\mathcal{M}$. To account for multimodality of $P(Y|S, \mathcal{M})$, we model it as a mixture distribution with L mixture components. Each mixture component consists of predicted location co-ordinates at discrete time-steps over a prediction horizon t_f.

$$Y_l = [Y_l^{t_{pred}+1}, \ldots, Y_l^{t_{pred}+t_f}],\ l = 1, \ldots, L. \tag{4.3}$$

Here, superscripts denote time, while subscripts denote the mixture component. Each predicted location Y_l^t is modeled as a bivariate Gaussian distribution. Our model outputs the means μ_l^t and variances Σ_l^t of the Gaussian distributions for each mixture component at each time step.

$$\Theta_l = [\Theta_l^{t_{pred}+1}, \ldots, \Theta_l^{t_{pred}+t_f}],\ \text{where } \Theta_l^t = (\mu_l^t, \Sigma_l^t),\ l = 1, \ldots, L. \tag{4.4}$$

Additionally, our model outputs the probabilities P_l associated to each generated trajectory.

4.1.3 Encoding Layer

The state vectors of the target vehicle and its surrounding agents and the rasterized representation of the map covering the interaction space are fed as input features to the encoding layer. The latter is composed of two modules:

The trajectory encoding module: The state vector S_i^t of each agent is embedded using a fully connected layer to a vector e_i^t and encoded using an LSTM encoder,

$$h_i^t = LSTM(h_i^{t-1}, e_i^t; W_{enc}), \;\; t = t_{pred} - t_h, \ldots, t_{pred}. \tag{4.5}$$

h_i^t and h_T^t are the hidden states vector of the i^{th} surrounding agent and the target vehicle respectively at time t. All the LSTM encoders share the same weights W_{enc}.

The scene feature extractor module: We use a CNN to extract high level features from map. Actually, the CNN transforms the input image to a 3D tensor F_m of size (M, N, P_m), where (M, N) corresponds to the size of the considered 2D spatial grid of features and P_m the number of features.

4.1.4 Separate Agent-Map plus Joint Conditioning MHA

The interactions of the target vehicle with the static and dynamic context cues help to infer its future trajectory. Separate attention mechanisms enable to extract the salient parts of the static scene features and the surrounding agents motion encodings and gather the essential information from each modality independently from the other. However, the latter two cues are tightly linked; each can inform the most salient parts of the other.

In this subsection, we augment the separate attention with additional module in order to establish a connection between static scene and agent motion attention and integrate information from both of them. Consequently, we model the target vehicle interactions with its context using two steps: a separate agent-map attention and a joint conditioning attention.

4.1.4.1 Separate Attention:

The separate attention is a baseline method that considers the agents motion encodings F_s and the map features F_m separately. It uses two separate attention modules. The first attends to the agent motion encoding conditioned on the target agent motion. The target agent motion is projected to form the query vector while the agents motion encodings F_s are mapped to keys and values. This module captures the agents motions that influence

the most the target vehicle future trajectory independently of the static scene context. The second module establishes attention over map regions F_m conditioned on the target agent motion encoding. Each attention module operates independently from the other. Then, we concatenate the outputs of the two attention modules (cf. $head_{al}$, $head_{ml}$ in Figure 4.2).

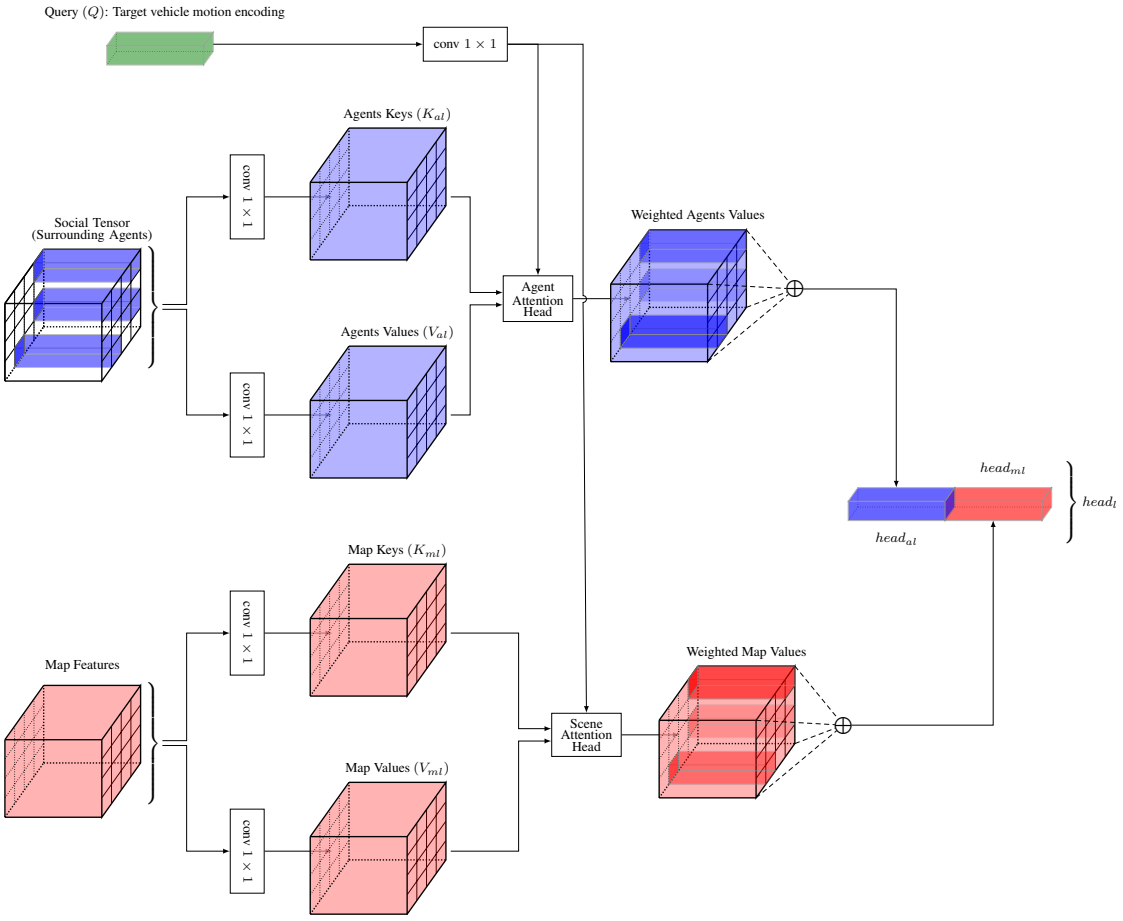


Figure 4.2: Separate agent-map representation for the attention heads: a baseline where attention weights are separately generated for the map and agents features by generating keys and values for each set of features independent of the other.

The separate attention generates an integrated vector which is simply the concatenation of the two attention vectors $head_{al}$ and $head_{ml}$ (cf. Figure 4.2). This vector is projected and the target agent encoding vector is added to it using a skip connection to form an intermediate context representation C_{int}.

4.1.4.2 Joint Conditioning Attention:

The second step of this method is also composed of two attention modules; The first one attends to the surrounding agents motion encodings and the second aggregates map features. Each module receives the intermediate context representation C_{int} and use it as

a conditioning vector (a query) of the two attention modules. Actually, this combined vector fuses information about the target agent motion, the surrounding agents trajectories and the static scene to form a joint representation. Therefore, it is used in the first attention module to attend to surrounding agents motion encodings. It extracts from them features conditioned on the target agent motion encoding and the static scene context. Similarly, the combined vector is used as a query for the second attention module. It captures the important map features based on the target agent motion encoding and its surrounding agents motions. The output vectors of the two steps and the target agent motion encoding are combined to form the context vector z_l.
This method enables to model the dependencies between the dynamic agents of the static scene elements using the joint conditioning attention. To do so, it deploys four attention modules which is computationally expensive. As a remedy, we propose another method based on a joint agent-map representation.

4.1.5 Joint Agent-Map Attention

The first step in modeling vehicle-agents and vehicle-map interactions here, is to build a combined representation of the global context. To do so, we first build a social tensor F_s composed of the trajectories encoding of the surrounding agents $h_i^{t_{pred}}$ placed on their corresponding positions on top of the 2D spatial grid of size (M, N) covering the interaction space,

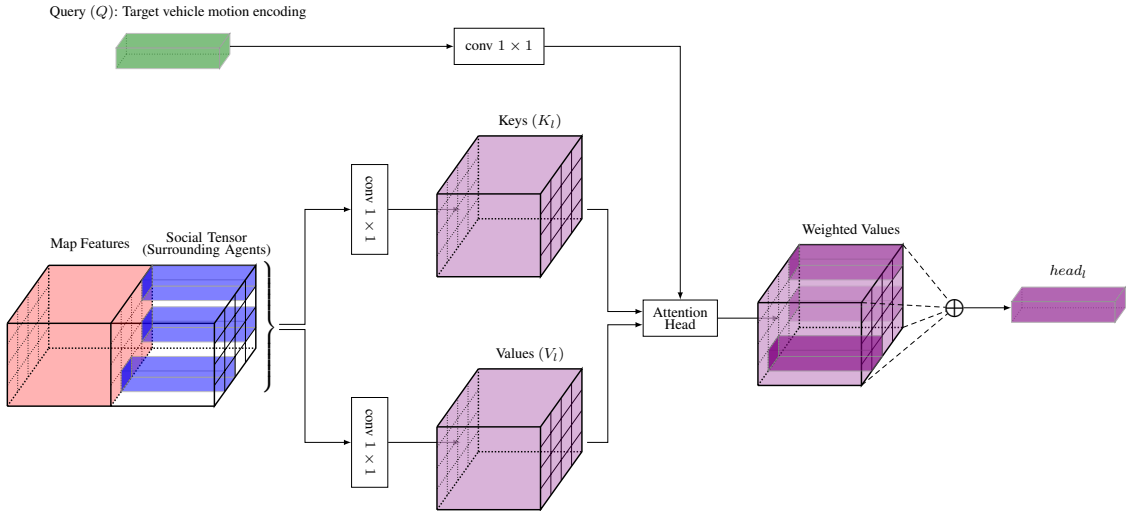


Figure 4.3: Attention modules in MHA-JAM: We generate keys and values by applying 1x1 convolutional layers to a joint representation of the map and surrounding agents, while the trajectory encoding of the target agent serves as the query.

$$F_s(m, n, :) = \delta_{mn}(x_i^{t_{pred}}, y_i^{t_{pred}}) h_i^{t_{pred}}, \quad \forall i \in \mathcal{A}_T, \tag{4.6}$$

where $\delta_{mn}(x, y)$ is an indicator function equal to 1 if and only if (x, y) is in the cell (m, n) of the 2D spatial grid at the prediction time.
Then, we concatenate the social features F_s and map features F_m to generate a spatio-temporal representation of the global context C of size (M, N, P_c),

$$C = Concat(F_s, F_m). \tag{4.7}$$

We use the attention mechanism to explicitly capture salient combined spatio-temporal context features composed of maps and trajectories as follows (cf. Figure 4.3):

- The hidden state of the target vehicle is projected to form different queries $Q_l = \theta_l(h_T^{t^{pred}}, W_{\theta_l})$.
- The combined trajectories and map features are projected in a joint space to form different keys $K_l = \phi_l(C, W_{\phi_l})$ and values $V_l = \rho_l(C, W_{\rho_l}),\ l = 1 \dots L$.

θ_l, ϕ_l and ρ_l are linear functions with the weight matrices, learned in each attention head l, W_{θ_l}, W_{ϕ_l} and W_{ρ_l}.
An attention feature $head_l$ is calculated as a weighted sum of values vectors $V_l(m, n, :)$,

$$head_l = \sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_l(m, n) V_l(m, n, :),\ \ l = 1, \dots, L. \tag{4.8}$$

$\alpha_l(m, n)$ weights the effect of each context feature vector (m, n) on the target vehicle future trajectory,

$$\alpha_l(m, n) = softmax(\frac{Q_l K_l^T(m, n, :)}{\sqrt{d_m}}). \tag{4.9}$$

$Q_l K_l^T(m, n, :)$ is a vector multiplication used to calculate the dot product between the query Q_l and each key context feature vector $K_l^T(m, n, :)$. d_m is a scaling factor that equals to the dimensionality of the projection space.
We use multiple attention heads $l = 1, \dots, L$ to extract different latent representations of the context z_l.

$$z_l = Concat(h_T^{t_{pred}}, head_l),\ \ l = 1, \dots, L. \tag{4.10}$$

We use each context representation z_l to generate a trajectory $\hat{Y}^l$.

The main difference between the separate MHA and joint MHA is that the latter generates keys and values in MHA using a joint representation of the map and agents while the separate MHA computes keys and values of the map and agents features separately.

4.1.6 Decoding Layer

Each context vector z_l, representing the selected information about the target vehicle's interactions with the surrounding agents and the scene, and its motion encoding are fed to l LSTM Decoders. The decoders generate the predicted parameters of the distributions over the target vehicle's estimated future positions of each possible trajectory for next t_f time steps,

$$\Theta_l^t = \Lambda(LSTM(h_{dec}^{t-1}, z_l; W_{dec})), t = t_{pred} + 1, \ldots, t_{pred} + t_f. \tag{4.11}$$

All the LSTM decoders share the same weights W_{dec} and Λ is a linear layer.
Similar to MTP [21], we also predict the likelihood of each predicted trajectory. To do so, we concatenate all the scene representation vectors z_l, feed them to two successive fully connected layers separated by an activation function. Then, we apply the softmax function to estimate the probability of each trajectory.

4.1.7 Intention-based Prediction

A multimodal trajectory prediction solution consists in generating multiple plausible trajectories corresponding to the possible future motions of a target vehicles. In order to enhance the prediction performance, we attempt to ensure that the predicted trajectories are diverse and cover most of the possible future paths using an intention based trajectory prediction. In the following, we aim to enhance the diversity of the generated trajectories in a multimodal solution by associating each predicted trajectory to a distinct intention. The challenge here is how to define different intentions that guarantee the generation of diverse trajectories.

In our basic MHA-JAM, we use multiple attention heads to generate different trajectories. We assign to the network the task of learning different modes and generating trajectories without defining a discrete implicit intention of each generated mode or trajectory class. We deploy the best of L regression loss to train our model in order to generate a diverse set of predicted trajectories (cf. Section 4.1.9).

In order to enhance the diversity of the predicted trajectories, we extend our approach using three intention definition techniques: goal directed, anchors and region based intention definition.

4.1.7.1 Goal directed Trajectory Generation

A first intention definition technique consists in defining goals based on the final destinations of the agents. In order to generate diverse trajectories that cover different possible paths. We split the interaction space to distinct predefined regions corresponding to specific goals or destinations $gl_l,\ l = 1..L$. Then, we train each attention head to learn to generate trajectories with goals (end point or destination after the prediction time t_f) in one specific region. This approach consists in defining trajectories classes based on their goals. Each class characterises the trajectories having an endpoint in a specific region. To do so, we proceed by dividing the training set trajectories into subsets accordingly (the number of subsets equals the number of prediction modes). Then, we train each prediction head to generate the trajectories corresponding to a specific class. In this case, we don't use the minimum over L loss. We compute the regression loss between the ground truth trajectory with goal gl_l and the predicted trajectory by the prediction head l. Figure 4.4 shows the interaction space division into $L = 16$ regions used in our experiments on the nuScenes dataset. To divide the interaction space into different destinations, we apply the K-means clustering algorithm on the end points of the future trajectories of the training data prior to the training and we define each region by the center of each cluster.

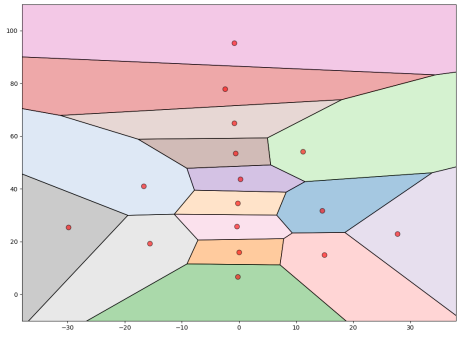


Figure 4.4: Interaction space Division into distinct Destination regions. The lines delimit each considered destination region and the red dots present the centers of each endpoints' cluster.

4.1.7.2 Anchor based Trajectory Generation

Taking into account the destination helps to determine the intention of an agent. However, the general path pattern followed by the agent reveals additional information about its future motion. Therefore, we investigate the use of a fixed set of future state-sequence or anchor trajectories that correspond to modes of the trajectory distribution [16] as the

generally followed trajectories. To generate the anchors, we use the K-means clustering over the training set trajectories using the Euclidean distance and we consider the center of each cluster. Figure 4.5 presents the generated anchors when the number of clusters that corresponds to the number of modes equals 16. We notice that the anchors describe different trajectories with different patterns, destinations and velocities. During the training, each attention head generates trajectories that are the closest to a specific anchor with reference to the Euclidean distance.

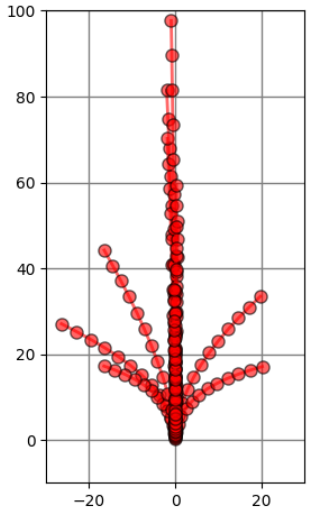


Figure 4.5: Anchor Trajectories

4.1.7.3 Region based Trajectory Generation

The possible trajectories of a target vehicle can be inferred using the map (road structure and lanes configuration) and surrounding agents information. Our model learns to extract different representations of the context using different attention heads. Each extracted context representation corresponds to a latent intention. In order to enhance the diversity of the predicted trajectories, we operate on the attention weight corresponding to each attention head in order to direct the attention to a specific region.

A first way to define intentions is to split the generated joint map and agent features (with respect to the spatial dimension) into regions prior to the training and force each attention head to accord attention only to a specific region. In the evaluation section, we present an application of this method with three attention heads and therefore we divide the context into three regions (cf. Figure 4.15). This method can be extended using heuristics to divide the context into a bigger number of regions.

A second way to enhance the diversity of the generated trajectories is to force the attention heads to learn to specialise in specific regions using an auxiliary loss function denoted diversity loss (cf. Section 4.1.9).

4.1.8 Dynamically-feasible Prediction

In order to ensure that the predicted positions are realizable by the vehicle control, we incorporate dynamics constraints to our proposed model **MHA-JAM**.

