Smart vehicule trajectory prediction in various autonomous driving scenarios

Gilles Thomas

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES Paris-PSL

Smart vehicule trajectory prediction in various autonomous driving scenarios

Prédiction intelligente des trajectoires de véhicules, dans différents scenarii de conduite autonome

Soutenue par
Thomas Gilles
Le 21 avril 2023

Dirigée par
Fabien MOUTARDE

Co-encadrée par
Bogdan STANCIULESCU
Dzmitry TSISHKOU

École doctorale n°621
Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité
Informatique temps réel, robotique et automatique.

Composition du jury :

Alexandre Alahi Professeur assistant, EPFL	*Rapporteur*
Fawzi Nashashibi Directeur de recherche, INRIA Paris	*Rapporteur*
David Filliat Professeur, ENSTA Paris	*Examinateur*
Anne Spalanzani Professeure, INRIA Grenoble-Alpes	*Examinateur*
John Folkesson Associate Professor, KTH	*Examinateur*
Bogdan Stanciulescu Maître de conférences, Mines Paris	*Examinateur*
Dzmitry Tsishkou Huawei Paris Research Center	*Examinateur*
Fabien Moutarde Professeur, Mines Paris	*Directeur de thèse*

MINES PARIS | PSL

Abstract

Recent advances in machine learning methods have enabled tremendous progress in autonomous driving, namely through the perception step thanks to deep learning and deep neural networks, combined with all-around progress in sensors, mapping and proprioception techniques. The focus is now therefore shifting towards the next steps in the autonomous pipeline, where prediction plays an important role. Once the surrounding road agents have been detected and tracked, the driving system needs to predict their future trajectory and plan accordingly to have a collision-less course.

This trajectory prediction must follow multiple requirements. First, it should obviously be accurate and trustworthy, so that its output can be reliably used in the following processes. The future can present multiple possibilities, from which it may not always be possible to disambiguate solely based on past historical data. The forecast must therefore be multimodal, by predicting multiple simultaneous probable futures. Since the prediction is to be made on all surrounding agents, and these agents behaviors are very much influenced by their interactions with each other, the model should take these interactions into account, and its multimodal predictions should be coherent with each other. Finally, for safety and reliability, the trajectory prediction should be easy to interpret, extensively evaluated, able to provide confidence evaluates and designed with its final use in the pipeline in mind.

In the first part of this dissertation, after recapitulating existing non-learning methods for trajectory forecasting, we study different existing representations and approaches for learning-based motion forecasting. We then propose to tackle the trajectory prediction problem using probability heatmaps to facilitate multimodality. We design three different ways of generating these heatmaps and evaluate them against each other and the existing state-of-the-art. We also provide a complete sampling method to extract actual trajectories from these heatmaps, and study the pro and cons of these heatmap methods compared to other commonly used frameworks. In the next chapter, we focus on multi-agent prediction, and more specifically consistent scene-level outputs, for these type of heatmap models through sampling and learned post-processing. Finally, we explore different ways of expanding prediction model evaluation by uncertainty assessment, calibration and cross-dataset generalizability analysis.

Résumé en Français

Les récentes avancées dans les méthodes d'apprentissage automatique ont permis des progrès considérables dans le domaine de la conduite autonome, notamment dans l'étape de perception, grâce à l'apprentissage profond et aux réseaux de neurones, combinés aux progrès généralisés des capteurs, de la localisation et des techniques de proprioception. L'attention se porte donc désormais sur les étapes suivantes du pipeline de la conduite autonome, où la prédiction joue un rôle important. Une fois que les agents routiers environnants ont été détectés, suivis et filtrés, le système de conduite doit prédire leur trajectoire future et planifier en conséquence pour éviter les collisions.

Cette prédiction de trajectoire doit répondre à de multiples exigences. Tout d'abord, elle doit être évidemment précise et sûre, afin que son résultat puisse être utilisé de manière fiable dans les processus suivants. Le futur peut présenter de multiples possibilités, qu'il n'est pas toujours possible de differencier sur la seule base des données historiques passées. La prévision doit donc être multimodale, en prédisant plusieurs futurs probables simultanés. Puisque la prévision doit être faite sur tous les agents environnants, et que les comportements de ces agents sont très influencés par leurs interactions, le modèle doit prendre en compte ces interactions, et ses prévisions multimodales doivent être cohérentes entre elles. Enfin, pour la sécurité et la fiabilité, la prédiction de trajectoire doit être facile à interpréter, largement évaluée, capable de fournir des évaluations de confiance et conçue avec son utilisation finale dans le processus global à l'esprit.

Dans la première partie de cette thèse, après avoir récapitulé les méthodes existantes de prévision de trajectoire n'utilisant pas l'apprentissage machine, nous étudions les différentes représentations et approches existantes pour l'estimation de mouvement par apprentissage. Nous proposons ensuite d'aborder le problème de la prédiction de trajectoire en utilisant des grilles probabilistes pour faciliter la multimodalité. Nous concevons trois manières différentes de générer ces cartes thermiques et nous les évaluons les unes par rapport aux autres et par rapport à l'état de l'art existant. Nous fournissons également une méthode d'extraction complète pour obtenir les trajectoires réelles à partir de ces cartes de probabilités, et nous étudions les avantages et les inconvénients de ces méthodes de grilles par rapport à d'autres approches couramment utilisés. Dans le chapitre suivant, nous nous concentrons sur la prédiction multi-agents, et plus particulièrement sur les prédictions cohérentes au niveau de la scène, pour ce type de modèles de grilles par le biais de l'extraction et d'une seconde étape apprise. Enfin, nous explorons différentes manières d'étendre l'évaluation des modèles de prédiction par l'évaluation de l'incertitude, la calibration et l'analyse de la généralisabilité entre jeux de données.

À Papi,

Remerciements

Un grand merci tout d'abord à Fabien et Bogdan pour leur encadrement aux Mines et leurs conseils. Je tiens également à remercier tous les membres du jury pour leur disponibilité, leur bienveillance et leurs conseils éclairés.

Merci beaucoup aux collègues de Huawei pour cette super ambiance pendant ces trois années, pour les parties de ping pong et les foots: Dzmitry, Nathan, Moussab, Luis, Quentin, Arthur et les anciens: Stefano évidemment, Laurent et Yann.

Un grand plaisir de ces années de thèse est de les partager avec d'autres doctorants, et j'ai pu faire partie d'une super équipe dans la V026 avec Mr Catastrophe, Raphaël, Camille, Joboule, Jesus et Sascha, rejoints par les nouveaux doctorants, les anciens et ceux des autres bureaux: Sofiane, Sami, Louis, Hugo, Amandine, Simon, Angelika, Jérémie, et j'en oublie.

J'ai également été supporté et soutenu par de nombreux amis, dans des supers colocation avec Thomas et Quentin, puis avec les bidasses Pierre-Louis et Adrien. Mes nombreuses réunions avec Xib, Malou et Zoz m'ont permis de développer une approche manutentionnaire appliquée dans mon travail.

Et enfin un grand merci évidemment à ma famille sans qui rien n'aurait été possible, à mes grands parents, mes oncles et tantes, mon frère Quentin. Papa Maman je vous aime.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

This chapter covers the background and motivation of this thesis, which tackles trajectory prediction, a problem relevant to both theoretical and practical applications.