4.1.8.1 Dynamically-extended Unicycle Model

Similar to [91], we deploy the dynamically-extended unicycle [57] to model vehicles dynamics control. In the real world, vehicles are controlled by accelerator pedals and the steering wheel. Therefore, we choose the dynamically-extended unicycle model since it utilizes the acceleration acc and the yaw rate ω as control inputs. It has the following nonlinear continuous-time dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{vel} \end{bmatrix} = \begin{bmatrix} vel\cos(\theta) \\ vel\sin(\theta) \\ \omega \\ acc \end{bmatrix} \tag{4.12}$$

where x and y are the coordinates, vel is the velocity, and θ is the heading. Since we consider discrete trajectories, we use the following discrete equivalent dynamics where Δt represents the length of the sampling interval.

$$\begin{bmatrix} x^{(t+1)} \\ y^{(t+1)} \\ \theta^{(t+1)} \\ vel^{(t+1)} \end{bmatrix} = \begin{bmatrix} x^{(t)} \\ y^{(t)} \\ \theta^{(t)} \\ vel^{(t)} \end{bmatrix} + \begin{bmatrix} vel^{(t)} D_S^{(t)} + \frac{acc^{(t)}\sin(\theta^{(t)}+\omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} + \frac{acc^{(t)}}{\omega^{(t)}} D_C^{(t)} \\ -vel^{(t)} D_C^{(t)} + \frac{acc^{(t)}\cos(\theta^{(t)}+\omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} + \frac{acc^{(t)}}{\omega^{(t)}} D_S^{(t)} \\ \omega^{(t)}\Delta t \\ acc^{(t)}\Delta t \end{bmatrix} \tag{4.13}$$

where $D_S^{(t)} = \dfrac{\sin(\theta^{(t)} + \omega^{(t)}\Delta t) - \sin(\theta^{(t)})}{\omega^{(t)}}$

$D_C^{(t)} = \dfrac{\cos(\theta^{(t)} + \omega^{(t)}\Delta t) - \cos(\theta^{(t)})}{\omega^{(t)}}$

We use a slightly adapted set of dynamics [91] when $|\omega| < 10^{-3}$ to avoid singularities in Equation 4.13. When the yaw rate value ω is small, we adopt the following dynamics, obtained by computing the limit as $\omega \to 0$.

$$\begin{bmatrix} x^{(t+1)} \\ y^{(t+1)} \\ \theta^{(t+1)} \\ vel^{(t+1)} \end{bmatrix} = \begin{bmatrix} x^{(t)} \\ y^{(t)} \\ \theta^{(t)} \\ vel^{(t)} \end{bmatrix} + \begin{bmatrix} vel^{(t)}\cos(\theta^{(t)}\Delta t + 0.5\, acc^{(t)}\cos(\theta^{(t)}(\Delta t)^2 \\ vel^{(t)}\sin(\theta^{(t)}\Delta t + 0.5\, acc^{(t)}\sin(\theta^{(t)}(\Delta t)^2 \\ 0 \\ acc^{(t)}\Delta t \end{bmatrix} \tag{4.14}$$

In summary, the full dynamics are:

$$\begin{bmatrix} x^{(t+1)} \\ y^{(t+1)} \\ \theta^{(t+1)} \\ vel^{(t+1)} \end{bmatrix} = \begin{cases} \text{Equation } (4.13) & \text{if } |\omega| > \epsilon \\ \text{Equation } (4.14) & \text{otherwise} \end{cases} \tag{4.15}$$

4.1.8.2 Output Form

At each timestep, and for a specific latent value z_l, our **MHA-JAM** generates a Gaussian distribution over control actions instead of directly generating a Gaussian distribution over positions as deployed previously. Then, the dynamically-extended unicycle model is applied to obtain the positions sequence corresponding to the control actions distribution generated by our **MHA-JAM**. In this case, our **MHA-JAM** outputs:

$$\mu_{\mathbf{u}} = \begin{bmatrix} \mu_{\omega} \\ \mu_{acc} \end{bmatrix} \qquad \mathbf{\Sigma}_{\mathbf{u}} = \begin{bmatrix} \sigma_{\omega}^2 & \rho_{\omega acc}\sigma_{\omega}\sigma_{acc} \\ \rho_{\omega acc}\sigma_{\omega}\sigma_{acc} & \sigma_{acc}^2 \end{bmatrix} \tag{4.16}$$

Mean Positions In order to obtain the output mean positions described in Section 4.1.2, we apply the Equation 4.15 to the the mean control actions generated our **MHA-JAM**.

Covariance of the Positions In order to obtain the covariance, the Jacobians F and G of the system dynamics in Equation 4.13 are computed as follows:

$$\mathbf{F}^{(t)} = \frac{\partial \mathbf{f}}{\partial \mu_{\mathbf{s}}{}^{(t)}} = \begin{bmatrix} 1 & 0 & vel^{(t)} D_c^{(t)} - \frac{acc^{(t)} D_S^{(t)}}{\omega^{(t)}} + \frac{acc^{(t)} \cos(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} & D_S^{(t)} \\ 1 & 0 & vel^{(t)} D_S^{(t)} + \frac{acc^{(t)} D_C^{(t)}}{\omega^{(t)}} + \frac{acc^{(t)} \sin(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} & -D_C^{(t)} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{G}^{(t)} = \frac{\partial \mathbf{f}}{\partial \mu_{\mathbf{u}}{}^{(t)}} = \begin{bmatrix} G_{11}^{(t)} & \frac{D_C^{(t)}}{\omega^{(t)}} + \frac{\sin(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} \\ G_{21}^{(t)} & \frac{D_S^{(t)}}{\omega^{(t)}} - \frac{\cos(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$$

$$\begin{aligned} \text{where } G_{11}^{(t)} =& \frac{vel^{(t)} \cos(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} - \frac{vel^{(t)} D_S^{(t)}}{\omega^{(t)}} - \frac{2acc^{(t)} \sin(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{{\omega^{(t)}}^2} - \frac{2acc^{(t)} D_C^{(t)}}{{\omega^{(t)}}^2} \\ &+ \frac{acc^{(t)} \cos(\theta^{(t)} + \omega^{(t)}\Delta t)(\Delta t)^2}{\omega^{(t)}} \\ G_{21}^{(t)} =& \frac{vel^{(t)} \sin(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{\omega^{(t)}} + \frac{vel^{(t)} D_C^{(t)}}{\omega^{(t)}} + \frac{2acc^{(t)} \cos(\theta^{(t)} + \omega^{(t)}\Delta t)\Delta t}{{\omega^{(t)}}^2} - \frac{2acc^{(t)} D_S^{(t)}}{{\omega^{(t)}}^2} \\ &+ \frac{acc^{(t)} \sin(\theta^{(t)} + \omega^{(t)}\Delta t)(\Delta t)^2}{\omega^{(t)}} \end{aligned}$$

Then, by applying the equations for the covariance of a sum of Gaussian random variables [101], the covariance in position space $\mathbf{\Sigma}_{\mathbf{p},\theta,v}$ is obtained as follows:

$$\mathbf{\Sigma}_{\mathbf{p},\theta,v}^{(t+1)} = \mathbf{F}^{(t)} \mathbf{\Sigma}_{\mathbf{p},\theta,v}^{(t)} {\mathbf{F}^{(t)}}^T + \mathbf{G}^{(t)} \mathbf{\Sigma}_{\mathbf{u}}^{(t)} {\mathbf{G}^{(t)}}^T \tag{4.17}$$

4.1.9 Loss Functions

We train the model using the following loss functions:

4.1.9.1 Regression loss

While the model outputs a multimodal predictive distribution corresponding to L distinct futures, we only have access to 1 ground truth trajectory for training the model. In order to not penalize plausible trajectories generated by the model that do not correspond to the ground truth, we use a variant of the best of L regression loss for training our model, as has been previously done in [35]. This encourages the model to generate a diverse set of predicted trajectories. Since we output the parameters of a bivariate Gaussian distribution at each time step for the L trajectories, we compute the Negative Log-Likelihood (NLL) of the ground truth trajectory under each of the L modes output by the model, and consider the minimum of the L NLL values as the regression loss. The regression loss is given by

$$L_{reg} = \min_{l} \sum_{t=t_{pred+1}}^{t_{pred}+t_f} -log(P_{\Theta^t}(\mathbf{y}_l^t|S)). \tag{4.18}$$

4.1.9.2 Classification loss [21]

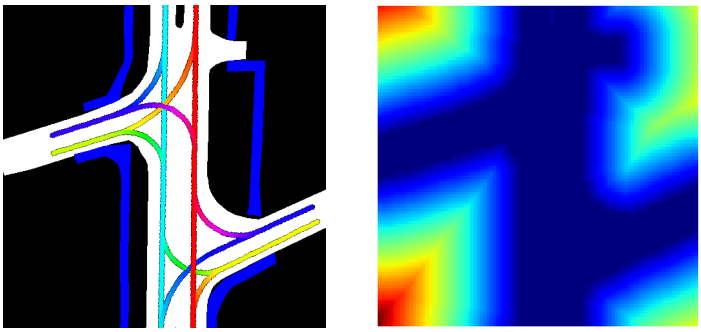
In addition to the regression loss, we consider the cross entropy,

$$L_{cl} = -\sum_{l=1}^{L} \delta_{l^*}(l) log(p_l), \tag{4.19}$$

where δ is a function equal to 1 if $l = l^*$ and 0 otherwise. Here l^* is the mode corresponding to the minimum NLL in equation 4.18. $\hat{Y}^{l^*}$ is the predicted trajectory corresponding to l^* and p_l^* its predicted probability.

4.1.9.3 Off-road loss

While the loss given by equation 4.18 encourages the model to generate a diverse set of trajectories, we wish to generate trajectories conforming to the road structure. Since the regression loss only affects the trajectory closest to the ground-truth, we consider the auxiliary loss function proposed in [73, 77] that penalizes points in any of the L trajectories that lie off the road. The off-road loss L_{or} for each predicted location is the minimum distance of that location from the drivable area. Figure 4.6b shows a heatmap of the off-road loss for the layout in figure 4.6a.



(a) Input scene representation **(b)** Off-road loss distance map

Figure 4.6: Off-road loss: an auxiliary loss function that penalizes locations predicted by the model the fall outside the drivable area. It is proportional to the distance of a predicted location from the nearest point on the drivable area.

4.1.9.4 Diversity loss

In order to avoid generating similar predictions and ensure that the attention mechanism does not provide similar attention weights for all attention heads, we use a diversity loss. This loss encourages the network to attend to a specific aspect (set of features) consistently. It incites the attention map for each head to focus on a single region and different heads attend to distinct regions.
We define the diversity loss as a soft subspace orthogonality constraint between the attention weight vector (formed from attention map) of each head (space) as follows:

$$L_{div} = \frac{1}{Nb_{samples}} \sum_{m=1}^{Nb_{samples}} \sum_{p=1}^{L} \sum_{\substack{q=1 \\ p \neq q}}^{L} \sum_{k=1}^{MN} \alpha_p(k)\alpha_q(k) \tag{4.20}$$

where α_p are the attention weights of the attention head p and $Nb_{samples}$ is the number of samples considered.
The overall loss for training the model is given by

$$Loss = L_{reg} + \lambda_{cl}.L_{cl} + \lambda_{or}.L_{or} + \lambda_D.L_D, \tag{4.21}$$

where the weights λ_{cl}, λ_{or} and λ_D are empirically determined hyperparameters.

In this section, we introduced our main contribution: the MHA with Joint Agent Map representation method (**MHA-JAM**). It deploys multiple attention heads and uses each head to generate a plausible trajectory. We also deployed three different ways to define intentions and forced each attention head to generate trajectories corresponding to a specific intention. In the following, we propose another attention based method that generates multiple possible trajectories based on global features of the joint agent and map context.

4.2 Double Attention Based Model

In order to predict trajectories of its surrounding agents, an autonomous vehicle needs to analyze the traffic scene and build a global understanding of it. This understanding requires different cues from the target vehicle and its context. In addition, a vehicle's trajectory is very correlated to the behavior of its surrounding agents and to the static scene context and salient context information lies in both nearby and distant elements. e.g., intersection road, crosswalks. Therefore, recent studies about trajectory prediction deploy attention mechanism in order to capture long-range dependencies to represent interactions between vehicles and extract the global understanding of the visual scene. However, the complexity of multi-head attention block is important since attention captures pair-wise relations between the target vehicle representation and all feature map positions.
In this part, we deploy the attention mechanism differently to reduce the complexity of the multi-head attention block deployed in the task of trajectory prediction by generating a query-independent attention map that capture the context global features. Then, we aggregate specific context representations based on the target vehicle trajectory using this attention map to form a multimodal output. This attention block has less computation cost than the original multi-head attention block, and it infers the future trajectories with almost no decrease in accuracy.

4.2.1 Input and Output Representation

In our double attention (DA) based trajectory prediction approach, we use the same input representations of agents trajectories and their surrounding scene as for our Multi-head Attention with Joint Agent Map Representation (cf. Section 4.1).

4.2.2 Encoding and Decoding layers

We use the same encoding layers to generate feature representations of the past trajectories S_i of each agent i in the interaction space and the map image $\mathcal{M}$. Similarly, we use the same decoding layer to output multimodal predictions.

4.2.3 Multi-head Double Attention

MHA consists in mapping the target vehicle to queries, and the 3D tensor presenting the map and the surrounding agents features to keys and values. Then, it generates attention vectors for the target vehicle. Such an approach exhaustively correlates the target vehicle motion encoding with that of its neighbors and the map features. The DA models spatial relationships in a computationally efficient manner.

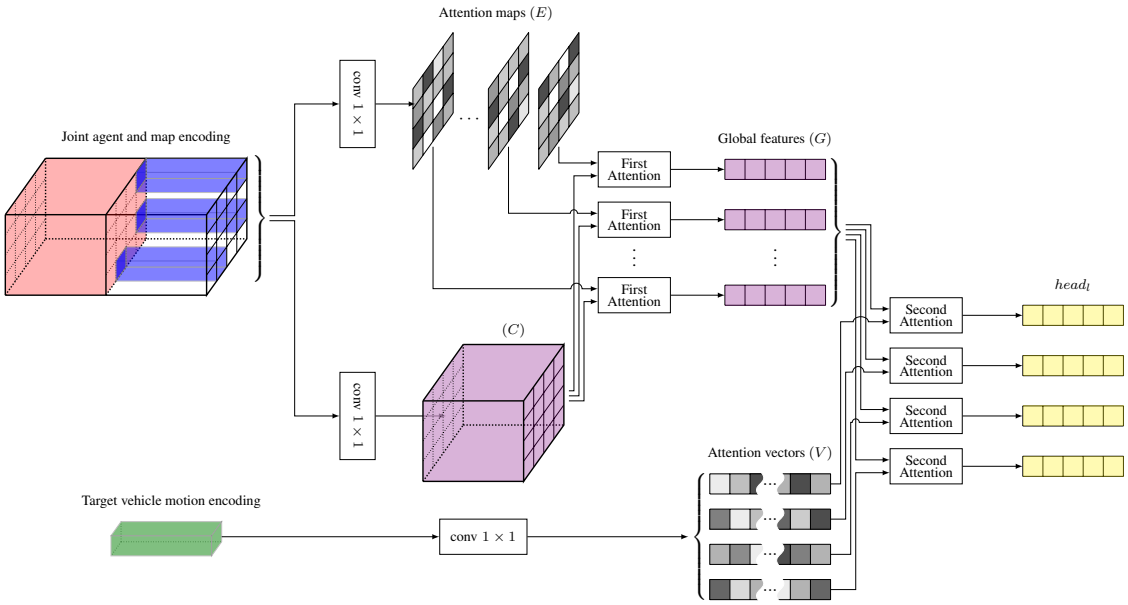


Figure 4.7: Double Attention Block applied on the joint agent and map representation conditioned on the target vehicle motion encoding.

4.2.4 Double Attention for Scene Interaction (DA-JAM)

In a driving neighborhood, the drivers' behaviors are correlated, which leads to similarities in trajectories. However, distinct behaviors always exist due to different driving styles and goals. Therefore, if we consider the entire interaction space, trajectories can be grouped together when they are similar and distinguishable ones can be extracted to form a compact representation of trajectory features. Moreover, some static elements of the scene (e.g. lanes, road borders, crosswalks ...) have more influence on the vehicles trajectories than others. Therefore, the key scene features can be aggregated together with the surrounding agents motions to form a global understanding of the traffic context. Then, the collective impact of the selected set of context features on each possible future behavior of the target vehicle can be extracted.
This process can be modeled by a DA mechanism, introduced by Chen et al. [18] for image/video recognition, to capture long-range vehicle motion feature interdependencies via universal gathering and distribution functions. It is composed of two steps:

- **Global features extractions**: gathers a set of key scene features from the entire grid into a compact set using attention.
- **Adaptive features distribution**: adaptively selects and distributes the key scene features to form different contexts for the target vehicle possible future behavior via another attention.

4.2.4.1 Global features extractions

The first step of the DA network consists in generating global features. To do so, the 3D tensor F_m presenting the joint context features is projected into two different spaces to form two context representations C and E. Actually, C and E are the outputs of the two different convolution layers (kernel size $= (1, 1)$) applied to the context features F_m.

$$C = \theta(F_m; W_\theta)^T,\ C = [\mathbf{c}_1, \ldots, \mathbf{c}_{MN}] \in \mathbb{R}^{d_{GF} \times MN} \tag{4.22}$$

$$E = \phi(F_m; W_\phi)^T,\ E = [\mathbf{e}_1, \ldots, \mathbf{e}_{MN}] \in \mathbb{R}^{N_{GF} \times MN}, \tag{4.23}$$

where d_{GF} and N_{GF} are the dimension and the number of the global features.
We rewrite the feature matrix E as:

$$E = [\bar{\mathbf{e}}_1, \ldots, \bar{\mathbf{e}}_{N_{GF}}], \tag{4.24}$$

where $\bar{\mathbf{e}}_i \in \mathbb{R}^{MN}$ is an MN-dimensional row vector of the matrix E.
The global features $G = \left[\mathbf{g}_1, \ldots, \mathbf{g}_{N_{GF}}\right]$ are computed as:

$$\mathbf{g}_i = C\ softmax(\bar{\mathbf{e}}_i)^T \tag{4.25}$$

The term $softmax(\bar{\mathbf{e}}_i)$ corresponds to the i^{th} global attention map. This map is composed of attention weights extracted from the context features F_m independently of the input sequence of states.
We denote G_{gather} the function that adaptively aggregates global features from the entire input joint context feature map F_m:

$$G_{gather}(F_m) = Csoftmax(E)^T \tag{4.26}$$

4.2.4.2 Adaptive features distribution

The previously extracted global features are used to generate specific features adapted to the target vehicle past motion. To do so, we project the target vehicle motion encoding in L different spaces of dimension N_{GF}.

$$V = softmax(\rho(h_T^{t_{pred}}; W_\rho)),\ V = [\mathbf{v}_1, \ldots, \mathbf{v}_L] \in \mathbb{R}^{N_{GF} \times L} \tag{4.27}$$

We note that $\sum_{j=1}^{N_{GF}} v_{lj} = 1$. The vector $\mathbf{v}_l$ represents the attention weights of the global vectors based on the target vehicle trajectory.
The function F_{distr} distributes the gathered information to form a different context

vectors characterizing each possible trajectory, conditioned on the target vehicle motion encoding vector v_l:

$$head_l = F_{distr}(G_{gather}(F_m), v_l) \tag{4.28}$$

$$= G_{gather}(F_m)v_l \tag{4.29}$$

$$= \sum_{\forall j} v_{lj}\mathbf{g}_j \tag{4.30}$$

In sum, the DA network is composed of two attention blocks; the first gathers the global features from the entire context representation and the second distributes them to form different context representations, taking into account the target vehicle past motion encoding. We denote these context representations $head_l$, $(l = 1, \ldots, L)$ since they play the same role as the attention heads deployed in the MHA-JAM method. But, they are built in a more optimal way. It reduces the computation of MHA-JAM by LMN/N_{GF}^2. Actually, instead of relating the target vehicle motion encoding to every element of the context feature map, the double attention computes the correlation between the target vehicle motion encoding and the query independent global features.
As a result, each possible trajectory representation is composed of the target vehicle motion encoding and its customized global information that is complementary to the motion encoding features, facilitating the learning of complex relations with the target vehicle surrounding environment.
These trajectory representations are fed to the LSTM decoder that generates simultaneously the predicted possible trajectories of the target vehicle. We use each representation to generate a trajectory $\hat{Y}^l$,

$$z_l = Concat(h_T^{t_{pred}}, head_l), \;\; l = 1, \ldots, L. \tag{4.31}$$

4.3 Simultaneous multi-vehicle trajectory prediction

Let us now tackle the task of simultaneous multi-vehicle trajectory prediction. For this purpose, we jointly reason about the interactions between a set of target vehicles and their static and dynamic context elements and predict their future trajectories accordingly.

4.3.1 Problem Definition

In this section, we consider the problem of jointly predicting the future trajectories of a selected set of vehicles belonging to the interaction area while considering their interactions with their static and dynamic contexts. This problem is challenging since we have limited information about the influence of neighboring vehicles on the borders of

the interaction area. However, it represents a more realistic representation of the driving conditions with potential limited visibility.
In order to predict the trajectories of multiple vehicles simultaneously, we use two types of frame of reference; one related to the interaction area and one associated to each agent. To characterise the past trajectory of each vehicle in the scene, we use:

- its coordinates at the prediction time and its orientation in a stationary frame of reference related to the interaction area.
- its past trajectory states (coordinates x_i^t and y_i^t, velocity vel_i^t, acceleration acc_i^t and yaw rate ω_i^t) with reference to a stationary frame of reference centered on its position at the prediction time and oriented toward its direction of motion.

4.3.2 Multi-Head Attention for simultaneous multi-vehicle trajectory prediction

A first idea to work on the problem of simultaneous multi-vehicle trajectory prediction would be to extend the target vehicle trajectory prediction method presented in Section 4.1 to a subset of vehicles present in the interaction area and perform a similar prediction process for each vehicle with a surrounding limited to the interaction area. This consists in mapping the considered subset of target vehicles to queries and the surrounding joint agent map context to keys and values and generating attention heads for these vehicles. Such an approach exhaustively correlates each vehicle motion encoding with all the features of its context with each attention head. This method will be our baseline method for simultaneous multi-vehicle trajectory prediction, called the Multi-Vehicle MHA-JAM (**MV-MHA-JAM**), and compared with the following approach that uses the double attention mechanism.

4.3.3 Multi-Head Double Attention for simultaneous multi-vehicle trajectory prediction

Here, we use the DA mechanism for simultaneous multi-vehicle trajectory prediction to avoid relating each target vehicle to each of the context features. To do so, first, we propose to extract global features from the joint Agent Map context to form a compact representation of the context features. Then, the collective impact of the extracted global features on each of the considered target vehicles in the scene can be extracted. Our **MV-MHDA-JAM** method is composed of two steps (cf. Figure 4.8):

- **Global features extractions**: gathers a set of key context features from the joint Agent Map context to form a compact set using a first attention mechanism.

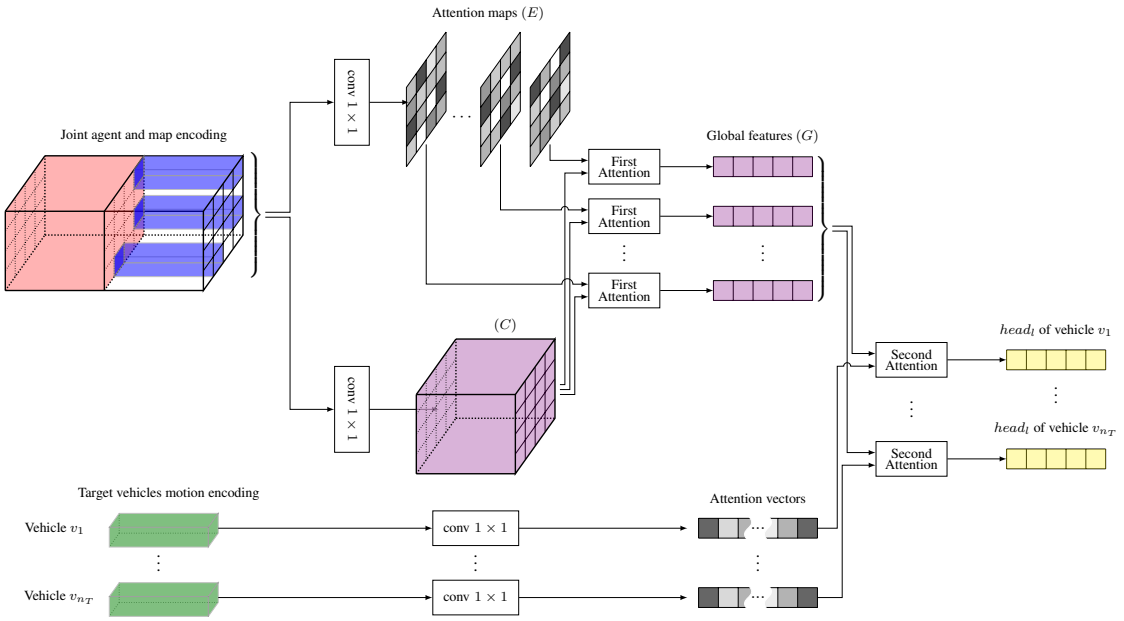


Figure 4.8: Double Attention Block applied on the joint agent and map representation for simultaneous multiple trajectory prediction. This figure presents the prediction of one mode of trajectory for all the target vehicles. We use L double attention blocks to generate multimodal predictions.

- **Adaptive features distribution**: adaptively selects and distributes the key context features to each vehicle to represent its interaction with the context via another attention mechanism. We generate the trajectories of all the target vehicles based on the same global context representation in order to increase the coherence between the generated trajectories.

In addition, in order to generate multimodal predictions, we use multiple DA heads and each head generates a possible trajectory for all the target vehicles. In other words, instead of producing one global context, we extract L global contexts and we generate possible trajectories for all target vehicles based on each global context.

4.4 Experimental Analysis and Evaluations

In order to evaluate our proposed methods, we train and test them on the publicly available naturalistic dataset nuScenes. First, we compare our proposed methods to recent state of the art methods according to different metrics. Then, we evaluate the relative contributions of various cues and modules on the overall performance of our main method MHA with Joint Agent Map representation (MHA-JAM) using ablation experiments. Moreover, we conduct a qualitative evaluation on our MHA-JAM where we visualize and analyse the generated trajectories and the attention maps. We also analyse the performance our MHA-JAM when defining a distinct intention to each generated trajectory.

4.4.1 Training and Implementation Details

The input states are embedded in a space of dimension 32. We use an image representation of the scene map (cf. figure 4.6a) of size of $(500, 500)$ with a resolution of 0.1 meters per pixel. Similar to [81] representation, our input image extents are $40\ m$ ahead of the target vehicle, $10\ m$ behind and $25\ m$ on each side. We use ResNet-50 pretrained on ImageNet to extract map features. This CNN outputs a map features of size $(28, 28, 512)$ on top of them we place the trajectories encodings. The deployed LSTM encoder and decoder are of 64 and 128 randomly initialized units respectively. For MHA-JAM, we use $L = 16$ parallel attention operations applied on the vectors projected on different spaces of size $d_m = 64$. Concerning our DA-JAM, we use $N_{GF} = 16$ global features with the dimension of the model $d_m = 64$ and $L = 10$ output modes. We use a batch size of 32 and Adam optimizer [52]. The model is implemented using PyTorch [78].

4.4.2 Evaluation Metrics

MinADE$_K$ and MinFDE$_K$: We report the minimum average and final displacement errors over K most probable trajectories similar to prior approaches for multimodal trajectory prediction [16, 21, 25, 35, 59].

$$\text{MinADE}_K = \min_k \left(\frac{1}{t_f - t_{pred}} \sum_{t=t_{pred}+1}^{t_f} \left\| Y^t - Y_K^t \right\| \right), \tag{4.32}$$

$$\text{MinFDE}_K = \min_k \left\| Y^{t_f} - Y_K^{t_f} \right\|, \tag{4.33}$$

Our model outputs the likelihood of each of the L generated trajectories. $Y_1, \ldots, Y_K$ are the K most probable trajectories.

The minimum over K avoids penalizing the model for generating plausible future trajectories that don't correspond to the ground truth.

Miss rate: For a given distance d, and the K most probable predictions generated by the model, the set of K predictions is considered a miss based on,

$$\text{Miss}_{k,d} = \begin{cases} 1 \text{ if } \min_{\hat{\mathbf{y}} \in P_K} \left(\max_{t=t_{pred}}^{t=t_f} \left\| \mathbf{y}^t - \hat{\mathbf{y}}^t \right\| \right) \geqslant d. \\ 0 \quad \text{otherwise} \end{cases} \tag{4.34}$$

The miss rate MissRate$_{K,d}$ computes the fraction of missed predictions over the test set.

Off-road rate: Similar to [25], we consider the off-road rate, which measures the fraction of predicted trajectories that fall outside the drivable area of the map.

Diversity score: Similar to the diversity loss, we use the diversity score to measure the similarity between the attention maps of each head of predicted trajectories on the test set samples.

$$S_{div} = \frac{1}{Nb_{samples}} \sum_{m=1}^{Nb_{samples}} \sum_{p=1}^{L} \sum_{\substack{q=1 \\ p \neq q}}^{L} \sum_{k=1}^{MN} \alpha_p(k)\alpha_q(k) \tag{4.35}$$

4.4.3 Models Compared

We compare our model to six baselines, two physics based approaches, and four recently proposed models that represent the state of the art for multimodal trajectory prediction. All deep learning based models generate up to $L = 25$ trajectories and their likelihoods. We report the results considering the K most probable trajectories generated by each model.

4.4.3.1 Baselines

Constant velocity and yaw : Our simplest baseline is a physics based model that computes the future trajectory while maintaining constant velocity and yaw of the current state of the vehicle. Given the position (x_T^t, y_T^t) and the velocity (vel_{Tx}^t, vel_{Ty}^t) of the target vehicle, the model computes the next position using the following equation:

$$\begin{bmatrix} x_T^{t+1} \\ y_T^{t+1} \end{bmatrix} = \begin{bmatrix} x_T^t + vel_{Tx}^t \Delta t \\ y_T^t + vel_{Ty}^t \Delta t \end{bmatrix} \tag{4.36}$$

where Δt is the time gap between positions (x_T^t, y_T^t) and (x_T^{t+1}, y_T^{t+1}).

Physics oracle : An extension of the physics based model introduced in [81]. Based on the current state of the vehicle (position, velocity, acceleration, yaw and yaw rate), it computes the minimum average point-wise Euclidean distance over the predictions generated by four on different kinematic models:

- Constant velocity and yaw: the model previously explained (cf. Equation 4.36).
- Constant velocity and yaw rate: Assuming that the velocity vel_T^t and yaw rate $\dot{\theta}_T^t$ are constants and equal vel_T^{tobs} and $\dot{\theta}_T^{tobs}$ respectively, we compute the future positions as follows:

$$\begin{bmatrix} x_T^{t+1} \\ y_T^{t+1} \\ \theta_T^{t+1} \end{bmatrix} = \begin{bmatrix} x_T^t + vel_T^{tobs} cos(\theta_T^t)\Delta t \\ y_T^t + vel_T^{tobs} sin(\theta_T^t)\Delta t \\ \theta_T^t + \dot{\theta}_T^{tobs}\Delta t \end{bmatrix} \tag{4.37}$$

- Constant acceleration and yaw: With constant acceleration and the heading, we compute the future positions as follows:

$$\begin{bmatrix} x_T^{t+1} \\ y_T^{t+1} \end{bmatrix} = \begin{bmatrix} x_T^t + vel_{Tx}^t \Delta t + \frac{1}{2} acc_{Tx}^t \Delta t^2 \\ y_T^t + vel_{Ty}^t \Delta t + \frac{1}{2} acc_{Ty}^t \Delta t^2 \end{bmatrix} \tag{4.38}$$

- Constant acceleration and yaw rate: We also consider a constant acceleration and yaw rate to obtain:

$$\begin{bmatrix} x_T^{t+1} \\ y_T^{t+1} \\ vel_T^{t+1} \\ \theta_T^{t+1} \end{bmatrix} = \begin{bmatrix} x_T^t + vel_T^t cos(\theta_T^t) \Delta t \\ yel_T^t + vel_T^t sin(\theta_T^t) \Delta t \\ vel_T^t + acc_T^{tobs} \Delta t \\ \theta_T^t + \dot{\theta}_T^{tobs} \Delta t \end{bmatrix} \tag{4.39}$$

Multiple-Trajectory Prediction (MTP) [21]: The MTP model uses a CNN over a rasterized representation of the scene and the target vehicle state to generate a fixed number of trajectories (modes) and their associated probabilities (cf. Figure 4.9). It uses a weighted sum of regression (cf. Equation4.18) and classification (cf. Equation4.19) losses for training. We use the implementation of this model by [81].

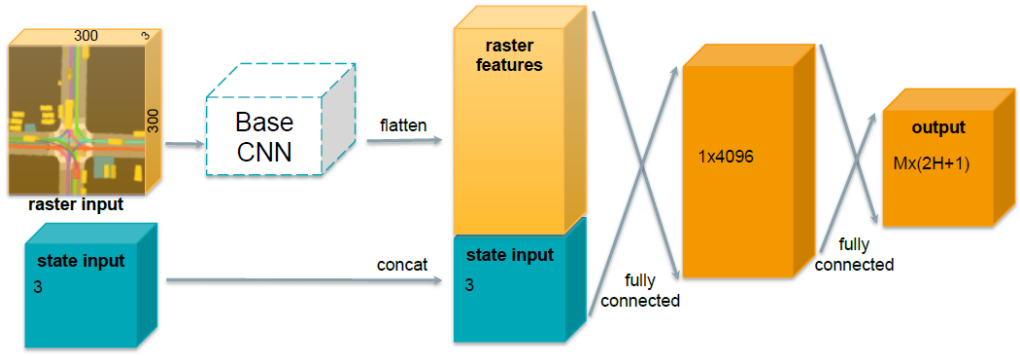


Figure 4.9: MTP network architecture

Multipath [16]: Similar to MTP, the Multipath model uses a CNN with same input. However, unlike MTP, it uses fixed *anchors* obtained from the train set to represent the modes, and outputs residuals with respect to anchors in its regression heads. We implement MultiPath as described in [16].

CoverNet [81]: The CoverNet model formulates multimodal trajectory prediction purely as a classification problem. The model predicts the likelihood of a fixed trajectory set (cf. Figure 4.10), conditioned on the target vehicle state.

Trajectron++ [91]: is a graph-structured recurrent model that predicts the agents trajectories while considering agent motions and heterogeneous scene data.

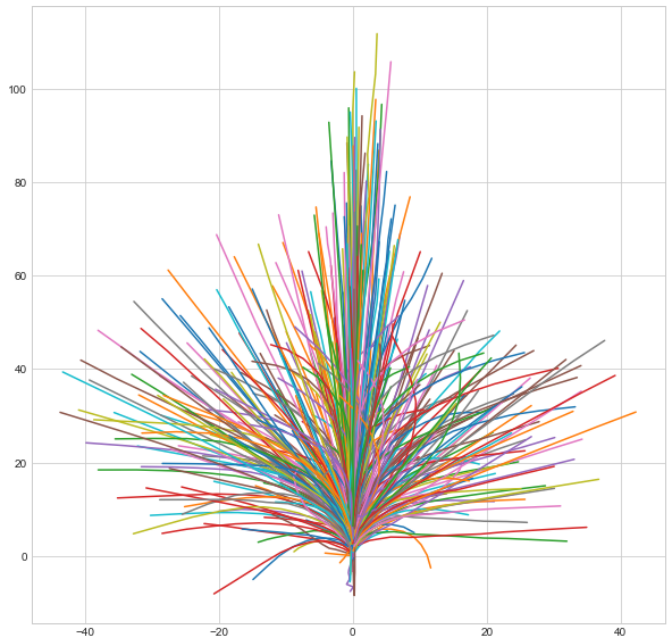


Figure 4.10: CoverNet Trajectory Set

4.4.3.2 Our proposed Methods

MHA-JAM: our basic MHA with joint agent map representation presented in Section 4.1.5.

MHA-JAM-DC our MHA with joint agent map representation and Dynamic Constraints presented in Section 4.1.8.

MHA-JAM-G: MHA-JAM with goal based intention definition presented in Section 4.1.7.1.

MHA-JAM-A: MHA-JAM with anchor based intention definition presented in Section 4.1.7.2.

MHA-JAM-D: MHA-JAM with diversity loss presented in Section 4.1.7.3.

MHA-SAM+JC: MHA with Separate Agent Map plus Joint Conditioning Attention presented in Section 4.1.4.

GA-JAM: our Global Attention (GA) based trajectory prediction with joint agent map representation. It uses only the first attention block of the DA network. In other words, it consists on generating L global features and passing each global feature with the target vehicle motion encoding as input to each LSTM decoder (cf. description in Section 4.2.4).

DA-JAM: our basic DA with joint agent map representation presented in Section 4.2.4.

Table 4.1: Results of comparative analysis on nuScenes dataset, over a prediction horizon of 6-seconds

	MinADE_1	MinADE_5	MinADE_{10}	MinADE_{15}	MinFDE_1	MinFDE_5	MinFDE_{10}	MinFDE_{15}	$\text{MissRate}_{5,2}$	$\text{MissRate}_{10,2}$	Off-Road Rate
Const vel and yaw	4.61	4.61	4.61	4.61	11.21	11.21	11.21	11.21	0.91	0.91	0.14
Physics oracle	**3.69**	3.69	3.69	3.69	9.06	9.06	9.06	9.06	0.88	0.88	0.12
MTP [21]	4.42	2.22	1.74	1.55	10.36	4.83	3.54	3.05	0.74	0.67	0.25
Multipath [16]	4.43	1.78	1.55	1.52	10.16	3.62	2.93	2.89	0.78	0.76	0.36
CoverNet[1] [81]	-	2.62	1.92	-	11.36	-	-	-	0.76	0.64	0.13
Trajectron++[1] [91]	-	1.88	1.51	-	9.52	-	-	-	0.70	0.57	0.25
MHA-JAM	3.77	1.85	1.24	**1.03**	8.65	3.85	2.23	**1.67**	0.60	0.46	0.10
MHA-JAM (off-road)[1]	**3.69**	1.81	1.24	**1.03**	8.57	3.72	2.21	1.70	0.59	**0.45**	**0.07**
MHA-JAM-DC	4.18	1.88	1.38	-	9.49	3.61	2.34	-	0.70	0.65	0.16
MHA-JAM-DC (off-road)	3.92	1.83	1.39	-	8.82	3.47	2.36	-	0.71	0.66	1.12
MHA-SAM+JC	3.70	**1.67**	**1.13**	-	**8.52**	**3.33**	**1.93**	-	**0.57**	0.48	**0.07**
GA-JAM (GF = 10)	3.80	1.71	1.25	-	8.85	3.48	2.30	-	0.61	0.54	0.10
DA-JAM (GF = 16)	3.76	1.77	1.20	-	8.76	3.62	2.13	-	0.60	0.53	0.10
MV-MHA-JAM	4.23	1.94	1.42	-	9.89	4.14	2.77	-	0.69	0.65	0.28
MV-MHDA-JAM (GF = 8)	4.10	1.84	1.32	-	9.53	3.87	2.51	-	0.65	0.58	0.20

MV-MHA-JAM: our basic MHA with joint agent map representation applied for simultaneous multi-vehicle trajectory prediction presented in Section 4.3.

MV-MHDA-JAM: our multi-head DA with joint agent map representation applied for simultaneous multi-vehicle trajectory prediction presented in Section 4.3.
Results of our proposed methods are evaluated when the models are trained without and with off-road loss function. We add the off-road term Method (off-road) to denote that the method is trained with off-road loss.

4.4.4 Quantitative Evaluation

First, we compare our model **MHA-JAM** (MHA with joint agent map representation trained without and with off road loss) to various baselines doing experiments with nuScenes dataset in table 4.1. Our model outperforms all baselines on 9 of the 11 reported metrics, while being second on the remaining two.

For the MinADE_K and MinFDE_K metrics, our model achieves the best results for $K \in \{1, 10, 15\}$ and second best to Multipath [16] when $K = 5$. Having the best performance for $K \in \{10, 15\}$ shows that our method generates a diverse set of plausible trajectories that match the ground truth. However, for $K = 5$, our classifier doesn't seem to succeed in selecting the closest trajectories to the ground truth among the 5 most probable ones, while the Multipath classifier does.

Moreover, our method presents significant improvements compared to others when considering miss rate and off-road rate metrics. Having the lowest miss rate suggests that our predicted trajectories are less likely to deviate from the ground truth over a threshold

[1] Results reported in the nuScenes challenge Leaderboard:
`https://evalai.cloudcv.org/web/challenges/challenge-page/591/leaderboard/1659`

of $d = 2m$. In addition, our model achieves significantly lower off-road rates especially when trained with the off-road loss that penalizes predictions outside of the drivable area. Therefore, it generates scene compliant trajectories.