1.1 Context: the autonomous driving pipeline

An autonomous driving vehicle shares a common architecture to most autonomous robotic systems, that can be summarized in 3 main steps illustrated in Fig. 1.1. This pipeline runs in closed loop and each of its steps are repeated one after another at a certain frequency, so that the system can assert and adapt to its evolving environment.

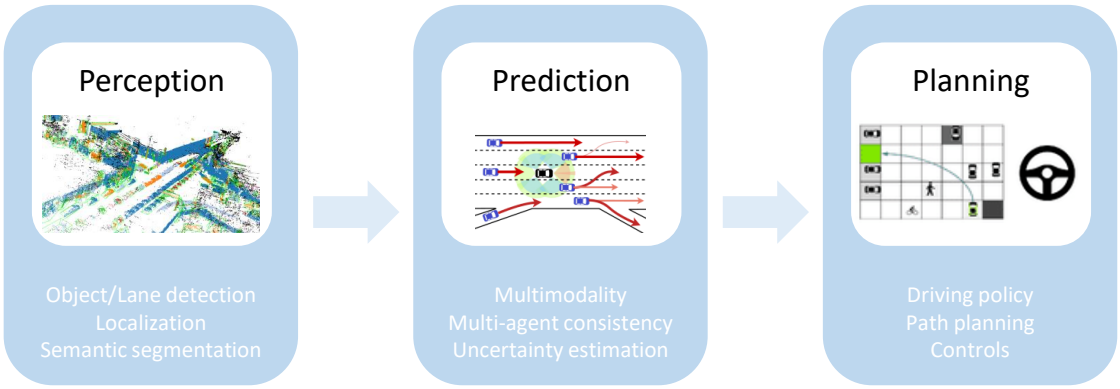


Figure 1.1: Autonomous driving pipeline. Illustrations from [Rigoulet, 2022, Deo and Trivedi, 2018b, Talpaert et al., 2019]

Perception The system needs to be aware of its environment. For that purpose, it is equipped with a set of sensors (cameras, radars, lidars, etc ...). The data from these sensors is processed by perception algorithms in order to extracts meaningful semantic information from it, such as depth estimation, drivable area, lane markings, objects, obstacles, other moving agents as well as localization. Historically, these perception methods have first been performed with handcrafted features [Lowe, 2004, Dalal and Triggs, 2005, Moutarde et al., 2007, Labatut et al., 2007, Stanciulescu et al., 2009, Deschaud and Goulette, 2010, Zaklouta and Stanciulescu, 2014], but recently deep learning methods have shown significant improvements in image processing [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2016] and have progressively replaced every step in the Perception stack, be it object detection [Girshick, 2015, Redmon et al., 2016, Liu et al., 2016, Lin et al., 2017, Carion et al., 2020, Horváth et al., 2022], depth estimation [Dosovitskiy et al., 2015, Luo et al., 2016, Godard et al., 2017, Kendall et al., 2017, Liang et al., 2018, Ranftl et al., 2021], semantic segmentation [Long et al., 2015, Badrinarayanan et al., 2017, Isola et al., 2017, Zhu et al., 2017] or point cloud operations [Qi et al., 2017, Yang et al., 2018, Lang et al., 2019, Thomas et al., 2019].

This first round of semantic information is then post-processed using temporal information in order to perform higher-level reasoning tasks such as localization [Brubaker et al., 2013, Kendall et al., 2015, Moreau et al., 2022, Moreau et al., 2023] or object tracking [Welch et al., 1995, Wojke et al., 2017].

Prediction Once the Autonomous Vehicle (AV) has established a representation of the scene surrounding it, it needs to reason about the future evolution of this scene. If placed in the framework of a

Markov decision process [Bellman, 1957], the driving agent should be able to estimate its future state given its current state and its chosen action. However, while most of the state transition can be easily derived from kinematics, some parts may evolve independently of the agent's action, following a logic and goal of their own. More specifically, the other detected road agents will move in the future, and the AV must estimate their future location so it doesn't collide with them and is able to plan its own trajectory in the long term.

Planning Given the current scene representation and its predicted future evolution, the AV must design a plan so that it can achieve its main task: get to its destination. However this goal must be achieved while respecting some sub-tasks such as respect the traffic rules or avoid collisions. The planning stack therefore takes the observed state and its predicted future into consideration to construct a safe and efficient sequence of actions which it has evaluated to be the best according to the defined criteria. The first action of this sequence will then be applied through a control stack into steering and throttle inputs.

Impact of Prediction Trajectory prediction is the hinge step of the autonomous pipeline, and is therefore critical for the autonomous behavior. It is responsible for possibly its most important criterion: safety, as it is the corner stone to collision avoidance and future anticipation, which is essential for a smooth, energy-efficient, comfortable and reliable behavior.

About the advantages of splitting the pipeline vs end-to-end While some approaches tackle autonomous driving in a single end-to-end differentiable model to train direct driving behavior from sensory inputs [Chekroun et al., 2021], these methods sacrifice interpretability and theoretical guarantees for a gain that is mostly the appeal of a simplified architecture and a unified training on labels that are difficult and costly to collect. Furthermore the learning of these end-to-end models is cumbersome, as they require the model to understand a great deal of causality between very high level controls and very low-levels sensor inputs, and necessitate sophisticated methods not to suffer from distributional shift [Codevilla et al., 2019].

A branch of prediction research still approaches the prediction problem in a end-to-end manner, but separates the tasks inside the model architecture [Luo et al., 2018, Casas et al., 2018], starting directly from sensor data and processing perception and tracking before inferring future prediction. We chose to follow most of the literature and isolate the problem of trajectory prediction from supposedly perfect perception tracking, as provided in most trajectory datasets and benchmarks [Chang et al., 2019, Zhan et al., 2019]. Interestingly, even in this constrained framework some works [Ivanovic et al., 2022a, Weng et al., 2022, Ivanovic et al., 2022b] focus on the possible prior errors in the perception and argue these should be taken into account for better performance.

1.2 Problem statement: predicting the future of road agents

We set our problem in an intermediary representation, where nearby objects have been detected and tracked in order to extract their temporal sequences of coordinates, and the surrounding map has been localized and/or infered from sensor data. As described in Fig. 1.2, the prediction model takes the local map and the surrounding/target agents trajectories as inputs, in order to predict the future sequence of coordinates, at a regular frequency and up to a pre-defined time horizon (e.g. 5 seconds at 10Hz), for one or all of the nearby agents. As we will show later in this work, these inputs can be used either in their raw vectorized shape, or through a rasterization to obtain bird-eye view images.

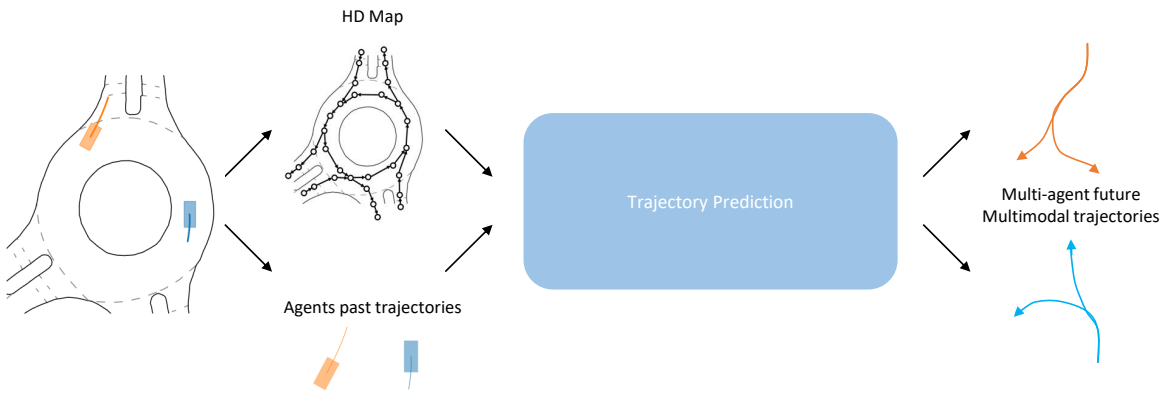


Figure 1.2: Inputs and Outputs for Trajectory Prediction

A trajectory prediction model needs to take into account a wide variety of factors. Its design may change according to the type of agent is has to predict. After encoding the temporal information sequences of each agent, it must be able to understand possible interactions between each of them, as well as traffic rules that may condition these interactions or add supplementary constraints. The surround map, with its road lanes, drivable area and intersections also has a very important impact on the trajectory of the vehicles.

Apart from raw performance, the motion forecasting pipeline must meet various challenges. In order to properly cover the different probable future scenarii, the prediction must be multimodal, such that every possible future is represented in a modality. Moreover multiple agents need to be predicted at the same time, fast enough to be implemented in a real-time system, and in a way that the scene-level forecast is consistent between each agents. These multiple simultaneous constraints also raise questions on how we should evaluate these prediction models, and if they should also be required to provide uncertainty estimates for their estimations.

1.3 Publications and communications

This PhD has been conducted in the center for robotics of Mines Paris, PSL University in the context of an industrial collaboration with Huawei Technologies France, in the Internet of Vehicles (IoV) team of Paris Research Center. The main publications and communications of this thesis can be synthesized as follows:

- T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, F. Moutarde. HOME: Heatmap Output for future Motion Estimationt. Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC 2021).
- T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, F. Moutarde. GOHOME: Graph-Oriented Heatmap Output for future Motion Estimation. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2022).

- T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, F. Moutarde. THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling. Proceedings of the International Conference on Learning Representations (ICLR 2022).

- T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, F. Moutarde. Uncertainty estimation for Cross-dataset performance in Trajectory prediction. Workshop on Fresh Perspectives on the Future of Autonomous Driving, IEEE International Conference on Robotics and Automation (ICRA 2022).

1.4 Outline

This thesis is laid out in seven chapters:

Chapter 1: Introduction. We give an overview of the autonomous driving pipeline to contextualize the task of trajectory prediction. We define the parameters of the motion forecasting problem and mention its main challenges.

Chapter 2: Non learning-based trajectory prediction. We review traditional existing motion estimation methods that are not based on machine learning. We present model families based on kinematics and physics, stochastic probability approaches, forecasts based on interactions and finally methods using intermediary representations in a hierarchical way, such as graph structures or maneuvers. We analyze the base principles of these foundational approaches and their shortcomings.

Chapter 3: Machine learning for trajectory forecasting. We provide an overview of existing machine learning techniques for motion estimation. We focus on each different axes that a complete driving model must incorporate. We organize along the natural process of the model, starting with the temporal encoding of trajectory sequences. We review agent interaction methods, which are primordial for both pedestrian and vehicle forecasting. We then observe the different ways of including map information, whether in the shape of an image or a graph. Finally, we list the possible output representations of these trajectory approaches, and weights both their respective advantages and limitations.

Chapter 4: Trajectory prediction with heatmaps: In this chapter we propose a heatmap-based representation for trajectory prediction outputs. We summarize the existing methods already leveraging similar representations, then present three different ways of generating these heatmaps. We benchmark the performance and speed of these proposals against each other and against the existing state-of-the-art. Finally we justify the advantages of such heatmap parametrizations in various ablation studies and experiments.

Chapter 5: Multi-agent consistent prediction: We tackle the lesser-explored problem of consistent multi-agent prediction. We recall the consistent scene-level problematic and the related works already dealing with this issue. We describe a novel post-processing multi-agent recombination module that reorders a marginal independent prediction into a joint coherent one. We evaluate our method on recent multi-agent challenges and demonstrate its added value on various experiments.

Chapter 6: Expanding the evaluation of trajectory prediction: Despite multiple existing challenges on trajectory prediction, these benchmarks usually use the same performance metrics that measure absolute similarity to the collected ground-truth data. However the qualities of a motion forecasting module may be more related to its robustness and interpretability than predicting an non-relevant pedestrian on the sidewalk or an opposite direction car to the nearest-centimeter. We explore more diverse ways

of evaluating a trajectory prediction model, by establishing cross-dataset performances for generalizability. We then show how uncertainty estimation can be leveraged in trajectory prediction, by presenting a heatmap-related uncertainty method along with a way of using this uncertainty to improve prediction quality. Finally, we study the calibration of the predicted heatmap itself.

Chapter 8: Conclusion Finally, we summarize our findings and open potential further exploration in the domain of trajectory prediction and its evaluation in the context of the full autonomous driving pipeline.

Chapter 2

Non learning-based trajectory prediction

Contents

This chapter reviews traditional methods for trajectory prediction, relying on simple principles such as physics, differentiability or causality. We refer to [Lefèvre et al., 2014] for a more extensive review of these classical approaches.

2.1 Kinematic models for motion estimation

2.1.1 Physical models

At its simplest, predicting the future position of a vehicle can simply be done using their speed v and heading θ, and approximating these as constants in the near future in the Constant Velocity (CV) model:

$$\mathbf{p_{t+\Delta t}} = \mathbf{p_t} + v\Delta t \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \tag{2.1}$$

As a first order approximation, this formula may be accurate in very short time horizons, but in longer term it does not represent the true physical behavior of an object that can accelerate or turn. It can therefore be refined with a Constant Turn Rate ω and Acceleration a (CTRA) model:

$$\begin{aligned} \mathbf{p_{t+\Delta t}} &= \mathbf{p_t} + v_t \Delta t \begin{pmatrix} \cos\theta_t \\ \sin\theta_t \end{pmatrix} \\ v_{t+\Delta_t} &= v_t + a\Delta t \\ \theta_{t+\Delta_t} &= \theta_t + \omega\Delta t \end{aligned} \tag{2.2}$$

As not all these quantities may be provided from the Perception module, a Kalman filter [Welch et al., 1995] can be used to derive these values from the past position sequence. These filter usually need hand-tuned hyperparameters for initial uncertainty values, however recent approaches have tried learning these values through back-propagation of the filter prediction errors [Jouaber et al., 2021].These models then intrinsically provide a prediction phase from their estimated state.