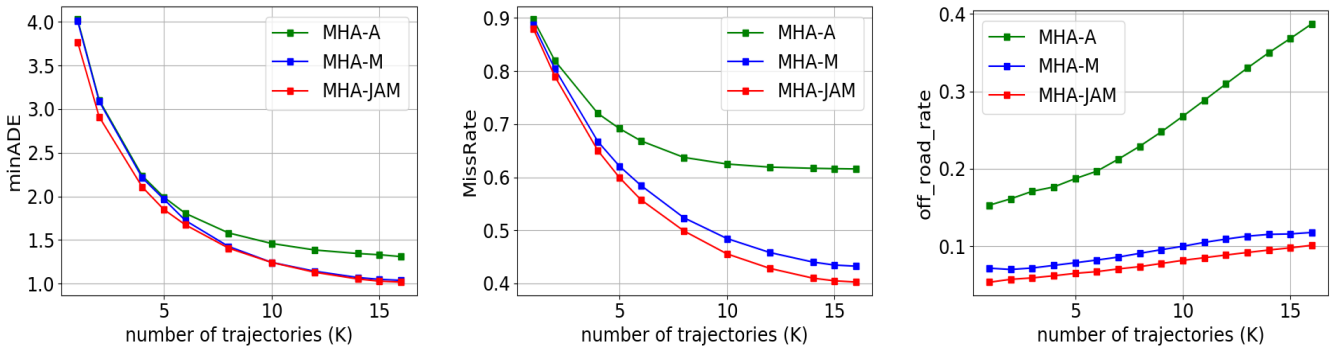
In order to evaluate the performance of our model when applying dynamic constraints **MHA-JAM-DC** (with and without using the off-road loss), we compare it with recent state-of-the-art works and with our basic **MHA-JAM**. While our **MHA-JAM-DC** produces competitive results with the state-of-the-art methods according to most of the metrics, we notice that imposing dynamics constraints on our **MHA-JAM** reduces its performance. Actually, the dynamic unicycle model is a very simplified model of the real motion of vehicles. It considers only the estimated action of the driver on the acceleration and the direction ($i.e.$ the steering wheel) and it omits many physical characteristics of the vehicle like the the moment of inertia, the grip limit of the tires with the road, etc. For specific values of these parameters, the vehicle could make unusual motion like over-steering and skidding. However, the unicycle model does not handle these kind of motion.

We notice that our MHA-SAM+JC performs well in most metrics. However, it has a relatively high complexity compared to the MHA-JAM since it encloses 4 MHA blocks that compute attention through two steps.
We also compare our proposed DA based trajectory prediction approach to the state of the art methods and to our proposed MHA-JAM. We observe that our DA-JAM has competitive results with the MHA-JAM while having a lower complexity (about 30 times). In addition, we conduct an ablation experiment on our DA based method. We only use the first attention block of the DA-JAM that generates global features and we feed each of these features to each of the LSTM decoders to generate different possible trajectories. This method that we denote GA-JAM, has comparable performance to that of DA-JAM. We believe that the second attention block is important to extract features adapted to the target vehicle motion. But, its impact is not clearly shown in the experiments on the nuScenes dataset.

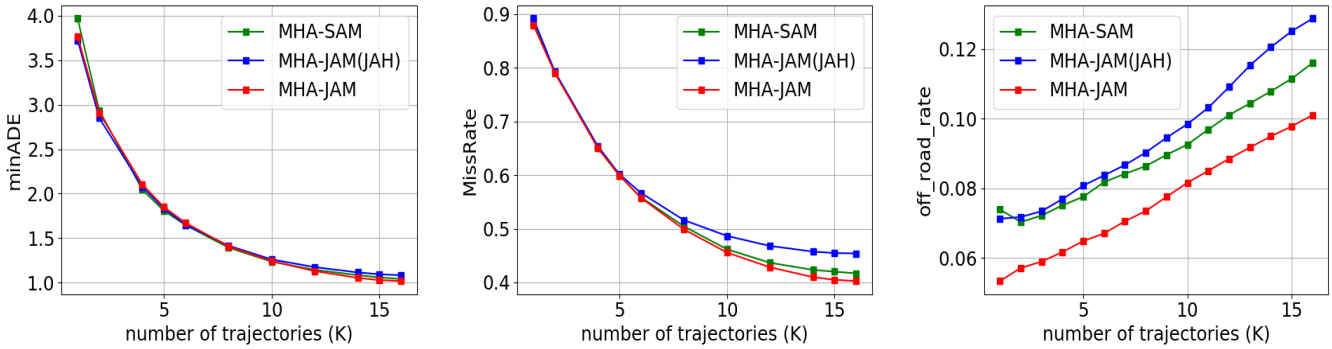
Finally, we consider the task of simultaneous multi-vehicle trajectory prediction. We compare our **MV-MHA-JAM** to our **MV-MHDA-JAM**. We note that our **MV-MHDA-JAM** slightly improves the prediction accuracy according to all the metrics compared to our **MV-MHA-JAM**. This infers that using two steps of attention performs better than applying MHDPA in the case of simultaneous multi-vehicle trajectory prediction.

4.4.5 Ablation Experiments

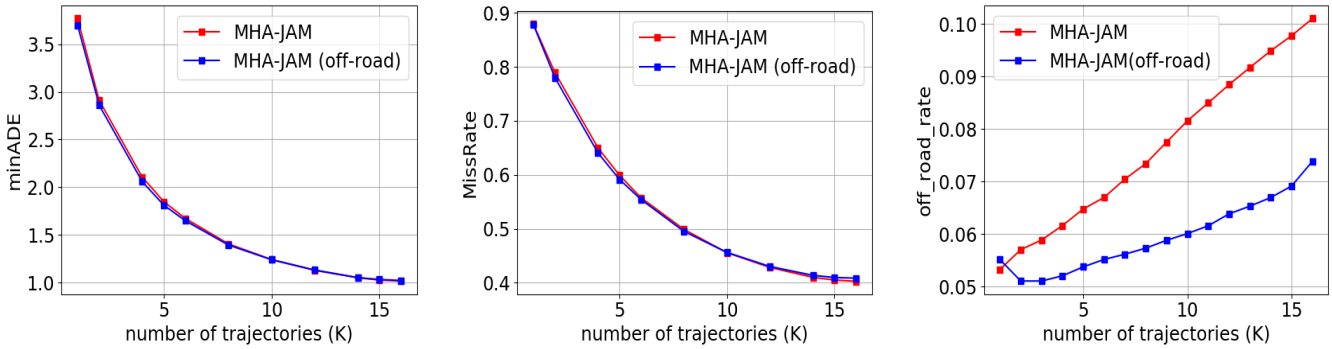
To get a deeper insight on the relative contributions of the various cues and modules affecting the overall performance, we perform the following ablation experiments.



(a) Input cues importance: evaluation metrics for different input features



(b) Context representation importance: comparison of MHA-SAM, MHA-JAM(JAH) and MHA-JAM



(c) Effect of off-road loss

Figure 4.11: Ablation experiments: We evaluate through ablation experiments, the importance of input cues (top), the effectiveness of a joint agent map representation for generating keys and values for attention heads (middle), the effectiveness of attention heads specialized for particular modes of the multimodal predictive distribution (middle), and finally the effectiveness of the auxiliary off-road loss (bottom). For each experiment we plot the metrics MinADE_K (left), $\text{MissRate}_{K,2}$ (middle) and off-road rate (right) for the K likeliest trajectories output by the models.

4.4.5.1 Importance of input cues

Our model relies on two main inputs that help illustrate the interaction of the target vehicle with its context: past motion of surrounding agents and the static scene context represented by high definition maps. To investigate the importance of each input, we compare our model MHA-JAM to two models: (1) MHA with purely agent inputs (MHA-A), and (2) MHA with purely map inputs (MHA-M). When considering only the surrounding agents without any information about the scene structure (MHA-A), the model shows poor results according to the three metrics minADE, missRate$_{K,2}$, and off-road rate (cf. Figure 4.11a). This highlights the importance of the map information to make more accurate and scene compliant predictions. Moreover, since MHA-JAM has the best performance, we infer that considering the surrounding agents also helps our model make a better prediction.

4.4.5.2 Advantage of using multiple attention heads

We train our model with different numbers of attention heads (L) and we compare the minADE$_L$, minFDE$_L$ and MissRate$_{L,2}$. Table 4.2 shows that all the metrics decrease when we increase the number of the attention heads. This proves the usefulness of using different attention heads to generate multimodal predictions.

Table 4.2: MinADE and MinFDE with different numbers of attention heads (L)

L	1	4	8	12	16	20
MinADE	3.48	1.72	1.26	1.13	1.02	1.00
MinFDE	8.01	3.54	2.29	1.91	1.64	1.60
MissRate$_{L,2}$	0.91	0.76	0.59	0.50	0.40	0.40

4.4.5.3 Effectiveness of joint context representation

To prove the effectiveness of using joint context representation, we compare our model to MHA-SAM (MHA with Separate-Agent-Map representation). MHA-SAM is composed of two separate MHA blocks (cf. Figure 4.2): MHA on surrounding agents (similar to MHA-A) and on map (similar to MHA-M). We concatenate their outputs to feed them to the decoders. The main difference between MHA-SAM and MHA-JAM is that MHA-JAM generates keys and values in MHA using a joint representation of the map and agents while MHA-SAM computes keys and values of the map and agents features separately. Figure 4.11b shows that MHA-JAM performs better compared to MHA-SAM especially according to the off-road rate metric. This proves the benefit of applying attention on a joint spatio-temporal context representation composed of map and surrounding agents motion, over using separate attention blocks to model vehicle-map and vehicle-agents interaction independently.

4.4.5.4 Effectiveness of a specialized attention heads:

We also compare our method to MHA-JAM (with Joint-Attention-Heads JAH). While MHA-JAM uses each attention head $head_l$ to generate a possible trajectory, MHA-JAM with joint attention heads uses a fully connected layer to combine the outputs of all attention heads $head_l,\ l = 1 \dots L$. It generates each possible trajectory using a learnt combination of all the attention heads.

Comparing MHA-JAM and MHA-JAM (JAH) reveals that conditioning each possible trajectory on a context generated by one attention head performs better than generating each trajectory based on a combination of all attention heads.

4.4.5.5 Role of the Off-road loss:

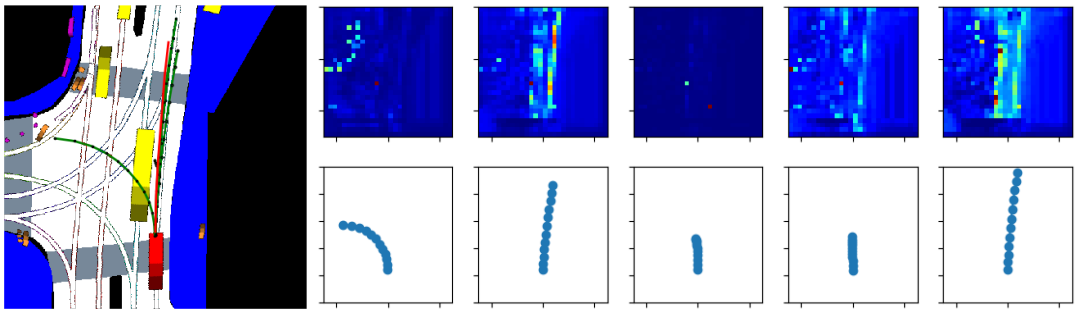
Figure 4.11c compares MHA-JAM trained with and without off-road loss (cf. Section 4.1.9). We notice that the off-road loss helps generating trajectories more compliant to the scene by reducing the off-road rate while maintaining good prediction precision.

4.4.6 Qualitative Evaluation

4.4.6.1 MHA-JAM Method

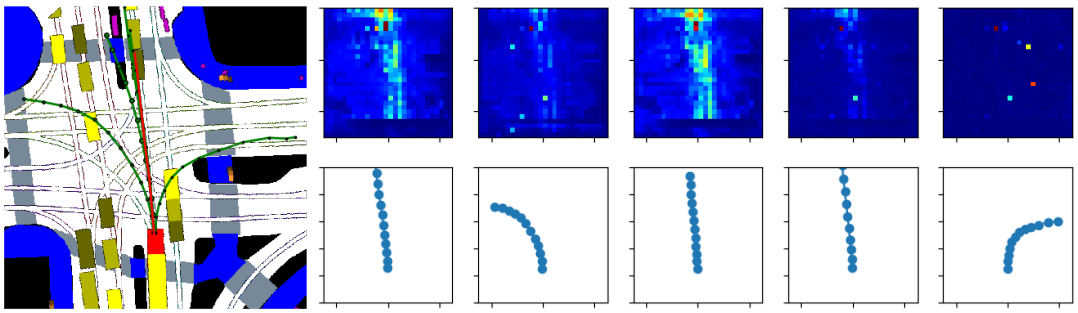
Figure 4.12 presents two examples of vehicle trajectory prediction, their corresponding 5 most probable generated trajectories and their associated attention maps. We notice that our proposed model MHA-JAM (off-road) successfully predicts diverse possible maneuvers; going straight on and turning left for the first Example 4.12a and going straight on, turning left and right for the second Example 4.12b. In addition, it produces different attention maps which implies that it learnt to create specific context features for each predicted trajectories. For instance, the attention maps of the going straight on trajectories, assign high weights to the drivable area in the straight direction and to the leading vehicles (the dark red cells). Moreover, They show focus on relatively close features when performed with low speed and further ones with high speed (cf. Example 4.12a). For the left and right turns, in both examples, the corresponding attention maps seem to assign high weights to surrounding agents that could interact with the target vehicle while performing those maneuvers. For instance, in the left turn (cf. Example 4.12b), the attention map assigns high weights to vehicles in the opposite lane turning right. For the left turn of the first example and for the right turn of the second example, the attention maps assign high weights to pedestrians standing on both sides of the crosswalks. However, for the right turn, the model fails to take into account the traffic direction.

Figure 4.13 shows the average attention maps, for 4 generated possible maneuvers (going straight with low and high speed, left and right), over all samples in the test set.



(i) Predicted trajectories **(ii)** 5 most probable trajectories and their corresponding attention maps

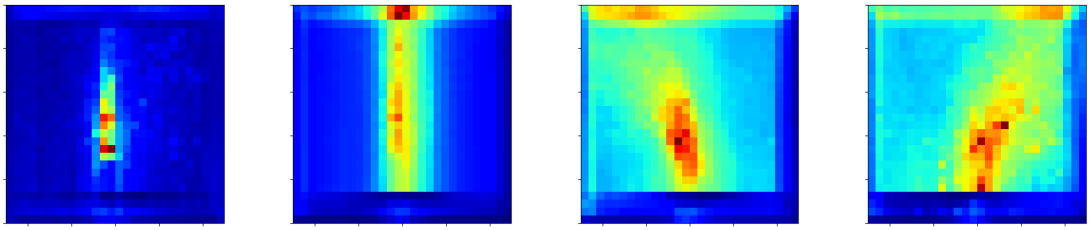
(a) Example 1



(i) Predicted trajectories **(ii)** 5 most probable trajectories and their corresponding attention maps

(b) Example 2

Figure 4.12: Examples of produced attention maps and trajectories with MHA-JAM (off-road) model



(a) Mean attention maps going straight (low speed)

(b) Mean attention maps going straight (high speed)

(c) Mean attention maps going left

(d) Mean attention maps going right

Figure 4.13: Visualisation of average attention maps over different generated maneuvers.

We note that each attention map assigns high weights, on average, to the leading vehicles, to surrounding agents and to the map cells in the direction of the performed maneuvers. This consolidates the previous observations in Figure 4.12. We conclude that our model generates interpretable attention maps that focus on specific surrounding agents and scene features depending on the future possible trajectory.

Table 4.3: Results of comparative analysis on nuScenes dataset, over a prediction horizon of 6-seconds

	MinADE_1	MinADE_5	MinADE_{10}	MinADE_{15}	MinFDE_1	MinFDE_5	MinFDE_{10}	MinFDE_{15}	$\text{MissRate}_{5,2}$	$\text{MissRate}_{10,2}$	Off-Road Rate
MHA-JAM	3.77	1.85	**1.24**	**1.03**	8.65	3.85	2.23	**1.67**	0.60	0.46	0.10
MHA-JAM (off-road)[1]	**3.69**	1.81	1.24	**1.03**	**8.57**	3.72	**2.21**	1.70	**0.59**	**0.45**	**0.07**
MHA-JAM-G	3.91	1.69	1.51	1.49	9.08	**3.45**	2.98	2.92	0.77	0.76	0.44
MHA-JAM-G (off-road)	4.20	1.75	1.33	1.21	9.41	3.67	2.55	2.24	0.69	0.57	0.08
MHA-JAM-A	3.94	1.69	1.55	1.53	9.06	3.48	3.04	2.99	0.79	0.78	0.40
MHA-JAM-A (off-road)	4.22	**1.68**	1.46	1.41	9.64	3.46	2.83	2.65	0.74	0.71	0.09
MHA-JAM-D	3.87	1.95	1.32	1.12	8.82	4.03	2.35	1.78	0.63	0.52	0.18
MHA-JAM-D (off-road)	3.87	1.94	1.34	1.13	8.86	4.04	2.40	1.82	0.63	0.50	0.10

4.4.6.2 Intention-based Prediction

In this subsection, we analyse the results obtained using intention based trajectory prediction. We notice that results using different methods are competitive depending on the deployed metric (cf. Table 4.3). In contrast to the basic MHA-JAM, results with intention based methods have relatively high off-road rate. This infers that when defining intentions, the model learning is more focused on the specific patterns characterizing each intention rather than on the map information. The use of off-road loss reduces the off-road rate significantly.

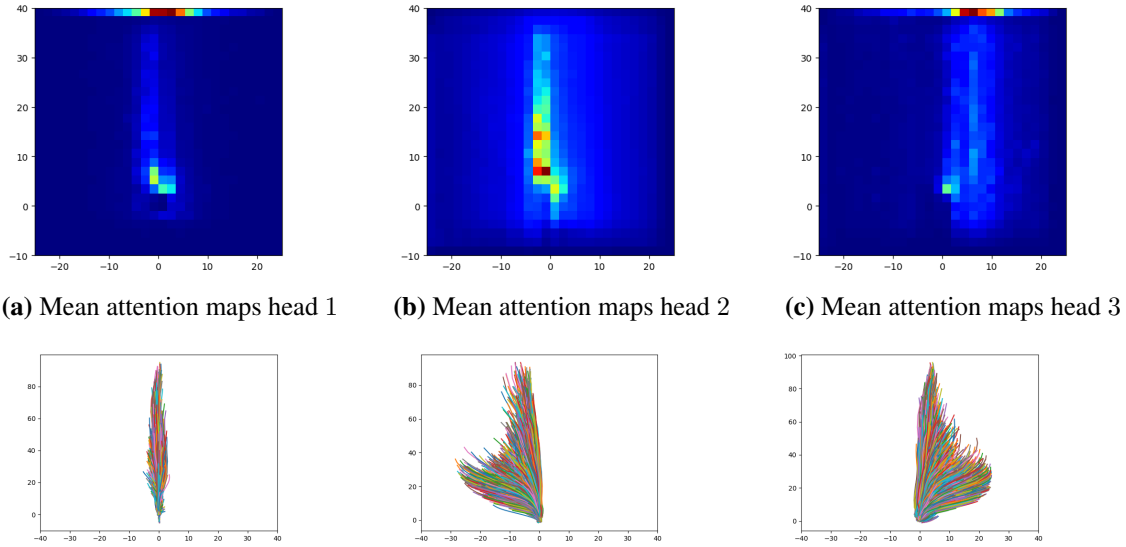
To further analyze the effect of the intention definition, we perform experiments using three attention heads (cf. Table 4.4) and we visualise the attention maps when using each intention definition method.