The physical model most commonly used in Kalman filters for autonomous driving is the bicycle model [Polack et al., 2017], which variates slightly from CTRA by assuming a constant steering value δ as illustrated in Fig. 2.1, and computing the turn rate from the steering, wheelbase L and speed:

$$\omega = v * \tan(\delta)/L \tag{2.3}$$

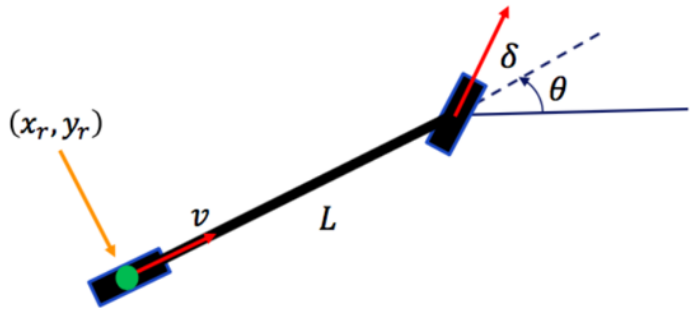
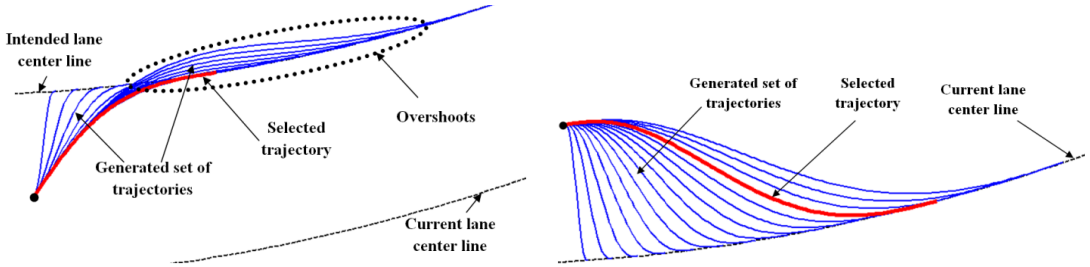


Figure 2.1: Bicycle model with constant steering angle δ and wheelbase L. Figure from [Ding, 2020]

[Schubert et al., 2008] provides a detailed survey and comparison of the different existing motion models in this category.

2.1.2 Time-series smoothness and extrapolation

One can consider the trajectory prediction task as an extrapolation problem, where the future points must be predicted from the historical data of a time series. [Wiest et al., 2012] fits Tchebychev polynomials to the trajectory of xy-positions, while [Houenou et al., 2013] fits a 5-degree polynomial for the lateral component and a 3-degree polynomial for the longitudinal component in Frenet coordinates as illustrated in Fig. 2.2. [Yi et al., 2015] also fits clothoids on trajectories, based on continually variable curvature.



(a) Change lane (b) Keep lane

Figure 2.2: Polynomial regression candidates from constraints on starting point (coordinates and derivative) and final point (derivatives). Figure from [Houenou et al., 2013].

2.2 Stochastic models

2.2.1 Hidden Markov Models

Hidden Markov Models (HMM) [Berndt and Dietmayer, 2009, Christopher, 2009, Firl et al., 2012] represent the car trajectory as a sequence of partial observations x derived from a hidden variable z. In the Markov chain formulation as illustrated in Fig. 2.4, z_t's probability distribution at each timestep t depends only on the previous variable z_{t-1}, and then the observed position x_t is conditioned on z_t only. This modelisation make HMMs very adapted for sequential data.

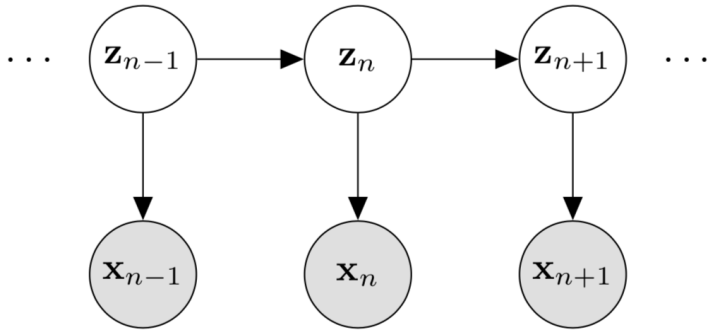


Figure 2.3: Markov chain with hidden variable z and observation x. Figure from [Gundersen, 2020]

The HMM models the joint probability of z and x:

$$p(x,z) = \prod_{t}^{T} p(x_t|z_t)p(z_t|z_{t-1}) \tag{2.4}$$

Where $p(x_t|z_t)$ is the emission probability and $p(y_t|z_{t-1})$ is the transition probability. It is therefore a generative model that represents the underlying probability disttribution.

2.2.2 Monte Carlo simulation

[Broadhurst et al., 2005, Eidehall and Petersson, 2006] use Monte Carlo sampling to simulate possible future paths from a sequence of uniform controls. Actions such as steering and acceleration are effectively sampled with respect to torque and friction limitations without any requirement for a linear physical model, while checking collisions to infer the most likely path avoiding collision.

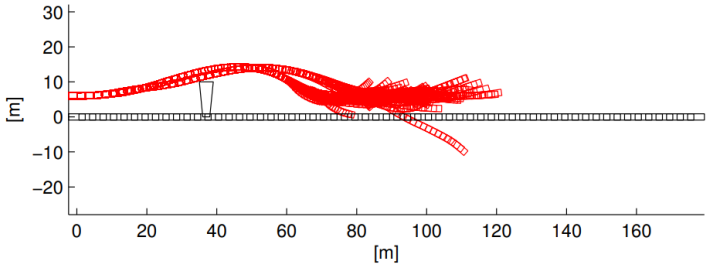


Figure 2.4: Monte Carlo sampling for collision avoidance from [Danielsson et al., 2007]

2.2.3 Conditional Random Fields

Conditional Random Fields differ from HMMs in that they are discriminative models for the conditional probability $p(z|x)$ only. They can also be applied for the prediction of latent events such as braking or overtaking [Ohn-Bar et al., 2015].

2.2.4 Bayesian networks

[Lefèvre et al., 2011] uses a Bayesian network to assess the probability of each possible lane while leveraging the topology of the intersection through a direct graph representing the dependencies between different variables. Compared to HMMs, Bayesian Networks are less focused on the temporal dependencies and more on relationships between different vaiables such as chosen maneuver, turn signal and so on.

2.3 Interaction models

2.3.1 Social forces

Relationship with pedestrian trajectory prediction Despite being fairly similar tasks in their final aim, pedestrian and car motion forecasting differ a lot in methodology and nature. Pedestrians mostly evolve in an unconstrained environment, with their motion being mainly influenced by other pedestrians in crowds. Cars on the other hand follow a very strict framework with lanes and traffic rules, where other

agents remain important for collision avoidance and space occupancy, but are not the main factors of the target car's movement.

Works such as [Helbing and Molnar, 1995] predict pedestrian movements through the application of social forces from other pedestrians and obstacles as illustrated in Fig. 2.5. Similar methods have been applied through many-particle systems and fluid dynamics on vehicle traffic, but these remain on a macroscopic level, and are not sufficient to efficiently predict individual car movements accurately.

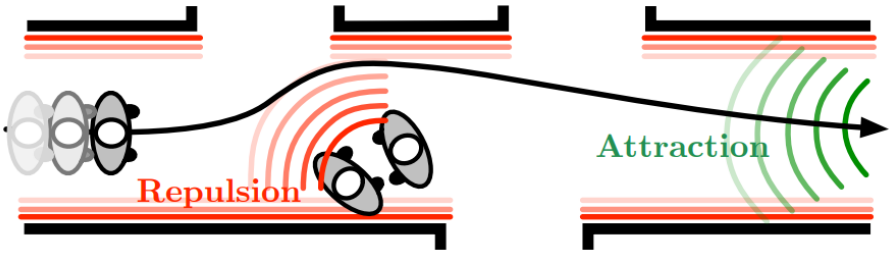


Figure 2.5: Social forces for pedestrian motion forecasting. Figure from [Rudenko et al., 2020]

2.3.2 Driver models

Relationship with simulation Since simulating a realistic driving behavior comes down to generating a set of possible trajectories that the driving agent could follow in real life, simulations could be considered as picking one of the possible trajectory for the agent and using it as internal planning. Once again, at simulation time the destination of the agent should be known, but most simple simulation models don't consider destinations anyway, focusing on car interactions.

The most commonly used behavior model is the Intelligent Driver Model (IDM) [Treiber et al., 2000], that directly controls the acceleration of the car according to the following equation:

$$\dot{v} = a[1 - (v/v_0)^{\delta} - (s^*(v, \Delta v)/s)^2] \tag{2.5}$$

With $\Delta v = v - v_l$ the speed of approach to the lead vehicle, Δ an exponent usually set to 4 and where v is compared to the desired speed v_0 and s to the desired distance s^*:

$$s^*(v, \delta v) = s_0 + max(0, vT + \frac{v\Delta v}{2\sqrt{ab}})$$

with T the time-gap to the leading vehicle, s_0 the minimum spacing, a the maximum acceleration and b the comfortable breaking deceleration. $s_0 + vT$ is here an equilibrium term, and $\frac{v\Delta v}{2\sqrt{ab}}$ a dynamical term that implements an "intelligent" braking strategy, where the kinematic acceleration necessary for safety is self-regulating towards the comfort deceleration.

[Liebner et al., 2013] adapts this driver model from simulation into behavior prediction by defining prior clusters of driver behaviors from data and estimation their a posteriori probabilities of theses hypotheses given the past trajectory.

This model, like most others, mostly transcribe a simple car following model for traffic simulation and collision avoidance, and covers only specific scenarios such as highways. It is more adaptive to its surrounding that physical models, but trades-off this adaptability against realism.

2.3.3 Game-theoretic approaches

The prediction/planning of a future sequence of actions for multiple driving agents can also be considered as solving a game where each agent wants to maximize its own reward while sometimes also considering the others reward [Schwarting et al., 2019] as illustrated in Fig. 2.6. The solution is then obtained by solving an interactive game through a game tree [Bahram et al., 2015] or a Stackelberg game [Li et al., 2017] where the leader vehicle first chooses an action, and then the following vehicle, and so on, in order to reach a Nash equilibrium where no player can gain by changing his action alone.

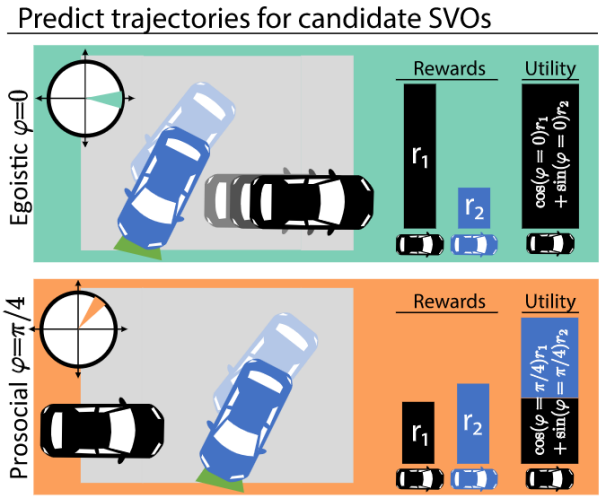


Figure 2.6: Different driving behaviors from game-theory solutions, depending on whether the agent only considers his own reward (egoistic) or also the other's (prosocial). Figure from [Schwarting et al., 2019]

2.4 Hierarchical prediction

In order to represent the various granularities of a realistic trajectory, the movement can be decomposed into intermediary steps, which can be either planning steps towards a destination or predefined operations derived from real human behavior.

2.4.1 Graph search

Relationship with planning The task of trajectory prediction shares many similarities with planning. Indeed, predicting the future of one surrounding car is almost the same as planning for this car as a driving agent. Main differences would be that, while a driving agent knows its destination for planning, this goal of arrival must me estimated during prediction, or not follow any assumption on a specific goal and cover all possible destinations.

The graph can represent continuous space as in Hybrid A* [Dolgov et al., 2008] that extends the heuristic-based graph search method A* to a continuous state search by storing continuous states in reached discrete cells, or rapidly exploring random trees (RRT) [LaValle et al., 1998] that constructs their own nodes iteratively in the actions space, guaranteeing kinematic feasibility. With the surrounding road map usually available as a graph, the future path of a car can also be estimated by exploring the

reachable nodes in the map. Speed profile can then be considered afterwards, or during the graph search by using a spatio-temporal graph [Rowold et al., 2022] as illustrated in Fig 2.7.

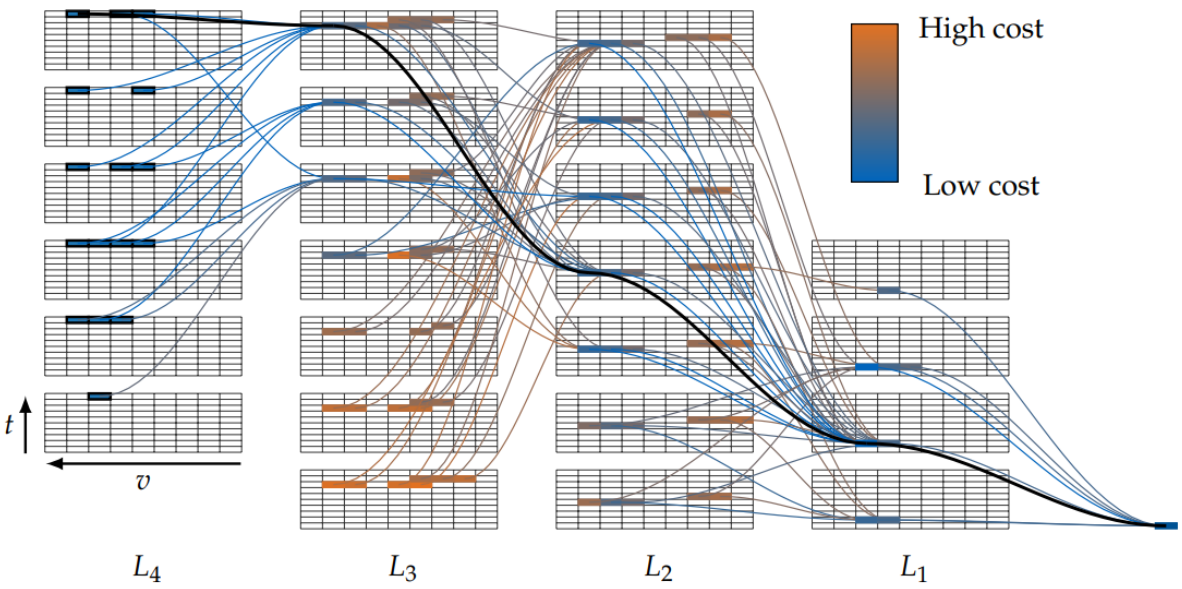


Figure 2.7: Spatio-temporal graph search fro trajectory planning. Figure from [Rowold et al., 2022]

2.4.2 Maneuver-based prediction

In order to reduce the prediction space complexity, some works propose to restrict the possibilities to a discrete set of maneuvers .