Table 4.4: Results of comparative analysis of intension based methods on nuScenes dataset, over a prediction horizon of 6-seconds using 3 prediction modes

	MinADE_1	MinADE_3	MinFDE_1	MinFDE_3	$\text{MissRate}3,2$	Off-Road Rate	Diversity Score
MHA-JAM	3.71	**1.95**	8.53	**4.15**	0.82	0.09	0.170
MHA-JAM-G	**3.48**	2.77	**8.01**	6.18	0.89	0.28	0.121
MHA-JAM-MA	4.02	2.04	9.12	4.27	0.84	0.10	**0.000**
MHA-JAM-D	3.92	2.01	9.00	**4.15**	**0.81**	**0.08**	0.004

Comparing the results in Table 4.1 and Table 4.4, we notice that the performances and the ranking of the different methods depend on the number of prediction modes considered.

The results with three prediction modes show competitive performance when using the different intention definition techniques. Our goal directed MHA-JAM-G has the best MinADE_1 and MinFDE_1. Therefore, it has the best classifier that enables it to infer the trajectory that fits the ground truth the best among the generated trajectories. However, it has the worst off-road rate. This rate can be reduced using the off-loss. Our basic MHA-JAM performs the best considering the metrics MinADE_3 and MinFDE_3. This infers that, using our basic MHA-JAM, we are more likely to generate a prediction



(a) Mean attention maps head 1 **(b)** Mean attention maps head 2 **(c)** Mean attention maps head 3

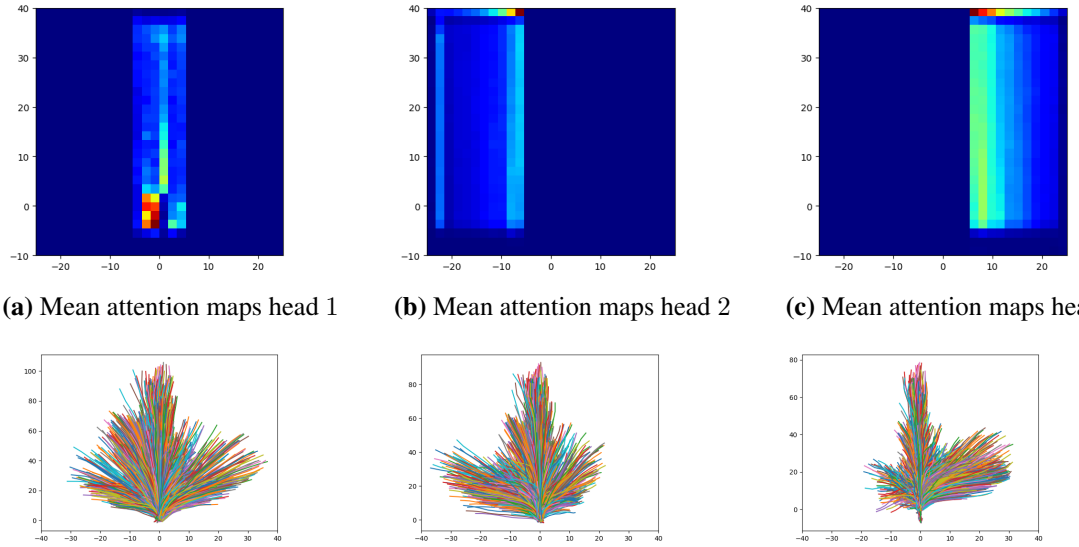
(d) Predicted Trajectories head 1 **(e)** Predicted Trajectories head 2 **(f)** Predicted Trajectories head 3

Figure 4.14: Visualisation of average attention maps and predicted trajectories over different generated maneuvers

(among the three generated trajectories) that fits the best the ground truth according to the metrics ADE and FDE.

The first experiment consists in defining three basic destinations corresponding to continuing straight, turning right and turning left maneuvers. Figure 4.14 shows the mean of each of the attention maps of the trajectories of the test set generated by each attention head. It also presents the generated trajectories when using each attention head. We notice that the model learns to generate trajectories with different intentions where each intention corresponds to a specific destination. Moreover, the network learns to attribute higher weights to the features in the direction of each specified destination. This method guaranties to have diverse predictions. However, the main limitation of this method is that the generated trajectories are more likely to be outside of the drivable area in some situations. For example, if the road presents a deviation to the left, while the two first attention heads can generate plausible trajectories, the last one will fail to adapt to the road structure since it always generate trajectories with destinations on the right. This justifies this method having the highest off-road rate (cf. Table 4.4).

The second intention based method consists in forcing the attention weights to focus on specific regions defined prior to the training. To do so, we divide the map of features into three regions of interest (left, middle and right) and we associate each region to one of the attention heads. Then, we force the attention weights of each attention head to be equal to zero outside its corresponding region of interest (cf. Figure 4.15). We notice that the network generates trajectories in different directions with a focus on the



(a) Mean attention maps head 1 **(b)** Mean attention maps head 2 **(c)** Mean attention maps head 3

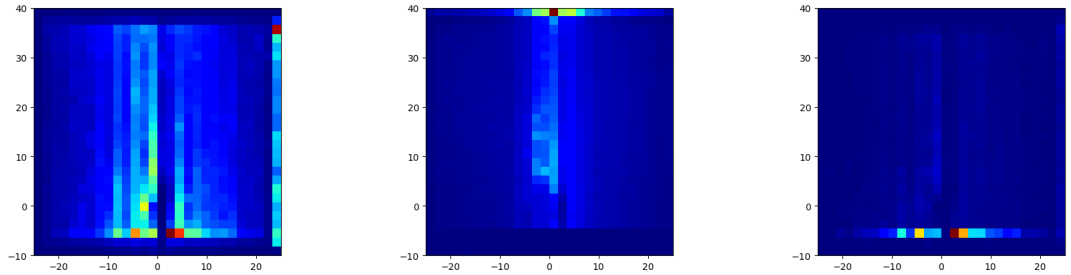
(d) Predicted Trajectories head 1 **(e)** Predicted Trajectories head 2 **(f)** Predicted Trajectories head 3

Figure 4.15: Visualisation of average attention maps and predicted trajectories when considering different regions of attention

direction of the considered region. For example, the third attention head corresponding to the right region generates trajectories consisting mainly of going straight and going right maneuvers. It also predicts going left but only with low speed since the features of the left side of the scene are masked. We also see that adjusting the attention weights influences the generated trajectories.

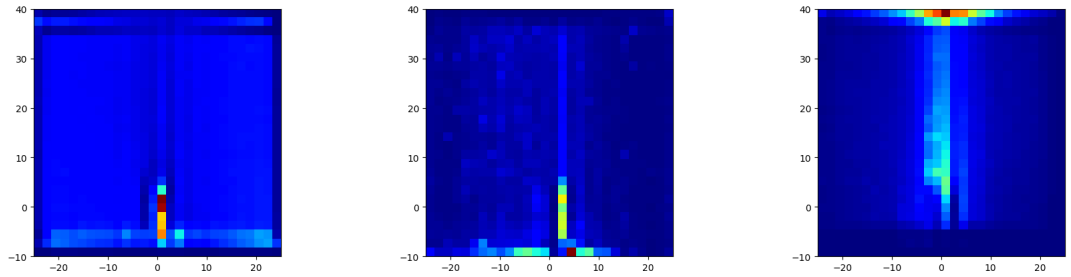
Knowing the influence of the attention weights on the generated trajectories, we further put constraints on the attention weights to enhance the diversity of the produced trajectories using the diversity loss previously defined. Then, we compare the attention heads without using the diversity loss (cf. Figure 4.16) and after using it (cf. Figure 4.17). We notice that, even without using the diversity loss the attention maps look different and each one has high weights in different regions than the others. The use of the diversity loss enhances the difference between the heads and forces each attention head to focus on a specific region on the feature map different than the other attention heads. The diversity score validates our observations concerning the diversity of the attention maps.

Figure 4.18 shows an example of trajectory prediction near an intersection using four variants of **MHA-JAM**. We can see that the predicted trajectories are diverse enough using all the four method. This diversity is exhibited through the generation of different maneuvers (different directions of motion and different speed profiles are adopted in each prediction mode).



(a) Mean attention maps head 1 **(b)** Mean attention maps head 2 **(c)** Mean attention maps head 3

Figure 4.16: Visualisation of average attention maps of our basic MHA-JAM method

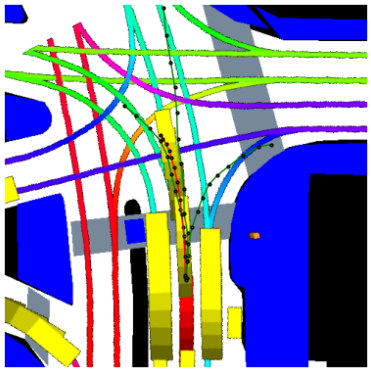


(a) Mean attention maps head 1 **(b)** Mean attention maps head 2 **(c)** Mean attention maps head 3

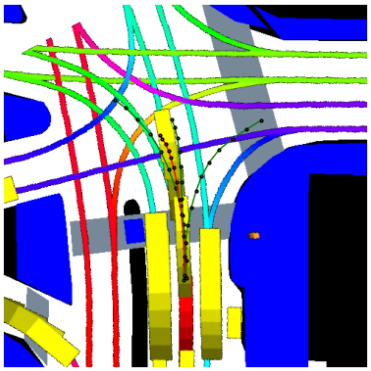
Figure 4.17: Visualisation of average attention maps when using the diversity loss

4.5 Conclusion

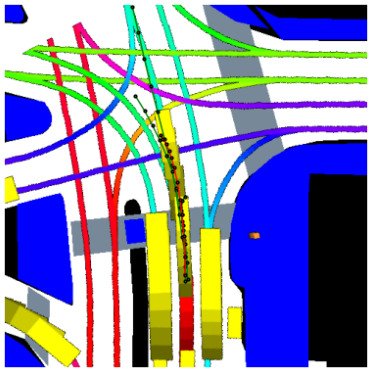
In this chapter, we tackled the task of vehicle trajectory prediction in an urban environment while considering interactions between the target vehicle, its surrounding agents and the scene. To this end, we deployed a MHA-based method on a joint agents and map global context representation. The model enabled each attention head to explicitly extract specific agents and scene features that help to infer the driver's diverse possible behaviors. Furthermore, the visualization of the attention maps reveals the importance of joint agents and map features and the interactions occurring during the execution of each possible trajectory. It confirmed that the attention is concentrated in the direction of the future maneuver. In addition, we augmented our MHA-JAM with different methods of attributing an intention to each prediction mode in order to enhance the diversity of the predicted trajectories. We also proposed another attention based approach with lower complexity and comparable performance. Experiments showed that our proposed approaches outperform the existing methods according to most of the metrics considered, especially the off-road metric. This highlights that the predicted trajectories comply with the scene structure.



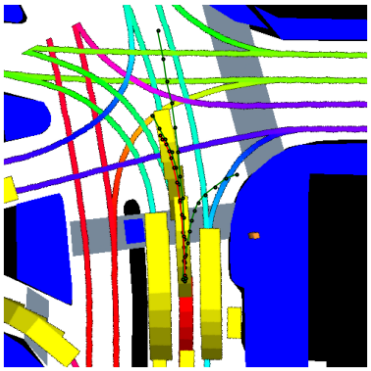
(a) Predicted trajectories using **MHA-JAM**



(b) Predicted trajectories using **MHA-JAM-G**



(c) Predicted trajectories using **MHA-JAM-MA**



(d) Predicted trajectories using **MHA-JAM-D**

Figure 4.18: Visualisation of an example of trajectory prediction near an intersection using different MHA-JAM based methods with three attention heads and a prediction horizon of three seconds. The target vehicle presented in red and its surrounding agents in yellow. The three predicted trajectories are green and the ground truth one is red.

5

Scene Aware Trajectory Prediction in an Urban Environment: A Non Recurrent Approach

Contents

5.1 Transformers and LSTMs for Trajectory Prediction 112
5.2 Multi-Modal Transformer for Trajectory Prediction 113
5.2.1 Model Inputs 113
5.2.2 Model Output 114
5.2.3 Positional Encoding 115
5.2.4 Encoding Layer 116
5.2.5 Decoding Layer 117
5.2.6 Transformer Refinement 120
5.2.7 Classification Layer 122
5.2.8 Multimodal Network Architecture 122
5.2.9 Loss Functions 123
5.3 Experimental Analysis and Evaluations 123
5.3.1 Training and Implementation Details 123
5.3.2 Models Compared 124
5.3.3 Quantitative Evaluation 124
5.3.4 Ablation Experiments 124
5.3.5 Qualitative Evaluation 125
5.4 Conclusion 127

In this chapter, we propose a novel way of generating multimodal prediction using Transformer-based architecture to perform trajectory prediction in an urban environment. This approach is a non-autoregressive alternative to our methods deployed in the previous chapter. It considers predefined anchor trajectories as generic intentions, and then refines and adapts them to the context.

The Tranformer has shown a significant performance in sequence learning and generation tasks [26, 107]. Recent studies deployed it in the task of trajectory prediction [32, 116]. However, they do not use information about the scene structure. While Giuliari et al. [32] make trajectory predictions independently of the context, STAR (Spatio-Temporal grAph tRansformer framework) [116] models the spatio-temporal dependencies between agents. To this end, it deploys two types of TFs; a spatial TF that focuses on agents' interactions and a temporal TF that models the temporal evolution of the motion. This framework interleaves the spatial and temporal TFs in two encoder blocks. They mount the two types of TFs in parallel to form the first encoder and sequentially for the second encoder. We consider this implementation comprising 4 TFs computationally expensive. Therefore, we opt for an architecture based on the regular TF encoder-decoder and we adapt it to integrate scene information.

The attention mechanism is widely used with entries from different modalities (text, image, etc.). Libovicky et al. [66] add layers to the decoder in order to take into account for information from different sources. Inspired by their work, we augment the regular TF decoder module with an additional layer that we denote decoder map attention to integrate map information.
Our main contributions are:

- Augmenting the TF architecture to simultaneously model the spatial and temporal information.
- Generating probabilistic multimodal intention-based prediction using anchors defined prior to the training.
- Applying a refinement approach to further model the dependencies between the output sequence elements.

5.1 Transformers and LSTMs for Trajectory Prediction

An agent's motion is a continuous temporal evolution of its states. Each state depends on the previous ones and influences the future ones. This dependency needs to be integrated in modeling surrounding agents' behaviors.

An LSTM is the primary method for modeling temporal evolution in trajectory prediction. However, its structure is unable to simultaneously learn spatial and temporal dependencies. Therefore, in the methods we previously proposed, we augment the LSTM encoder and decoder with MHA to model the interactions between the target vehicle, its surrounding agents and the static scene elements and we deduce that attention performs well in the task of trajectory prediction.

MHA was introduced [107] as a stand-alone building block of the Transformer (TF) network that replaces the entire LSTM encoder-decoder. It has shown significant performance in sequence learning and generation tasks [26, 107]. Actually, the TF addresses some of the limitations of LSTM. First, the LSTM processes the input data sequentially by generating a sequence of hidden states, as a function of the previous hidden state and the current input. It has to go through every input state one after another to generate the encoding vector. This precludes parallelization in training. As a remedy, the TF makes predictions by attending to input data in parallel and does not require any sequential computation, which reduces the training time. In addition, the LSTM learns dependencies between inputs sequentially which can result in loss of information between distant elements, especially when dealing with relatively long sequences. In contrast, the TF directly relates each element from the input sequence to the others and to the decoder. This makes us question the use of the LSTM models and replace the LSTM encoder and decoder with a TF to model temporal dependencies.

5.2 Multi-Modal Transformer for Trajectory Prediction

In the following, we describe our proposed spatio-temporal multimodal Transformer-based architecture (**ST-M-TF**). First, we presents the inputs and the outputs of our model. Then, we detail its building blocks.

5.2.1 Model Inputs

Similar to the MHA-JAM method, we use information about the target and its surrounding agents' trajectories and the scene map as input for our TF-based model. We also use the same definition of the interaction space of a target vehicle T. However, we use a slightly different representation of these inputs:

Trajectory representation: Similar to MHA-JAM, we present the target agent's trajectory as a sequence of its states, for t_h past time steps between $t_{pred} - t_h$ and t_{pred},

$$S_T = [S_T^{t_{pred}-t_h}, \ldots, S_T^{t_{pred}}]. \tag{5.1}$$

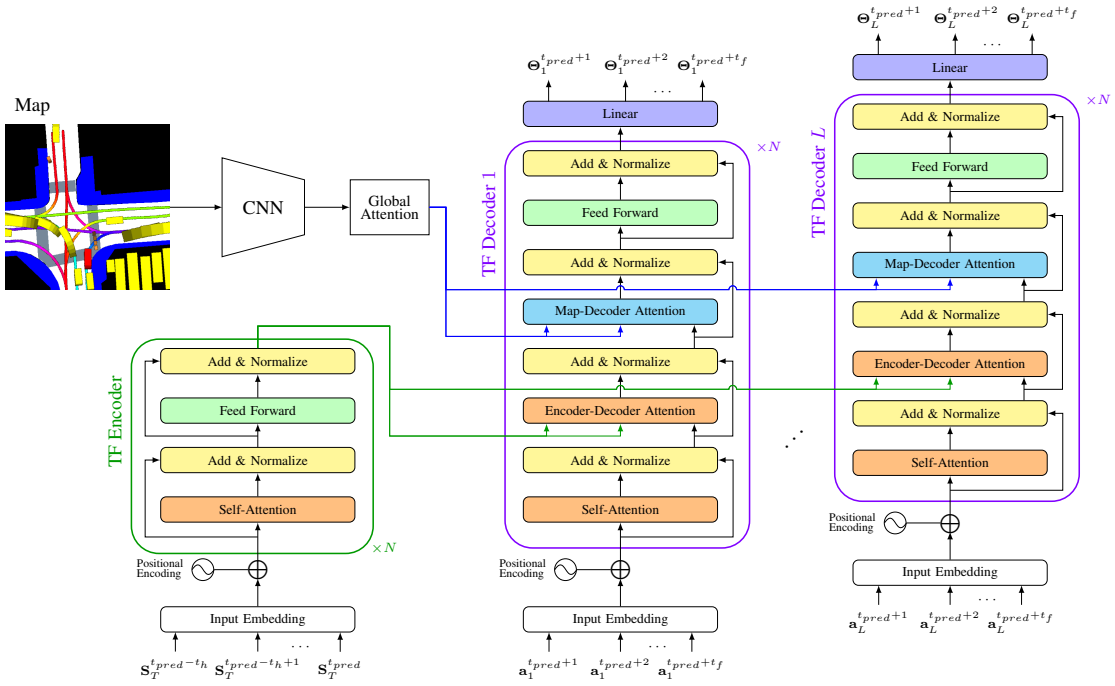


Figure 5.1: Spatio-Temporal Multimodal Transformer **SF-M-TF**

Each state is composed of a sequence of the target agent's relative coordinates x_T^t and y_T^t, velocity v_T^t, acceleration a_T^t and yaw rate $\dot{\theta}_T^t$,

$$S_T^t = (x_T^t, y_T^t, v_T^t, a_T^t, \dot{\theta}_T^t),\ t \in t_{pred} - t_h, \ldots, t_{pred}. \tag{5.2}$$

The trajectories of the surrounding agents present in the interaction area are presented as a sequence of oriented bounding boxes drawn on the input map representation image. Each sequence characterizes the positions and orientations of an agent for the past timesteps.

Map representation: The road geometry, driving area and lane divisions of the interaction space map as well as the surrounding vehicles representations are fed as a rasterized [21] RGB image $\mathcal{M}$ to the model.

5.2.2 Model Output

Given the past motion sequence $S_T = [S_T^1, ..., S_T^{t_{pred}}]$ and the context map $\mathcal{M}$, we aim to estimate the probability distribution $P(Y|S, \mathcal{M})$ over the future locations Y of the target vehicle. To account for multimodality of $P(Y|S, \mathcal{M})$, we model it as a mixture distribution with L mixture components. Each mixture component consists of predicted location co-ordinates at discrete time-steps over a prediction horizon t_f.

$$Y_l = [Y_l^{t_{pred}+1}, \ldots, Y_l^{t_{pred}+t_f}],\ l = 1, \ldots, L. \tag{5.3}$$

A regular auto-regressive model (such as LSTMs and TFs) generates the distribution over the output positions into a sequence of conditional probabilities with a left-to-right causal structure:

$$P(Y_l|S,M) = \prod_{t=t_{pred}+1}^{t_{pred}+t_f} p(Y_l^t|Y_l^{t_{pred}+1:t-1}, S, M) \tag{5.4}$$

We use a fixed set of anchor trajectories extracted prior to the training. These trajectories represent different ways of dependencies among the future time steps. In our approach, we assume that the time-step distributions are conditionally independent given a set of fixed anchor trajectories. Therefore, we write $p(Y_l^t|S,M)$ instead of $p(Y_l^t|Y_l^{t_{pred}+1:t-1}, S, M)$. This assumption enables us to generate predictions for all the future time steps in a single inference pass. This increases the efficiency of our model by reducing the inference time.