Anchors

The past of a trajectory can be used to find its nearest neighbor in a cluster of anchor trajectories [Atev et al., 2010] and then use the representative trajectory of the cluster to estimate the future [Vasquez and Fraichard, 2004, Hu et al., 2006, Hermes et al., 2009, Joseph et al., 2011]. If each cluster provides a weight or likelihood relative to the trajectory history, then the infered path can also be a weighted average of all the closest clusters. [Chang et al., 2019] also uses Nearest Neighbors as a baseline for its trajectory prediction benchmark, and explores the use of either cartesian or curvilinear coordinates as query. However these remain pretty limited to handle all variations present in traffic or new intersections layouts.

Maneuver intention

In order to model more abstract and general concepts, some works chose to model discrete intentions such as change lane, give the way, turn right ... etc. [Greene et al., 2011] tracks multiple maneuver hypotheses through a Kalman filter and selects the most likely hypothesis a posteriori. [Streubel and Hoffmann, 2014] uses Hidden Markov Models to estimate the likelihood of the vehicle going straight or turning. However most of these maneuver estimation methods rely solely on the target agent trajectory and therefore do not take interactions into account. [Käfer et al., 2010, Lawitzky et al., 2013] add a penalty on colliding trajectories during matching for interacting vehicles, but require to constrain the collision checking to simple pairs of vehicles, or to consider an exponential number of combinations among all agents as illustrated in Fig. 3.2.

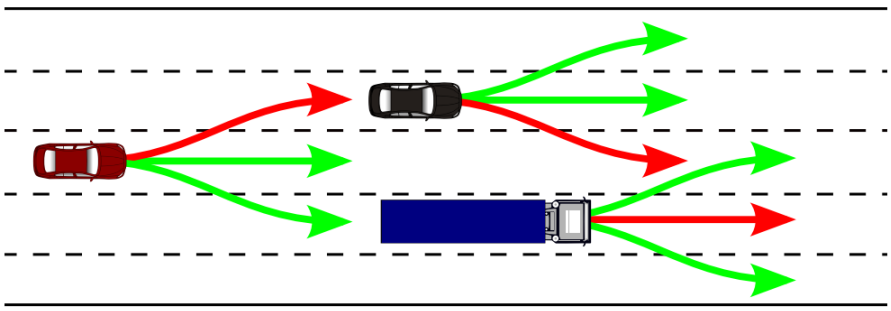


Figure 2.8: Possible maneuver combination between multiple agents on a highway. Figure from [Lawitzky et al., 2013]

Regardless of their methodology, most of these non-learning based methods fail to capture the require adaptability and flexibility to adapt to various road layouts and interactions outside of a predefined applicative scope.

Chapter 3

Machine learning for trajectory forecasting

Contents

We now present and analyze the main machine-learning based methods for trajectory forecasting.

Machine learning, and most notably deep learning, has recently brought an unequalled ability for models to adapt to any observed new scenarios without resorting to an ever-expending array of predefined fixed behaviors. We review here existing deep learning methods applied to trajectory prediction by categorizing the four main necessary steps for reliable trajectory prediction:

- Temporal encoding, as the main clues reside in the history of observations of the target vehicle in the past few seconds
- Agent interaction since our main motive is avoiding collisions with other agents, and many speed and position restrictions come directly from the distribution of surrounding agents
- Map context: even prior to interactions, our driving is greatly constrained by the drivable area and the local traffic rules we must obey
- Output representation: once all the necessary information has been processed through the previous step, even with theoretical perfect information we must choose the shape we want our model to output this prediction, as each choice will have its own impact on the coverage and possible uses of the prediction

Machine learning methods other than neural networks, such as Support Vector Machines [Aoude et al., 2011] or Random Forests [Völz et al., 2016, Schlechtriemen et al., 2015] have also been applied to predict future behaviors, but since most of these applications could also be treated by neural networks, and given their recently demonstrated superior performance when provided sufficient amounts of data, we will focus now on deep learning methods, onto which the research community has been expanding a great lot lately. Amongst the methods presented in the previous chapter, some could also be considered as learning methods, like HMMs where the transition probabilities from one state to another are learned from data.

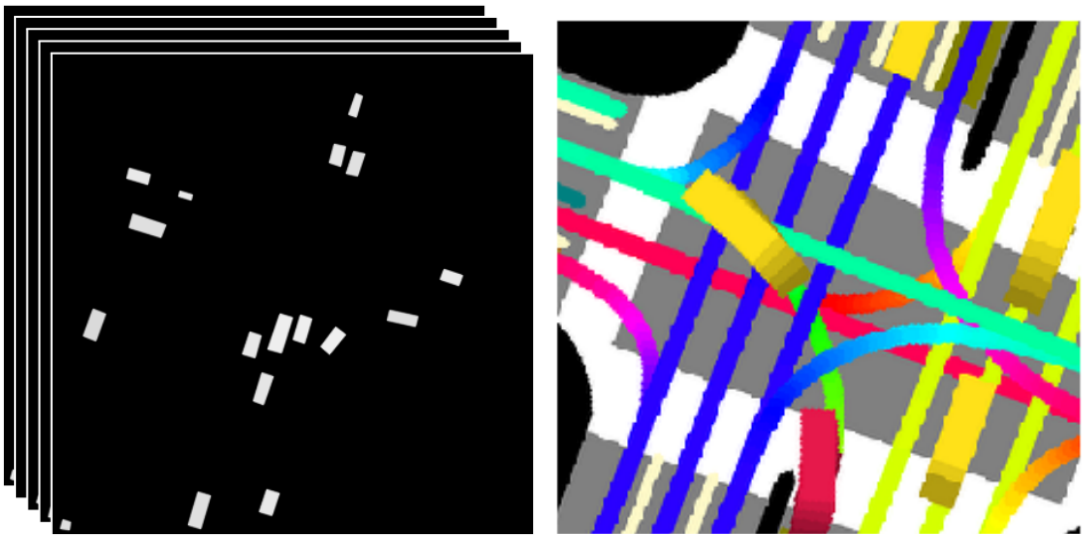
3.1 Temporal encoding

3.1.1 Multi-Layer Perceptron

[Yoon and Kum, 2016] applies a Multi-Layer Perceptron (MLP) on a sequence of past states to predict target lane and lane change parameters. While seemingly simple and disregarding possible adaptations to take the time dimension into account, this method actually works relatively well for short sequences, as it allows the model to have a complete overview of the whole sequence with positional information for every step. However, as the size of the sequence increases, so does the size of the model parameters, and the network becomes too slow and unable to generalize to the data. [Hu et al., 2018] also uses a MLP to predict the parameters of a Gaussian Mixture Model, but only applies it to the current timestep to respect the Markov assumption. [Bahari and Alahi, 2019] also explore feed-forward neural networks compared to other architectures, and find that they can achieve similar performance while leveraging a faster response time, while [Lenz et al., 2017] demonstrates that they perform better in closed-loop evaluation.

3.1.2 Convolution neural network (CNN)

In deep learning, a convolutional layer [LeCun et al., 1995] is a sliding filter that extracts features based on a learned kernel. Their design makes then translation-invariant and parameter-efficient, notably adapted for images [Krizhevsky et al., 2017] and time-series.



(a) Concatenated layers [Bansal et al., 2018] (b) Faded effect [Djuric et al., 2018]

Figure 3.1: Image inputs for CNN-based models, with different time evolution representation.

2D Convolutions In a wide family of models that take as inputs an image representing the semantic information in bird-eye view, time evolution must also be incorporated to represent moving neighbor agents and the history of the target vehicle. This time change can usually be depicted in two ways. First, keeping in mind the input image is not restricted to 3 RGB channels, a separate channel can be used for each timestep showing the position of each agent at this time [Bansal et al., 2018, Hong et al., 2019], as represented in Fig. 3.1a. However this can become quite cumbersome with a high number of timesteps, and faster inference time with less model parameters can be achieved by representing this evolution in a simple RGB image [Djuric et al., 2018, Cui et al., 2019, Chai et al., 2019, Phan-Minh et al., 2020] where moving vehicles are shown as a fading line as in Fig. 3.1b, with the furthest timesteps being the most transparent.

1D Convolutions Convolutions can also be leveraged in a vectorized setting, where the input is the sequence of coordinates for successive timesteps. In that case, the input can be shaped as a (T, D) tensor, with T the number of historical timesteps and D the feature size ($D = 2$ for xy at minimum). The T dimension can then be used to apply a 1D convolution as a sliding window along the time dimension in order to extract higher level features such as speed or acceleration [Mercat et al., 2020, Liang et al., 2020].

3.1.3 Recurrent neural network (RNN)

A RNN applies the same cell recurrently to each time features one timestep after another. The cell has a memory storage that is updated at each application, while choosing which information from the previous memory to retain. Common reccurent cell architectures are GRU [Cho et al., 2014, Chung

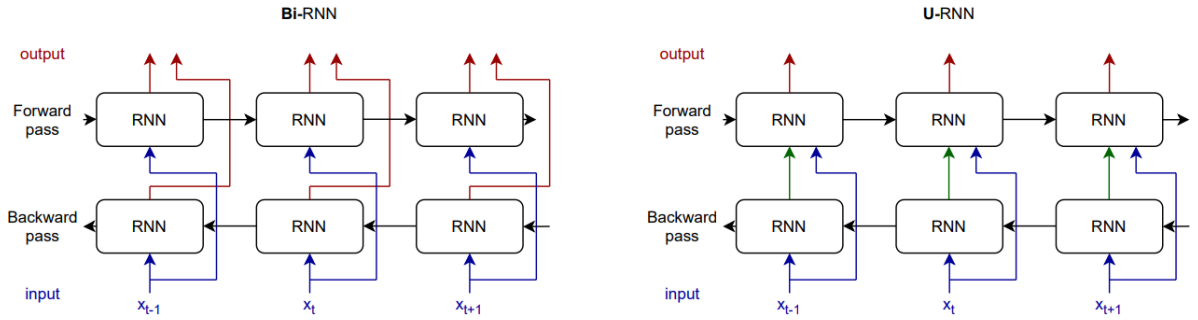
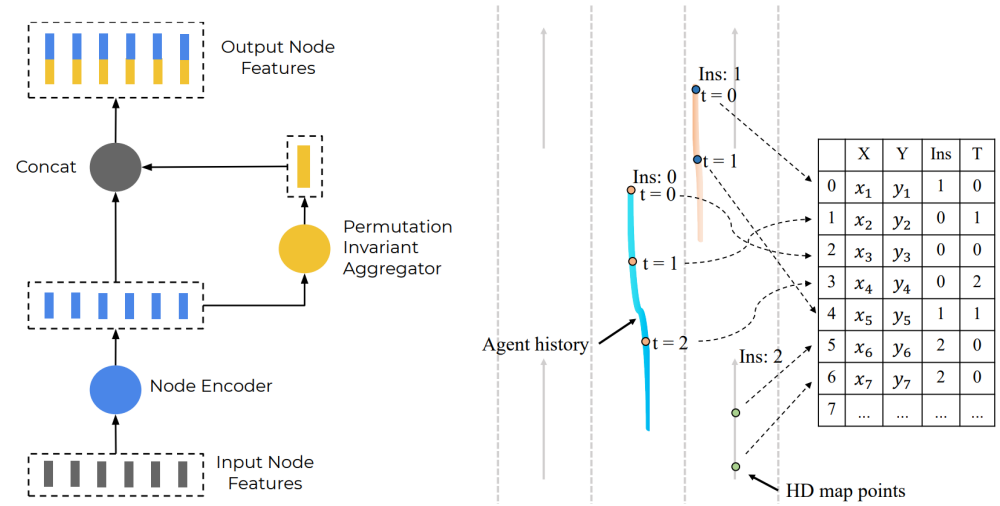


Figure 3.2: Bidirectional RNNs applied on sequential input using either the symmetrical (Bi-RNN) or Assymmetrical (U-RNN) layout. Figure from [Rozenberg et al., 2021]

et al., 2014] and LTSM [Hochreiter and Schmidhuber, 1997], which leverage gated activations to adapt the part of memory retained from the previous timesteps, and the part that will be updated with the new one, overall helping with longer term memory. [Altché and de La Fortelle, 2017] applies an LSTM to highway trajectory prediction, while [Khosroshahi et al., 2016, Deo and Trivedi, 2018b] use them to classify maneuvers on highways, and [Phillips et al., 2017] on intersections. [Messaoud et al., 2019] also use them for each separate vehicle encoding, sharing their weights across agents, while [Mercat et al., 2020] combines 1D CNNs for feature extraction and LSTMs for sequence encoding. [Alahi et al., 2016] also applies LSTMs to pedestrian prediction. LSTMs can be stacked on top of each other as in [Xin et al., 2018], sometimes in a bi-directional way [Xue et al., 2017, Yao et al., 2021] where one of the pass is backward to capture different sequential information that can be also used to enrich the subsequent forward pass [Rozenberg et al., 2021].