5.2.3 Positional Encoding

The LSTM receives and treats the input states sequentially. As the order of the input sequence contains the information about the time of each state, the LSTM is able to model the temporal evolution of the sequence. However, a TF processes the input positions in parallel. Therefore, additional information about the time of each state is required for the TF to be able to model the temporal evolution of the motion efficiently. The TF uses "positional encoding" to retain the positional information of the input sequence and integrate information about time in the embedding of each past and future input sequence state.

The state vector S_T^t of the target agent is embedded using a fully connected layer to a vector e_T^t. A positional encoding vector p^t is added to the embeddings, of the same dimensionality $D = d_{enc}$: $\xi_T^t = e_T^t + p^t$

$$p^t = \{p_{t,d}\}_{d=1}^{D}, \quad \text{where } p_{t,d} = \begin{cases} sin(\frac{t}{10000^{\frac{d}{D}}}) \\ cos(\frac{t}{10000^{\frac{d}{D}}}) \end{cases} \tag{5.5}$$

The sinusoid functions define the order of each dimension of the positional encoding vector using a specific frequency that ensures a unique time stamp for each element of the sequence considered. Therefore, for every input state, we have information about its respective order and time. The same encoding is used to encode the anchor vectors, as we detail in the following.

5.2.4 Encoding Layer

The state vectors of the target vehicle and the rasterized representation of the map covering the interaction area and describing the surrounding agents are fed as input features to the encoding layer. The latter is composed of two independent modules: the first models the temporal information using the target agent's history and the second extracts spatial main features from the rasterized representation of the map. It is composed of two modules:

The trajectory encoding module: The embedding vectors with the positional encoding ξ_T^t of the target agent trajectory are encoded using a TF encoder. The latter is composed of a stack of N_l layers, each one formed of three building blocks:

- A multi-head attention module
- A feed-forward fully-connected module
- A residual connection after each of the previous blocks

The TF encoder models the temporal evolution of the target agent's motion. It focuses on the attention over the past time sequence of states. The embedding vectors of the input states are projected to form queries, keys and values. The multi-head self-attention block enables each vector presenting an input state to perform a relational reasoning with the other vectors using each attention head in order to capture the temporal dependencies in the input sequence. The sequence of information gathered by the attention block and presenting each position is passed to a feed forward layer. The latter is applied to each position separately. A residual connection is used after each of the previous blocks. It helps preserve the positional and spatio-temporal information from the input sequences. Then, each position in the encoder subsequent MHA layers attends to all positions in the previous layer. Finally, the TF encoder generates a high level representation of the motion dynamics of the target vehicle in the form of a sequence of vectors. This sequence is projected to form the keys K_{enc} and values V_{enc} and passed on to the TF encoder-decoder attention block.
We note that in contrast to the LSTM, which generates an encoding vector, the TF encoder maps the input sequence to a higher representation sequence of equal length.

The scene feature extractor module: Similar to the MHA-JAM, we use a CNN to extract high level features from map. The CNN transforms the input image to a 3D tensor F_m of size (M, N, P_m), where (M, N) presents the size of the 2D spatial grid considered and P_m the number of features.
CNNs, composed mainly of convolution operators, extract salient features by operating on spatial and channel-wise elements within local receptive fields at each layer. Thus, a CNN misses global information. We aim to strengthen the representational power of the CNN and enhance its spatial encodings by directly extracting the most informative features regardless of their locations using a Global Attention (GA) mechanism. GA models spatial relationships in a computationally efficient manner. The 3D tensor F_m presenting the joint context features is projected into two different spaces to form two context representations C and E. Actually, C and E are the outputs of the two different convolution layers (kernel size $= (1, 1)$) applied to the context features F_m.

$$C = \theta(F_m; W_\theta)^T,\ C = [\mathbf{c}_1, \ldots, \mathbf{c}_{MN}] \in \mathbb{R}^{d_{GF} \times MN} \tag{5.6}$$

$$E = \phi(F_m; W_\phi)^T,\ E = [\mathbf{e}_1, \ldots, \mathbf{e}_{MN}] \in \mathbb{R}^{N_{GF} \times MN}, \tag{5.7}$$

where d_{GF} and N_{GF} are the dimension and the number of the global features.
We rewrite the feature matrix E as:

$$E = [\bar{\mathbf{e}}_1, \ldots, \bar{\mathbf{e}}_{N_{GF}}], \tag{5.8}$$

where $\bar{\mathbf{e}}_i \in \mathbb{R}^{MN}$ is an MN-dimensional row vector of the matrix E.
The global features $G = \left[\mathbf{g}_1, \ldots, \mathbf{g}_{N_{GF}}\right]$ are computed as:

$$\mathbf{g}_i = C\ softmax(\bar{\mathbf{e}}_i)^T \tag{5.9}$$

The term $softmax(\bar{\mathbf{e}}_i)$ presents the i^{th} global attention map composed of attention weights extracted from the context features F_m independently of the input sequence of states.
The scene feature extractor module generates N_{GF} salient features where N_{GF}/L is the number of global features passed on to each TF decoder.

5.2.5 Decoding Layer

A regular TF decoder is composed of four building blocks:

- A self-attention module where each position in the decoder attends to the preceding positions.

- An encoder-decoder MHA module that receives the queries from the previous decoder layer, and the keys and values from the encoder. Hence, every position in the decoder attends to the high level representation sequence of the input generated by the encoder.
- A feed-forward fully-connected module
- A residual connection after each of the previous blocks.

During the training, since we have the ground truth trajectories, the target sequence is fed as input to the decoder in order to learn the dependencies between the output sequence states. During the inference, the decoder performs an auto-regressive generation of the predicted sequence; it generates a predicted position for the first iteration. This prediction is fed to the decoder as input so that the next prediction attends to it (cf. Figure 5.2). Then, it proceeds by generating each prediction by conditioning it on the previously generated outputs till the end of the sequence. The output sequence generation requires t_f iterations.

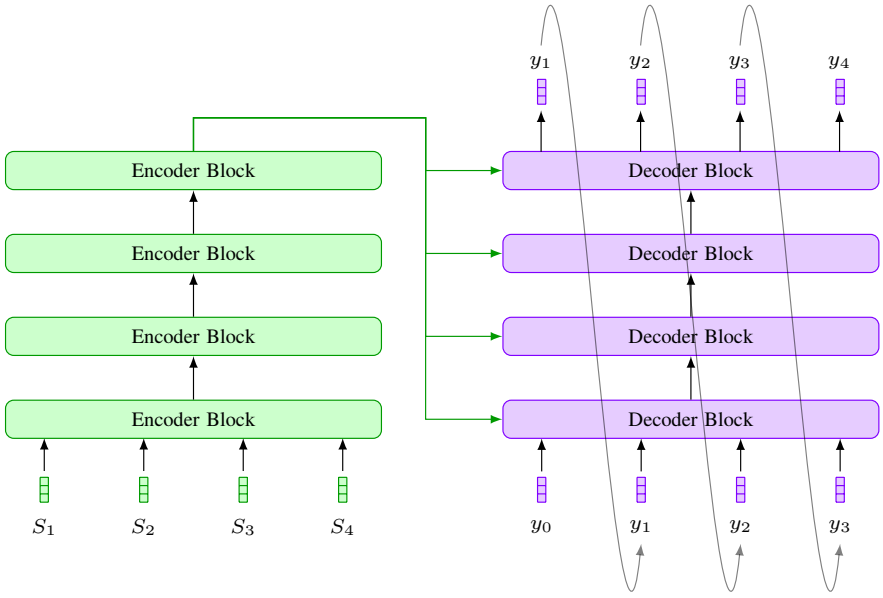


Figure 5.2: Regular Auto-regressive Transformer

We adapt the decoder architecture to meet the trajectory prediction requirements as follows:

5.2.5.1 Non Auto-regressive Multimodal Trajectory Generation

The auto-regressive property of TFs induces an important inference latency since an auto-regressive model generates one predicted position at each iteration. As a remedy, we propose to use a non-auto-regressive TF solution that generates all positions in one

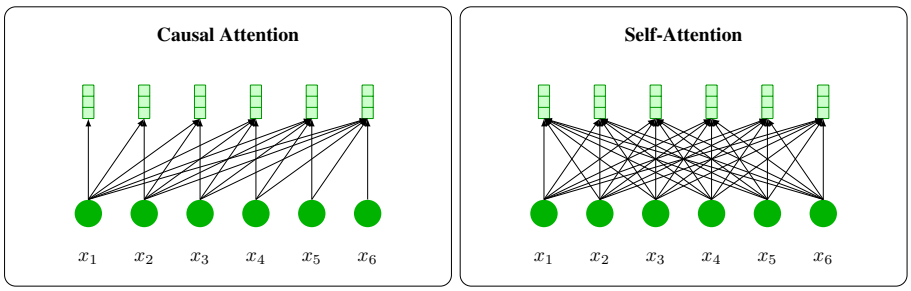
pass in the task of trajectory prediction. This increases the computational efficiency by enabling parallel processing. This was inspired by the recent study on non-autoregressive neural sequence modeling by Gu et al. [34] in the context of machine translation. In addition, in order to consider the future uncertainty, we augment our Spatio-temporal TF to an intention based multimodal trajectory predictor. Indeed, instead of feeding the ground truth trajectories as input to the TF decoder as in the regular TF, we input to the decoder a trajectory that describes a specific agent's intention defined prior to the training. In addition, we do not use a single intention, we define multiple trajectories defining different intentions denoted as follows:

$$A_l = [\mathbf{a}_l^{t_{pred}+1}, \ldots, \mathbf{a}_l^{t_{pred}+t_f}], \quad l = 1 \ldots L \tag{5.10}$$

We also deploy L decoders. Each one generates a trajectory corresponding to a specific intention. Deploying multiple intentions enables the network to learn the different aspects of dependency between the output sequence states.

5.2.5.2 Full Decoder Self-Attention

The regular TF decoder self-attention layer uses a causal self-attention mask that forces each position to attend to only the previous positions. In our approach, we remove that mask in order to enable all positions to attend to all other positions so that each generated prediction is conditioned on the full information about the global future intention (cf. Figure 5.3).



(a) Causal self-attention **(b)** Full self-attention

Figure 5.3: Causal vs Full self-attention: The inputs sequence to the attention is marked as circles, the outputs of the attention blocks are presented by vectors and the arrows present the dependencies.

5.2.5.3 Map-Decoder MHA Layer

In order to integrate information about the scene structure in the trajectory predictor, we augment the regular TF decoder with an additional layer that we insert after the encoder-decoder MHA module and we denote it map-decoder MHA. This layer enables the resulting sequence from the previous decoder layer to attend to the map features in order to extract information about the scene structure and therefore possible paths. The GA extracts the global features of the map independently of the target vehicle's trajectory and intention. The map-decoder layer has as inputs the encoder-decoder layer output and the global context features. It enables the network to attend to specific context features based on the target vehicle's past motion and its global intention.

5.2.6 Transformer Refinement

Non-autoregressive models generate all of the output sequences in one pass. However, they attempt to directly model the joint distribution of all the sequence elements simultaneously without taking into consideration the dependencies of the decoding history during generation. In order to further investigate modeling the dependencies among the future trajectory states, we augment our non-autoregressive TF previously described with an iterative refinement strategy [58].

5.2.6.1 Latent variable model

The main idea behind the refinement based TF model [58] is to exploit latent variables to model the dependencies among the target sequence elements using an iterative inference strategy. It consists in introducing latent variables and using them as a process of iterative refinement without auto-regression. It infers the dependencies among target trajectory states given the history and the map features by deploying R intermediate random variables $I_l^0, \ldots, I_l^R$ and marginalizing them out:

$$P(Y_l|S,\mathcal{M}) = \sum_{I_l^0,\ldots,I_l^R} \left(\prod_{t=t_{pred}+1}^{t_{pred}+t_f} p\left(Y_l^t \mid I_l^R, S, \mathcal{M}\right) \right) \left(\prod_{t=t_{pred}+1}^{t_{pred}+t_f} p\left((Y_l^t)^R \mid I_l^{R-1}, S, \mathcal{M}\right) \right) \cdots \left(\prod_{t=t_{pred}+1}^{t_{pred}+t_f} p\left((Y_l^t)^0 \mid S, \mathcal{M}\right) \right) \tag{5.11}$$

The conditional probability $P(Y_l|S,\mathcal{M})$ of each of the possible trajectories is the sum of the product of the probabilities of each output sequence state conditioned on an intermediate variable I_l^r for $r = 1, \ldots, R$, the past trajectory and the map features. Each product term is implemented by a function approximator NN that receives as inputs the previous intermediate variable, the past trajectory and the map features outputs the conditional distribution over the future trajectory for each time step.

5.2.6.2 Deterministic Approximation

The marginalization in Equation 5.11 is intractable. Therefore, a deterministic approximation is deployed so that the entire lower bound can be written as:

$$log(P(Y_l|S,\mathcal{M})) \geqslant \left(\sum_{t=t_{pred}+1}^{t_{pred}+t_f} log\ p\left(Y_l^t \mid \hat{I}_l^R, S, \mathcal{M}\right)\right) + \cdots + \left(\sum_{t=t_{pred}+1}^{t_{pred}+t_f} log\ p\left((Y_l^t)^1 \mid \hat{I}_l^0, S, \mathcal{M}\right)\right) + \left(\sum_{t=t_{pred}+1}^{t_{pred}+t_f} log\ p\left((\hat{Y}_l^t)^0 \mid S, \mathcal{M}\right)\right) \tag{5.12}$$

where $(\hat{Y}_l^t)^0$ refers to the most likely value according to its distribution $p\left((Y_l^t)^0 \mid S, \mathcal{M}\right)$.

5.2.6.3 Iterative Refinement

The latent variables are constrained to be of the same type as the target sequence in order to share an underlying neural network. Each set of latent variables becomes a future trajectory we are predicting. Therefore, each conditional $p\left(I_l^r \mid \hat{I}_l^{r-1}, X\right)$ can be considered as a step of refinement of a future trajectory $\hat{I}_l^{r-1}$. The task of generating those intermediate latent variables intuitively becomes the task of refining our target sequence R times before we generate the final target sequence. This allows us to use a simple supervised optimization to train those latent variables. At each refinement step, we minimise the loss between the intermediate sequence that is generated and the ground truth sequence.

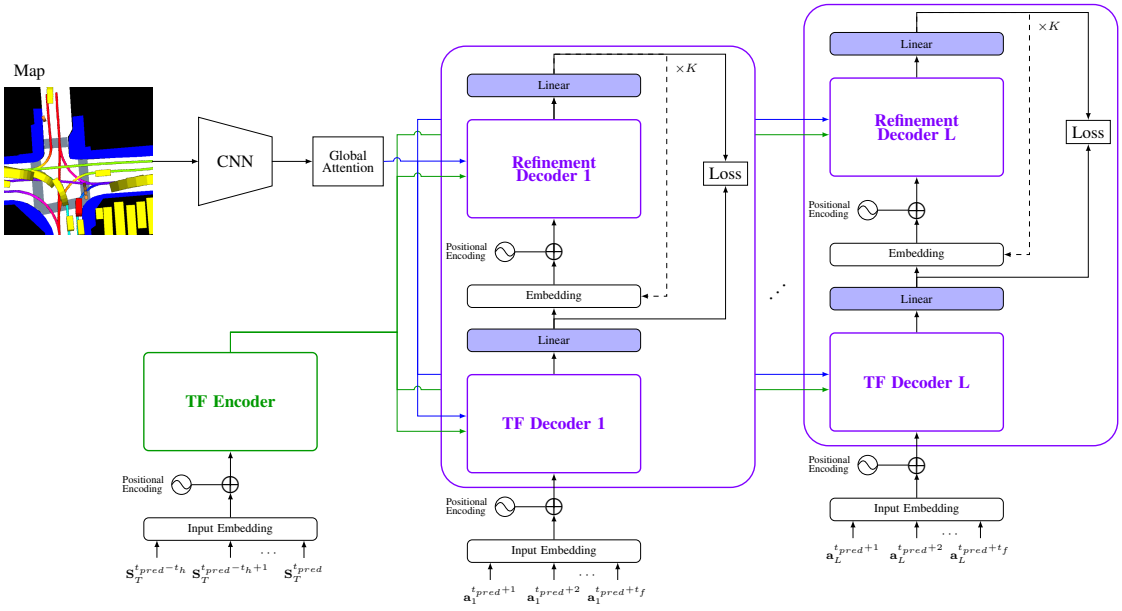


Figure 5.4: Spatio-Temporal Multimodal Transformer with refinement (**ST-M-TF-R**)

5.2.7 Classification Layer

Our TF-based decoders generate L possible predictions of the future trajectory. However, based on the target vehicle's recent motions and its surrounding environment, some trajectories are more likely to happen than others. A classification layer is added to our model in order to infer the likelihood of each of the predicted trajectories. This layer receives as inputs the last encoding vector of the TF encoder and the map global features extracted by the global attention block. It is composed of two fully connected layers separated by an activation function. A *softmax* function is applied to the output to generate the probability of each of the predicted trajectories.

5.2.8 Multimodal Network Architecture

We augment our spatio-temporal TF with refinement blocks in form of TF decoders sharing the parameters across the refinement steps. As a result, our overall model is composed of the CNN and three transformer-based network blocks: a TF encoder and two types of TF decoders.

- The first block is the TF encoder that encodes the input S.
- The second block is a decoder TF that models the first conditional $log\ p\left((\hat{Y}_l^t)^0 \mid S, \mathcal{M}\right)$. Since we have a multimodal solution, we deploy L decoders with shared weights to generate L possible trajectories.
  The two previous blocks are mounted and deployed identically to our first approach without refinement (cf. Figure 5.1).
- The third block is a decoder TF that we denote a refinement decoder modeling $log\ p\left(I_l^r \mid \hat{I}_l^{r-1}, X\right)$. The refinement is applied on top of the output of each of the previous L decoders and therefore L refinement decoders with shared weights are used. Each one receives as input the embedding of the trajectory predicted by the previous decoder. In addition, the refinement is applied iteratively using R refinement steps. As a result, the total number of refinement decoders is $L \times R$ decoders with shared weights.

5.2.8.1 Prediction

The inference of our proposed multi-modal solution is entirely deterministic. We start from the input S and first predict the initial L possible target sequences by $(\hat{Y}_l^t)^0 = argmax\ log\ p\left((Y_l^t)^0 \mid S, \mathcal{M}\right)$, for $t = t_{pred} + 1, \cdots, t_{pred} + t_f$ and $l = 1, \cdots, L$. We continue refining each target sequence l by $\hat{I}_t^r = argmax\ log\ p\left(I_t^r \mid S, \mathcal{M}\right)$, for $t =$

Table 5.1: Results of comparative analysis on nuScenes dataset, over a prediction horizon of 6 seconds

	MinADE_1	MinADE_5	MinADE_{10}	MinFDE_1	MinFDE_5	MinFDE_{10}	$\text{MissRate}_{5,2}$	$\text{MissRate}_{10,2}$	Off-Road Rate
Multipath [16]	4.43	1.78	1.55	10.16	3.62	2.93	0.78	0.76	0.36
CoverNet[1] [81]	-	2.62	1.92	11.36	-	-	0.76	0.64	0.13
Trajectron++[1] [91]	-	1.88	1.51	9.52	-	-	0.70	0.57	0.25
MHA-JAM	3.77	1.85	1.24	8.65	3.85	2.23	0.60	0.46	0.10
MHA-JAM (off-road)[1]	3.69	1.81	1.24	8.57	3.72	2.21	**0.59**	**0.45**	0.07
T-M-TF	4.23	2.04	1.48	9.90	4.46	2.92	0.69	0.66	0.31
ST-M-TF	**3.66**	1.75	1.23	**8.34**	3.66	**2.08**	0.62	0.52	0.07
ST-M-TF-R	3.88	**1.71**	**1.21**	8.62	**3.45**	2.09	0.64	0.55	**0.05**

$t_{pred}+1, \cdots, t_{pred}+t_f$.
We note that the number of refinement iterations increases the trajectory generation time since each refinement is applied based on the previous output.

5.2.9 Loss Functions

5.2.9.1 Refinement loss

At each refinement step, we minimise the regression loss between the intermediate sequence that is generated and the ground truth sequence.

$$L_{ref} = -\sum_{r=0}^{R+1}\left(\sum_{t=1}^{T} log\ p\left(y_t \mid \hat{I}^{r-1}, X\right)\right) \tag{5.13}$$

5.2.9.2 Total loss

We train the model using a weighted sum of the regression, classification (cf. definitions in Section 4.1.9) and refinement loss functions:

$$Loss = L_{reg} + \lambda_{cl}.L_{cl} + \lambda_{ref}.L_{ref}, \tag{5.14}$$

where the weights λ_{cl} and λ_{ref} are empirically determined hyperparameters.

5.3 Experimental Analysis and Evaluations

We train and test our proposed methods on the nuScenes dataset and we evaluate them using the metrics described in Section 4.4.2.

5.3.1 Training and Implementation Details

The input states are embedded in a space of dimension 128. We use the same dimension and resolution of the image representation of the scene map (cf. Figure 4.6a) as with the MHA-JAM method. Similarly, we use ResNet-50 pretrained on ImageNet to extract

map features. This CNN outputs a map features of size $(28, 28, 512)$. The deployed TF encoder and decoder are of 4 layers and 8 attention heads randomly initialized. We use $L = 10$ parallel TF Decoders having the same parameters and $R \times L = 1 \times 10$ refinement decoders with the same parameters. We use a batch size of 32 and the Adam optimizer [52]. The model is implemented using PyTorch [78].

5.3.2 Models Compared

We compare our model to three recently proposed baselines for multimodal trajectory prediction, and two of our proposed models (cf. description in 4.4.3). We report the results considering the K most probable trajectories generated by each model.
our proposed TF-based models:

Temporal Multimodal Transformer (T-M-TF): A basic Temporal Multimodal TF without modeling the interactions with the scene.

Spatio-Temporal Multimodal Transformer (ST-M-TF): Our proposed Spatio-Temporal Multimodal TF presented in Section 5.2.

Spatio-Temporal Multimodal Transformer with Refinement (ST-M-TF-R): Our proposed Spatio-Temporal Multimodal TF with Refinement described in Section 5.2.6.

5.3.3 Quantitative Evaluation

We compare our model Spatio-Temporal TF with the predefined Anchors and Refinement approach **ST-M-TF-R** to various baselines and to our recurrent approach **MHA-JAM** using different metrics (cf. definitions in Section 4.4.2) in Table 5.1. Our model has competitive results with the compared methods. It outperforms the state-of-the-art methods Multipath [16], CoverNet [81] and Trajectron++ [91] according to all the metrics considered. However, it performs better than our **MHA-JAM** on 5 of the 9 reported metrics.

In addition, our **ST-M-TF-R** achieves the lowest off-road rate even if it was not trained with the off-road loss. This highlights the importance of using the additional Map-Decoder MHA layer in generating scene compliant trajectories.

5.3.4 Ablation Experiments

To get a deeper insight into the relative contributions of the input cues and modules affecting the overall performance, we perform the following ablation experiments.

5.3.4.1 Importance of the refinement approach

Our first experiment consists in evaluating the performance of the refinement step. To do so, we train our model without the refinement step. We denote the resulting method spatio-temporal TF with predefined anchors **ST-M-TF**. We observe that the model with refinement **ST-M-TF-R** has comparable performance to the **ST-M-TF**. The contribution of the refinement approach is not quite clear. Additional experiments with different parameters will be conducted in future work to further investigate the importance of the refinement steps.

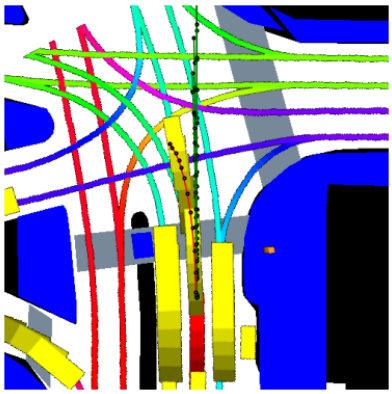
5.3.4.2 Importance of map information

In order to evaluate the importance of the map-decoder layer, we remove it from the TF decoder. We denote the resulting model temporal TF with predefined anchors **T-M-TF**. This experiment evaluates the importance of the map features in determining the future trajectory of the target vehicle. We train this model and we compare it with our **ST-M-TF**. We observe a big degradation in the performance of the **T-M-TF** compared to the **ST-M-TF** according to all the metrics considered (cf. Table 5.1). This highlights the importance of the map features in predicting the possible trajectories of a target agent. In addition, the **T-M-TF** has the lowest off-road rate. This implies that the generated trajectories are not compliant with the scene structure.

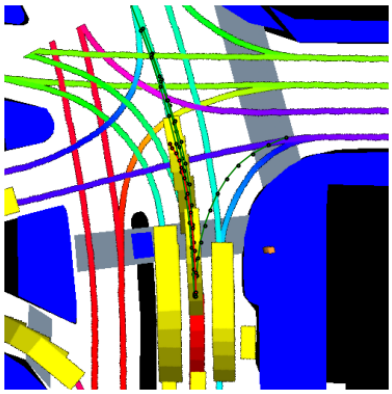
5.3.5 Qualitative Evaluation

We perform a quantitative evaluation in order to visualise samples of the generated trajectories and investigate the performance of our **ST-M-TF-R**. We also compare it with the basic **T-M-TF**. Figure shows the 5 most probable trajectories predicted using the two methods in different scenarios: an intersection, a roundabout and a curved road. Let's examine the general aspect of the generated trajectories. We note that the temporal evolution of the motion is consistent even though each position is generated independently of the previous ones.

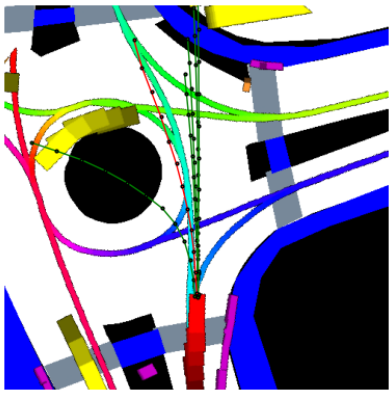
Considering the first scenario, we notice that our **ST-M-TF-R** succeeded in generating diverse trajectories compliant with the scene structure. Actually, it predicts two possible maneuvers; moving straight on or turning right. Even though among the 5 most probable trajectories, the turning right maneuver is not clearly predicted, we notice that they are diverse enough since the moving straight on maneuver is predicted with different speed profiles. We also observe that the **T-M-TF** predicts only the moving straight on maneuver without any deviation, in contrast to the **ST-M-TF-R**, which generates trajectories that follow the lane curvature.



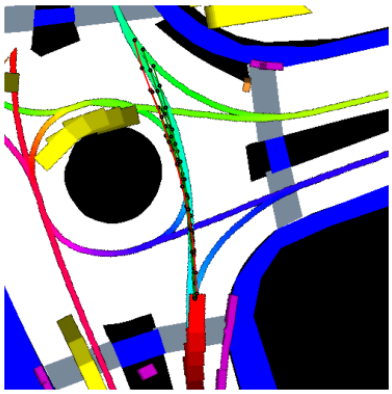
(a) Predicted trajectories using **T-M-TF**



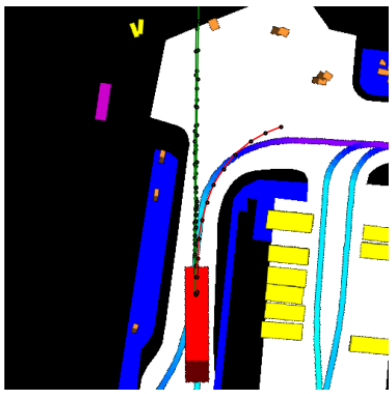
(b) Predicted trajectories using **ST-M-TF-R**



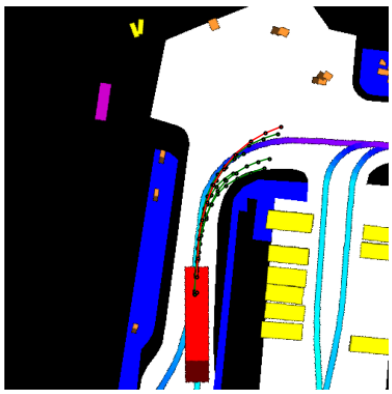
(c) Predicted trajectories using **T-M-TF**



(d) Predicted trajectories using **ST-M-TF-R**



(e) Predicted trajectories using **T-M-TF**



(f) Predicted trajectories using **ST-M-TF-R**

Figure 5.5: Visualisation of predicted trajectories using two different transformer models; the basic temporal TF (T-M-TF) and our spatio-temporal TF with prediction refinement (ST-M-TF-R). The predicted trajectories are marked in green and the ground truth in red. Three different contexts are presented: an intersection, a roundabout and a curved road.

The second example shows a trajectory prediction scenario near a roundabout. In contrast to the **ST-M-TF-R**, the **T-M-TF** generates more diverse predictions. However, the trajectories predicted using **ST-M-TF-R** are more compliant to the scene; they follow the lane mark and fit well to the ground truth.
The third example presents trajectory prediction on a curved road. Our **ST-M-TF-R** generates trajectories coherent with the context as well as fitting the best with the ground truth.

5.4 Conclusion

Following on the recent success of non-autoregressive neural sequence modeling, we propose a Non-Autoregressive Transformer to tackle the task of trajectory prediction for autonomous vehicles. This architecture is interesting since it has the ability of parallellized implementation both during the training and the testing phase. We augment the TF architecture with an additional layer to simultaneously model the spatial and temporal information. In addition, we build a multimodal solution using a set of predefined intentions in the form of anchor trajectories that we them pass as inputs to the TF decoder. We also apply a refinement approach in order to implicitly capture the dependencies among target sequence elements. We evaluate our proposed solution using nuScenes dataset. It gives competitive results with the state-of-the-art methods.

6
Conclusion and Perspective

Contents

6.1 Contributions 130
6.2 Discussion 131
6.3 Research Perspectives 132

This chapter highlights our main contributions in this thesis as well as the future work that can be derived from the accomplished goals along this thesis.

6.1 Contributions

In this thesis, we started by addressing the task of trajectory prediction in a highway environment using two different methods; First, we brought RRNNs to tackle the vehicle motion prediction problem. In other words, we proposed an RRNNs based encoder-decoder architecture where the encoder analyzes the patterns and the dynamics underlying the past trajectories and the decoder generates the future trajectory sequence. The originality of this network is that it combines the advantages of the LSTM blocks in representing the temporal evolution of trajectories and the attention mechanism to model the relative interactions between the surrounding vehicles. We compared the proposed approach with the LSTM encoder decoder using the new large scaled naturalistic driving highD dataset.

Second, we adopted the attention mechanism to derive the relative importance of surrounding vehicles with respect to their whole past motion: We use LSTM to encode the trajectories of the target vehicle and its surroundings. Then, the attention mechanism directly relates the target vehicle motion encoding to the surrounding vehicles' ones and deploy dot product operation to determine the importance of each vehicle. It selectively aggregates the features that model the interaction between vehicles based on their importance. We also used multiple attention heads in order to extract different types of interactions and combined them to capture higher order relationships. This provided a better understanding of the dynamic context of the target vehicle.

Then, we considered the task of trajectory prediction in an urban environment, We proposed a novel method of applying MHA (**MHA-JAM**), that enhances modeling the interdependence between the surrounding agents motion and the road elements, by considering a joint representation of agents and maps features to generate keys and values of the attention heads. To handle the future uncertainties, our proposed (**MHA-JAM**) presented multimodal predictions, with each attention head specializing in one mode of the predicted distribution. Our formulation allows each attention head to focus on different subsets of the scene and surrounding agents, relevant to the corresponding mode of the predictive distribution.

In order to enhance the diversity of the predicted trajectories, we extended our method using three intention definition techniques: goal directed, anchors and region based intention definition. Each of them associates an intention to each trajectory prediction mode.

Our MHA with Joint Agent Map representation (**MHA-JAM**) has competitive performance compared to the state of the art on the benchmark nuScenes dataset [13], while also allowing for additional interpretability through visualization of the attention weights. Notably, our method ranked second place in the nuScenes motion prediction challenge.

In addition, we proposed an alternative attention based method using the double attention (DA) network. This method reduces the complexity of the **MHA-JAM** while producing competitive predictions. In addition, we expand our proposed methods to deal with the problem of simultaneous multi-vehicle trajectory prediction.

Finally, we proposed a Non-Autoregressive Transformer to tackle the task of trajectory prediction for autonomous vehicles in an urban environment. This architecture is interesting since it has the ability of parallelized implementation during the training and the testing phase as well. In fact, we augmented the basic TF architecture with an additional layer to simultaneously model the spatial and temporal information. In addition, we built a multimodal solution using a set of predefined intentions in form of anchor trajectories that we pass as inputs to the TF decoder. We also applied a refinement approach in order to implicitly capture the dependencies among target sequence elements.

6.2 Discussion

Let us now describe the main characteristics of the trajectory prediction methods introduced in this manuscript:

When considering a highway environment, we proposed two different methods of trajectory prediction; an RRNN based model and an MHA based pooling model. Both methods combine the advantages of two modules: the multi-head dot product attention mechanism and LSTMs to capture the spatio-temporal dependencies between the input tracks. However, these modules are mounted differently which constitutes the singularity of each method. The RRNN based model simultaneously captures spacial and temporal dependencies between vehicles tracks for each past time step using an MHA encapsulated in LSTM blocks. Therefore, it requires important computational resources. In contrast, our MHA based pooling technique applies the MHA, only once at the prediction time, to build a context representation based on the relative importance of surrounding vehicles with respect to their overall motion representation. Another advantage of our *MHA-LSTM* consists on its interpretable interaction modeling. Indeed, the visualisation of the attention maps of our *MHA-LSTM* enabled us to derive the importance and the dependencies between vehicles. Finally, by comparing the results of the experiments conducted on the naturalistic large-scale driving highD dataset [55], we come up with the deduction that our *MHA-LSTM* has better performance than RRNNs.

Then, we move on to the task of vehicle trajectory prediction in an urban environment while considering interactions between the target vehicle, its surrounding agents and the scene. We proposed an MHA-based method on a joint agents and map global context representation (MHA-JAM). Our MHA-JAM used each attention head to generate a distinct future trajectory to address multimodality of future trajectories. Then, we augmented it with different methods of defining drivers intentions and attributing an intention to each mode. While defining an intention for each prediction mode enhances the generated trajectories diversity, it slightly reduces the performance of our basic MHA-JAM. Therefore, there is a trade-off between ensuring that the predicted trajectories cover all the possible ways and generating more accurate predictions. Moreover, we investigated the introduction of dynamic constraints [57] on our basic (MHA-JAM) model.We noticed that the imposed constraints reduces the performance of our (MHA-JAM) model. Indeed, there is a compromise between generating realistic trajectories that respect physical rules, and producing more accurate predictions.

In addition, we proposed an alternative attention based module for interaction modeling and multimodal trajectory generation based on the double attention network (DA-JAM). While our DA-JAM has reduced complexity compared to our MHA-JAM, it has also a competitive performance. However, the MHA-JAM has a more straight forward interpretation capability. Indeed, the visualization of the attention maps of our MHA-JAM reveals the importance of joint agents and map features and the interactions occurring during the execution of each possible maneuver.

Finally, we proposed a spatio-temporal multimodal Non-Autoregressive TF based model. This architecture deploys TF to model the target vehicle motion contrarily to the previous methods that used the recurrent LSTMs. Our TF based model is interesting since it has has competitive performance with our MHA-JAM. In addition, it has the ability of parallelized implementation both during the training and the testing phase. However, our current implementations of MHA-JAM and DA-JAM have comparable execution times (around 11 ms) while our TF-based model's execution time is on average 18 ms (run on NVIDIA GeForge GTX 1080-Ti GPU OC 11). In the future work, we will further investigate the use of TF in the task of trajectory prediction and its advantages compared to an LSTM based solution.

6.3 Research Perspectives

In the following, we present future work that can be derived from the accomplished goals along this thesis.

Additional Input Cues

In our work, we exploited two main contextual cues which are the surrounding agents recent motions and the high definition map in order to infer a target agent future motion. However, additional contextual cues are required for a more effective and plausible prediction especially in an urban environment. Actually, information about the traffic lights, road signs and one way roads are essential for inferring the possible future maneuvers. In future work, we plan to investigate these cues and create convenient representations for them.

Adapted Intention Definition

Drivers' behaviors are not deterministic. In similar driving situations, they can perform different maneuvers depending on their goal (destination) or even when doing the same maneuver, the execution can be different in terms of speed and pattern. In order to represent the uncertainty of the future, we proposed methods that predict a multi-modal finite set of trajectories that correspond to distinct intentions.
In this work, we used two main approaches of defining intentions:

- Implicit intentions in form of latent variables presenting the underlying possible modes.

- Explicit intentions (anchors or goals) defined prior to the training using heuristic methods.

Other methods of defining intentions or identifying the goals adapted to each scene can be explored in the future such as reinforcement learning methods.

Collision-free simultaneous multi-vehicle trajectory prediction

In this work, we considered the task of simultaneous multi-vehicle trajectory prediction. However, in most cases, vehicles trajectories are coherent and collisions and dangerous behaviors are avoided. In our future work, we are going to further explore this aspect by ensuring that the predicted trajectories does not present unrealistic conflicting behaviors and collisions.

Trajectory Prediction Methods Evaluation

In the future work, we aim to improve the way we are evaluating our predicted trajectories. Actually, a multimodal predictor generates multiple trajectories. However, we have only one ground truth trajectory. Therefore, we can evaluate only one of the predicted trajectories. The other predicted trajectories are supposed to be plausible and diverse. In our work, we consider the off-road metric to evaluate the plausibility of the generated trajectories. In addition, we visualize samples of the predicted trajectories to verify their diversity. These methods are not sufficient to accurately evaluate the performance of a multi-modal solution. Further metrics should be conceived to evaluate the plausibility and the diversity of the predicted trajectories for a fairer comparison of the different multi-modal predictors.

In addition, we aim to evaluate the performance of our proposed prediction methods in a prediction then planning pipeline so that the impacts of single vs multimodal trajectory prediction to the closed-loop performance can be evaluated. The trajectory prediction will be integrated in a simulator with many challenging and interactive scenarios in complex scenes to evaluate their performance. For each scenario, the simulator includes virtual, dynamic, and reactive agents as the surrounding agents of the target vehicle.

References

[1] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. “Social LSTM: Human Trajectory Prediction in Crowded Spaces”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. June 2016.

[2] Altché, F. and La Fortelle, A. de. “An LSTM network for highway trajectory prediction”. In: *IEEE International Conference on Intelligent Transportation Systems, ITSC*. Oct. 2017, pp. 353–359.

[3] Amirian, J., Hayet, J.-B., and Pettré, J. “Social ways: Learning multi-modal distributions of pedestrian trajectories with GANs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 0–0.

[4] Ammoun, S. and Nashashibi, F. “Real time trajectory prediction for collision risk estimation between vehicles”. In: *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*. 2009, pp. 417–422.

[5] Aoude, G., Joseph, J., Roy, N., and How, J. “Mobile Agent Trajectory Prediction using Bayesian Nonparametric Reachability Trees”. In: *AIAA Infotech at Aerospace Conference and Exhibit 2011*. Mar. 2011.

[6] Bahdanau, D., Cho, K., and Bengio, Y. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR*. May 2015.

[7] Barth, A. and Franke, U. “Where will the oncoming vehicle be the next second?” In: *IEEE Intelligent Vehicles Symposium, IV*. June 2008, pp. 1068–1073.

[8] Bartoli, F., Lisanti, G., Ballan, L., and Del Bimbo, A. “Context-Aware Trajectory Prediction”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, pp. 1941–1946.

[9] Bhattacharyya, A., Hanselmann, M., Fritz, M., Schiele, B., and Straehle, C.-N. “Conditional Flow Variational Autoencoders for Structured Sequence Prediction”. In: *arXiv e-prints*, arXiv:1908.09008 (Aug. 2019), arXiv:1908.09008. arXiv: 1908.09008 [cs.CV].

[10] Bock, J., Krajewski, R., Moers, T., Runde, S., Vater, L., and Eckstein, L. “The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections”. In: 2019.

[11] Broadhurst, A., Baker, S., and Kanade, T. “Monte Carlo road safety reasoning”. In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* 2005, pp. 319–324.

[12] Buhet, T., Wirbel, E., and Perrotton, X. “PLOP: Probabilistic poLynomial Objects trajectory Planning for autonomous driving”. In: *Proceedings of the 4th Conference on Robot Learning, CoRL*. Mar. 2020.

[13] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. "nuScenes: A Multimodal Dataset for Autonomous Driving". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[14] Casas, S., Gulino, C., Liao, R., and Urtasun, R. "SpAGNN: Spatially-Aware Graph Neural Networks for Relational Behavior Forecasting from Sensor Data". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9491–9497.

[15] Casas, S., Luo, W., and Urtasun, R. "IntentNet: Learning to Predict Intention from Raw Sensor Data". In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by A. Billard, A. Dragan, J. Peters, and J. Morimoto. Vol. 87. Proceedings of Machine Learning Research. PMLR, 29–31 Oct 2018, pp. 947–956.

[16] Chai, Y., Sapp, B., Bansal, M., and Anguelov, D. "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction". In: *CoRL*. 2019.

[17] Chang, M.-F. et al. "Argoverse: 3D Tracking and Forecasting with Rich Maps". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[18] Chen, Y., Kalantidis, Y., Li, J., Yan, S., and Feng, J. "A^2-Nets: Double Attention Networks". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. 2018, pp. 352–361.

[19] Chollet, F. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. July 2017, pp. 1800–1807.

[20] Coifman, B. and Li, L. "A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset". In: *Transportation Research Part B: Methodological* 105 (Nov. 2017), pp. 362–377.

[21] Cui, H., Radosavljevic, V., Chou, F.-C., Lin, T.-H., Nguyen, T., Huang, T.-K., Schneider, J., and Djuric, N. "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks". In: *International Conference on Robotics and Automation (ICRA)* (2019).

[22] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. "Universal Transformers". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HyzdRiR9Y7.

[23] Deo, N. and Trivedi, M. M. "Convolutional Social Pooling for Vehicle Trajectory Prediction". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW*. 2018, pp. 1468–1476.

[24] Deo, N. and Trivedi, M. M. "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs". In: *IEEE Intelligent Vehicles Symposium, IV*. 2018, pp. 1179–1184.

[25] Deo, N. and Trivedi, M. M. "Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans". In: *ArXiv* abs/2001.00735 (2020).

[26] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. cite arxiv:1810.04805. 2018. URL: http://arxiv.org/abs/1810.04805.

[27] Ding, W., Chen, J., and Shen, S. “Predicting Vehicle Behaviors Over An Extended Horizon Using Behavior Interaction Network”. In: *International Conference on Robotics and Automation, ICRA*. May 2019, pp. 8634–8640.

[28] Djuric, N., Radosavljevic, V., Cui, H., Nguyen, T., Chou, F., Lin, T., Singh, N., and Schneider, J. “Uncertainty-aware Short-term Motion Prediction of Traffic Actors for Autonomous Driving”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 2084–2093.

[29] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[30] Elnagar, A. “Prediction of moving objects in dynamic environments using Kalman filters”. In: *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*. 2001, pp. 414–419.

[31] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[32] Giuliari, F., Hasan, I., Cristani, M., and Galasso, F. “Transformer Networks for Trajectory Forecasting”. In: *International Conference on Pattern Recognition*. 2020.

[33] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[34] Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. “Non-Autoregressive Neural Machine Translation”. In: *International Conference on Learning Representations*. 2018.

[35] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. “Social GAN: Socially Acceptable Trajectories With Generative Adversarial Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. June 2018.

[36] He, K., Zhang, X., Ren, S., and Sun, J. “Deep Residual Learning for Image Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*. June 2016, pp. 770–778.

[37] Hermes, C., Wohler, C., Schenk, K., and Kummert, F. “Long-term vehicle motion prediction”. In: *IEEE Intelligent Vehicles Symposium, IV*. June 2009, pp. 652–657.

[38] Hong, J., Sapp, B., and Philbin, J. “Rules of the Road: Predicting Driving Behavior With a Convolutional Model of Semantic Interactions”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8446–8454.

[39] Hou, L., Xin, L., Li, S., Cheng, B., and Wang, W. “Interactive Trajectory Prediction of Surrounding Road Users for Autonomous Driving Using Structural-LSTM Network”. In: *IEEE Transactions on Intelligent Transportation Systems* (2019), pp. 1–11.

[40] Houenou, A., Bonnifait, P., Cherfaoui, V., and Yao, W. “Vehicle trajectory prediction based on motion model and maneuver recognition”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2013, pp. 4363–4369.

[41] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. `https://level5.lyft.com/dataset/`. 2020.

[42] *How to Build a Motion Prediction Model for Autonomous Vehicles*. `https://medium.com/lyftself-driving/how-to-build-a-motion-prediction-model-for-autonomous-vehicles-29f7f81f1580`. Accessed: 2021-04-26.

[43] Huang, Y., Bi, H., Li, Z., Mao, T., and Wang, Z. "STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6271–6280.

[44] J. Colyar and J. Halkias. "Us highway i-80 dataset." In: *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-06-137*. 2006.

[45] J. Colyar and J. Halkias. "Us highway 101 dataset." In: *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT07-030*. 2007.

[46] Jo, K., Lee, M., Kim, J., and Sunwoo, M. "Tracking and Behavior Reasoning of Moving Vehicles Based on Roadway Geometry Constraints". In: *IEEE Transactions on Intelligent Transportation Systems* 18.2 (2017), pp. 460–476.

[47] Kaempchen, N., Weiss, K., Schaefer, M., and Dietmayer, K. C. J. "IMM object tracking for high dynamic driving maneuvers". In: *IEEE Intelligent Vehicles Symposium, 2004*. 2004, pp. 825–830.

[48] Käfer, E., Hermes, C., Wöhler, C., Ritter, H., and Kummert, F. "Recognition of situation classes at road intersections". In: *IEEE International Conference on Robotics and Automation, ICRA*. May 2010, pp. 3960–3965.

[49] Karasev, V., Ayvaci, A., Heisele, B., and Soatto, S. "Intent-aware long-term prediction of pedestrian motion". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2543–2549.

[50] Kim, B., Kang, C. M., Kim, J., Lee, S. H., Chung, C. C., and Choi, J. W. "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 399–404.

[51] Kim, H., Kim, D., Kim, G., Cho, J., and Huh, K. "Multi-Head Attention based Probabilistic Vehicle Trajectory Prediction". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1720–1725.

[52] Kingma, D. P. and Ba, J. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR*. May 2015.

[53] Kingma, D. P. and Welling, M. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[54] Kosaraju, V., Sadeghian, A., Martıén-Martıén, R., Reid, I., Rezatofighi, H., and Savarese, S. "Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019, pp. 137–146. URL: https://proceedings.neurips.cc/paper/2019/file/d09bf41544a3365a46c9077ebb5e35c3-Paper.pdf.

[55] Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems". In: *IEEE International Conference on Intelligent Transportation Systems, ITSC*. Nov. 2018, pp. 2118–2125.

[56] Krajewski, R., Moers, T., Bock, J., Vater, L., and Eckstein, L. "The rounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–6.

[57] LaValle, S. M. *Planning Algorithms*. USA: Cambridge University Press, 2006, pp. 743–744.

[58] Lee, J., Mansimov, E., and Cho, K. "Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1173–1182. URL: https://www.aclweb.org/anthology/D18-1149.

[59] Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H. S., and Chandraker, M. "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2165–2174.

[60] Lefèvre, S., Vasquez, D., and Laugier, C. "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH Journal* 1.1 (2014), pp. 1–14.

[61] Lenz, D., Diehl, F., Le, M. T., and Knoll, A. "Deep neural networks for Markovian interactive scene prediction in highway scenarios". In: *IEEE Intelligent Vehicles Symposium, IV*. June 2017, pp. 685–692.

[62] Lerner, A., Chrysanthou, Y., and Lischinski, D. "Crowds by Example". In: *Computer Graphics Forum* 26 (2007).

[63] Li, J., Ma, H., Zhang, Z., and Tomizuka, M. *Social-WaGDAT: Interaction-aware Trajectory Prediction via Wasserstein Graph Double-Attention Network*. 2020. arXiv: 2002.06241 [cs.CV].

[64] Li, Y. "Pedestrian Path Forecasting in Crowd: A Deep Spatio-Temporal Perspective". In: *Proceedings of the 25th ACM international conference on Multimedia* (2017).

[65] Li, Y. "Which Way Are You Going? Imitative Decision Learning for Path Forecasting in Dynamic Scenes". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[66] Libovický, J., Helcl, J., and Mareček, D. "Input Combination Strategies for Multi-Source Transformer Decoder". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 253–260. URL: https://www.aclweb.org/anthology/W18-6326.

[67] Mercat, J., Gilles, T., El Zoghby, N., Sandou, G., Beauvois, D., and Gil, G. P. "Multi-Head Attention for Multi-Modal Joint Vehicle Motion Forecasting". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9638–9644.

[68] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. "Non-local Social Pooling for Vehicle Trajectory Prediction". In: *IEEE Intelligent Vehicles Symposium, IV*. June 2019, pp. 975–980. URL: https://hal.inria.fr/hal-02160409.

[69] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. "Relational Recurrent Neural Networks For Vehicle Trajectory Prediction". In: *IEEE Intelligent Transportation Systems Conference, ITSC*. Oct. 2019, pp. 1813–1818. URL: https://hal.inria.fr/hal-02195180.

[70] Messaoud, K., Deo, N., Trivedi, M. M., and Nashashibi, F. "Trajectory Prediction for Autonomous Driving based on Multi-Head Attention with Joint Agent-Map Representation". In: *IEEE Intelligent Vehicles Symposium, IV*. 2021.

[71] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. "Attention Based Vehicle Trajectory Prediction". In: *IEEE Transactions on Intelligent Vehicles* (Apr. 2020). URL: https://hal.inria.fr/hal-02543967.

[72] Misawa, H., Takenaka, K., Sugihara, T., Liu, H., Taniguchi, T., and Bando, T. "Prediction of driving behavior based on sequence to sequence model with parametric bias". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2017, pp. 1–6.

[73] Niedoba, M., Cui, H., Luo, K., Hegde, D., Chou, F.-C., and Djuric, N. "Improving Movement Prediction of Traffic Actors using Off-road Loss and Bias Mitigation". In: *Workshop on 'Machine Learning for Autonomous Driving' at Conference on Neural Information Processing Systems (ML4AD)*. 2019.

[74] Nikhil, N. and Tran Morris, B. "Convolutional Neural Network for Trajectory Prediction". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

[75] Palli-Thazha, V., Filliat, D., and Ibañez-Guzmán, J. "Trajectory Prediction of Traffic Agents: Incorporating context into machine learning approaches". In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. 2020, pp. 1–6.

[76] Park, S. H., Kim, B., Kang, C. M., Chung, C. C., and Choi, J. W. "Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture". In: *IEEE Intelligent Vehicles Symposium, IV*. June 2018, pp. 1672–1678.

[77] Park, S. H., Lee, G., Bhat, M., Seo, J., Kang, M.-S., Francis, J., Jadhav, A. R., Liang, P. P., and Morency, L.-P. "Diverse and Admissible Trajectory Forecasting through Multimodal Context Understanding". In: *ArXiv* abs/2003.03212 (2020).

[78] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. "Automatic differentiation in PyTorch". In: *NIPS Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*. Dec. 2017.

[79] Pellegrini, S., Ess, A., Schindler, K., and van Gool, L. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 261–268.

[80] Pfeiffer, M., Paolo, G., Sommer, H., Nieto, J., Siegwart, R., and Cadena, C. "A Data-driven Model for Interaction-Aware Pedestrian Motion Prediction in Object Cluttered Environments". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5921–5928.

[81] Phan-Minh, T., Grigore, E. C., Boulton, F. A., Beijbom, O., and Wolff, E. M. "CoverNet: Multimodal Behavior Prediction Using Trajectory Sets". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[82] Phillips, D. J., Wheeler, T. A., and Kochenderfer, M. J. "Generalizable intention prediction of human drivers at intersections". In: *IEEE Intelligent Vehicles Symposium, IV*. June 2017, pp. 1665–1670.

[83] Rhinehart, N., Kitani, K. M., and Vernaza, P. "R2P2: A ReparameteRized Pushforward Policy for Diverse, Precise Generative Path Forecasting". In: *The European Conference on Computer Vision, ECCV*. Sept. 2018, pp. 794–811.

[84] Rhinehart, N., McAllister, R., Kitani, K., and Levine, S. "PRECOG: PREdiction Conditioned on Goals in Visual Multi-Agent Settings". In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2019.

[85] Ridel, D. A., Deo, N., Wolf, D. F., and Trivedi, M. "Scene Compliant Trajectory Forecast With Agent-Centric Spatio-Temporal Grids". In: *IEEE Robotics and Automation Letters* 5 (2019), pp. 2816–2823.

[86] Robicquet, A., Sadeghian, A., Alahi, A., and Savarese, S. "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes". In: *ECCV*. 2016.

[87] Rudenko, A., Palmieri, L., Herman, M., Kitani, K. M., Gavrila, D. M., and Arras, K. O. "Human motion trajectory prediction: a survey". In: *The International Journal of Robotics Research* 39.8 (2020), pp. 895–935.

[88] Rummelhard, L., Nègre, A., Perrollaz, M., and Laugier, C. "Probabilistic Grid-based Collision Risk Prediction for Driving Application". In: *ISER*. Marrakech/Essaouira, Morocco, June 2014. URL: https://hal.inria.fr/hal-01011808.

[89] Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., and Savarese, S. "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. June 2019.

[90] Sadeghian, A., Legros, F., Voisin, M., Vesel, R., Alahi, A., and Savarese, S. "CAR-Net: Clairvoyant Attentive Recurrent Network". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[91] Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. "Trajectron++: Multi-Agent Generative Trajectory Forecasting With Heterogeneous Data for Control". In: *ArXiv* abs/2001.03093 (2020).

[92] Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. "Relational recurrent neural networks". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. 2018, pp. 7299–7310.

[93] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. “A simple neural network module for relational reasoning”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. Dec. 2017, pp. 4967–4976.

[94] Schlechtriemen, J., Wedel, A., Hillenbrand, J., Breuel, G., and Kuhnert, K. “A lane change detection approach using feature ranking with maximized predictive power”. In: *IEEE Intelligent Vehicles Symposium, IV*. June 2014, pp. 108–114.

[95] SHI, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and WOO, W.-c. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[96] Sierra González, D., Dibangoye, J. S., and Laugier, C. “High-speed highway scene prediction based on driver models learned from demonstrations”. In: *IEEE International Conference on Intelligent Transportation Systems, ITSC*. Nov. 2016, pp. 149–155.

[97] Sierra González, D., Romero-Cano, V., Dibangoye, J. S., and Laugier, C. “Interaction-aware driver maneuver inference in highways using realistic driver models”. In: *IEEE International Conference on Intelligent Transportation Systems, ITSC*. Oct. 2017, pp. 1–8.

[98] Sohn, K., Lee, H., and Yan, X. “Learning Structured Output Representation using Deep Conditional Generative Models”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015, pp. 3483–3491. URL: https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

[99] Sutskever, I., Vinyals, O., and Le, Q. V. “Sequence to Sequence Learning with Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014, pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[100] Tang, Y. C. and Salakhutdinov, R. “Multiple Futures Prediction”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[101] Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, pp. 54–64.

[102] Toledo-Moreo, R. and Zamora-Izquierdo, M. A. “IMM-Based Lane-Change Prediction in Highways With Low-Cost GPS/INS”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.1 (Mar. 2009), pp. 180–185.

[103] Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. “Wasserstein Auto-Encoders”. In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HkL7n1-0b.

[104] Tran, K., Bisazza, A., and Monz, C. "The Importance of Being Recurrent for Modeling Hierarchical Structure". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 4731–4736. URL: https://www.aclweb.org/anthology/D18-1503.

[105] Tran, Q. and Firl, J. "Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression". In: *IEEE Intelligent Vehicles Symposium, IV*. June 2014, pp. 918–923.

[106] Varshneya, D. and Srinivasaraghavan, G. "Human Trajectory Prediction using Spatially aware Deep Attention Models". In: *CoRR* abs/1705.09436 (2017). arXiv: 1705.09436. URL: http://arxiv.org/abs/1705.09436.

[107] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*.

[108] Veeraraghavan, H., Papanikolopoulos, N., and Schrater, P. "Deterministic sampling-based switching kalman filtering for vehicle tracking". In: *IEEE Intelligent Transportation Systems Conference, ITSC*. Sept. 2006, pp. 1340–1345.

[109] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. "Graph Attention Networks". In: *ArXiv* abs/1710.10903 (2018).

[110] Vemula, A., Muelling, K., and Oh, J. "Social Attention: Modeling Attention in Human Crowds". In: *IEEE International Conference on Robotics and Automation, ICRA*. May 2018, pp. 1–7.

[111] Wang, P., Huang, X., Cheng, X., Zhou, D., Geng, Q., and Yang, R. "The apolloscape open dataset for autonomous driving and its application". In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[112] Wang, X., Girshick, R., Gupta, A., and He, K. "Non-local Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. June 2018, pp. 7794–7803.

[113] Xin, L., Wang, P., Chan, C., Chen, J., Li, S. E., and Cheng, B. "Intention-aware Long Horizon Trajectory Prediction of Surrounding Vehicles using Dual LSTM Networks". In: *IEEE International Conference on Intelligent Transportation Systems, ITSC*. Nov. 2018, pp. 1441–1446.

[114] Xue, H., Huynh, D. Q., and Reynolds, M. "SS-LSTM: A Hierarchical LSTM Model for Pedestrian Trajectory Prediction". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 1186–1194.

[115] Yi, S., Li, H., and Wang, X. "Pedestrian Behavior Modeling From Stationary Crowds With Applications to Intelligent Surveillance". In: *IEEE Transactions on Image Processing* 25.9 (2016), pp. 4354–4368.

[116] Yu, C., Ma, X., Ren, J., Zhao, H., and Yi, S. "Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Aug. 2020.

[117] Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., and Urtasun, R. “End-To-End Interpretable Neural Motion Planner”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[118] Zhan, W. et al. “INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps”. In: *arXiv:1910.03088 [cs, eess]* (Sept. 2019).

[119] Zhang, H., Goodfellow, I. J., Metaxas, D. N., and Odena, A. “Self-Attention Generative Adversarial Networks”. In: *Proceedings of the 36th International Conference on Machine Learning ICML*. 2019, pp. 7354–7363.

[120] Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., and Wu, Y. N. “Multi-Agent Tensor Fusion for Contextual Trajectory Prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. June 2019.

[121] Zyner, A., Worrall, S., Ward, J., and Nebot, E. “Long short term memory for driver intent prediction”. In: *IEEE Intelligent Vehicles Symposium, IV*. June 2017, pp. 1484–1489.