3.1.4 PointNet



(a) PoinNet operation [Gao et al., 2020] (b) Point groups [Ye et al., 2021]

Figure 3.3: Illustrated point cloud operation and instantiation

Other works choose to consider the sequence of positions as a group of points, that can then be encoded using common point cloud methods such as PointNet [Qi et al., 2017]. In short, these models

alternate aggregating pointcloud information across points, usually by averaging their features, spreading this aggregated information by concatenating it to each point, and encoding the new information through a multi-layer perceptron as illustrated in Fig. 3.3a. Fig. 3.3b highlights how points can be grouped by either entities, time intervals or spatial regions. VectorNet [Gao et al., 2020] and subsequent works [Zhao et al., 2021, Gu et al., 2021] encode the coordinate sequence of each agent and lane through a PointNet, while TPCN [Ye et al., 2021] also variate time interval lengths.

3.1.5 Attention and Transformers

Attention works like a dictionary query between tokens: we want each token to select the other tokens of importance and use their only information. To do so three tensors are generated from the tokens, through a linear projection: the queries Q, the keys K and the values V. In the case of time encoding, all these tensors would be derived from the timesteps T and therefore have shape (T, d), with d the feature size. This case where all queries, keys and values come from the same tensor is called self attention. When queries come from a different source than the context keys and values, the operation is called cross-attention, which we will see later in this chapter. An attention score α is then computed through the scaled dot-product between queries and keys, onto which a softmax is applied for normalization:

$$\alpha = \text{softmax}(\frac{Q.K^T}{d}) \tag{3.1}$$

The attention update to the tokens is then the weighted sum of the values weighted by the attention score:

$$X = \alpha V \tag{3.2}$$

Since the whole operation is permutation-invariant, a positional embedding is usually added to each token to discriminate between them (in our case this would be an embedding of the timestamp).

An advantage from this technique shared with CNNs is that while accounting for all possible timesteps it remains invariant to their total number since the final context is a weighted average summed up to a single feature vector, and the only requirement is for the keys and values to have the same first dimension T, which they have by design. An attention-based temporal encoder could therefore in theory handle history of any sizes.

Following great demonstrated efficienty in the Natural Language Processing field [Vaswani et al., 2017], attention has been increasingly applied in the trajectory prediction framework as well, replacing RNNs. With attention, one timestep can access context information from any other timestep indiscriminately, not suffering from short term memory problems like RNNs or restricted receptive fields like CNNs. Stacked together in alternation with feed-forward layers, these attention models are called transformers. Such transformers have been applied to both pedestrian [Yu et al., 2020, Giuliari et al., 2021, Yuan et al., 2021] and vehicle [Girgis et al., 2021, Amirloo et al., 2022] trajectory prediction.

[Ngiam et al., 2021, Zhou et al., 2022] add an additional artificial token as an extra timestep to summarize all the temporal information into one single token, while [Nayakanti et al., 2022] introduces a whole new set of latent tokens to reduce the quadratic complexity of each timestep attending every other timestep.

3.2 Agent interactions

While driving, we keep paying attention to other road participants in order to avoid collision with them, respect traffic priority or follow their speed. This interactive behavior needs to be represented in our prediction model.

3.2.1 Fixed closest and neighboring vehicles

The difficulty in leveraging neighbor vehicles information comes from their varying number and relative position. The road can be either empty or filled with traffic, and most traditional fully-connected deep learning layers require a fixed size input, hence the need for specific rules as to which neighbor car to select. [Phillips et al., 2017, Hu et al., 2018, Sadeghian et al., 2019] uses the top-k closest vehicles according to the Euclidean distance, while [Altché and de La Fortelle, 2017, Lenz et al., 2017] predefined areas of interests based on lanes where vehicles will be considered if inside the area as illustrated in Fig. 3.4.

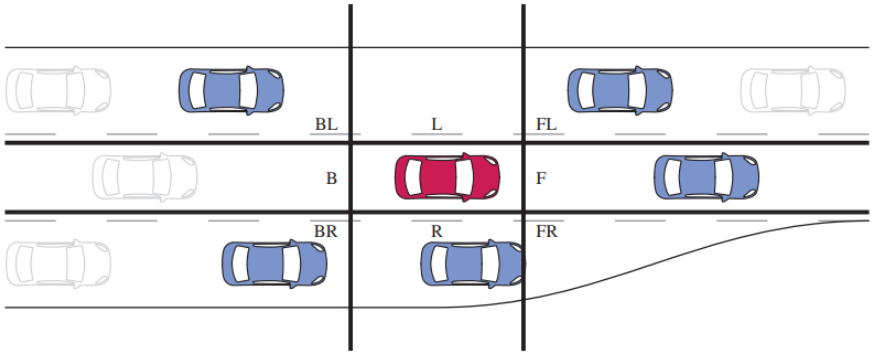


Figure 3.4: Fixed considerd neighbors based on lanes. Figure from [Lenz et al., 2017]

3.2.2 Shared aggregated representation

In order to deal with the restricted number of agents, [Alahi et al., 2016] pools the hidden state from neighbors inside a spatial grid by summing them and feeds the resulting pooled grid to the target agent through concatenation and a fully-connected layer to all grid positions.

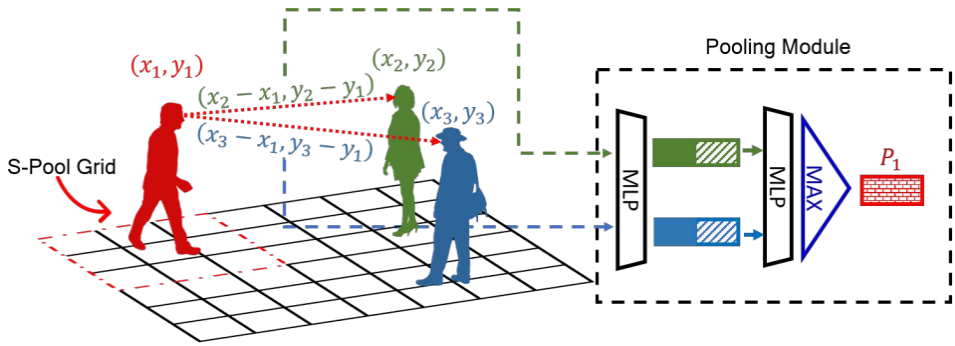


Figure 3.5: Social pooling for learned sparse interactions from [Gupta et al., 2018]

[Deo and Trivedi, 2018a] use a similar grid but apply convolutional layers to preserve the spatial structure. [Gupta et al., 2018] simplify this operation by replacing it with a MLP followed by MaxPooling depicted in Fig. 3.5 for better speed and capability for far-away interactions.

3.2.3 Graph Neural Networks (GNN)

The different agents in a scene can be seen as the nodes N of a graph (N, E) illustrated in Fig. 3.6, where the edges E represent the interactions between road users. The edges can be connected according to distance between agents as in [Ivanovic and Pavone, 2019, Salzmann et al., 2020] or fully-connected as in [Casas et al., 2020], given the relatively low number of agents.

A Graph Neural Network (GNN) can then be used to aggregate information between nodes across edges and update node features given their connected edges. For example, [Ivanovic and Pavone, 2019, Salzmann et al., 2020] aggregate connected node information with a sum operation, while [Casas et al., 2020] uses message passing and computes a feature vector for each edge from concatenated neighbor states through a MLP and then applies MaxPooling for every edge of a node. [Liang et al., 2020] also uses a MLP on concatenated edge features, but with sum aggregation.

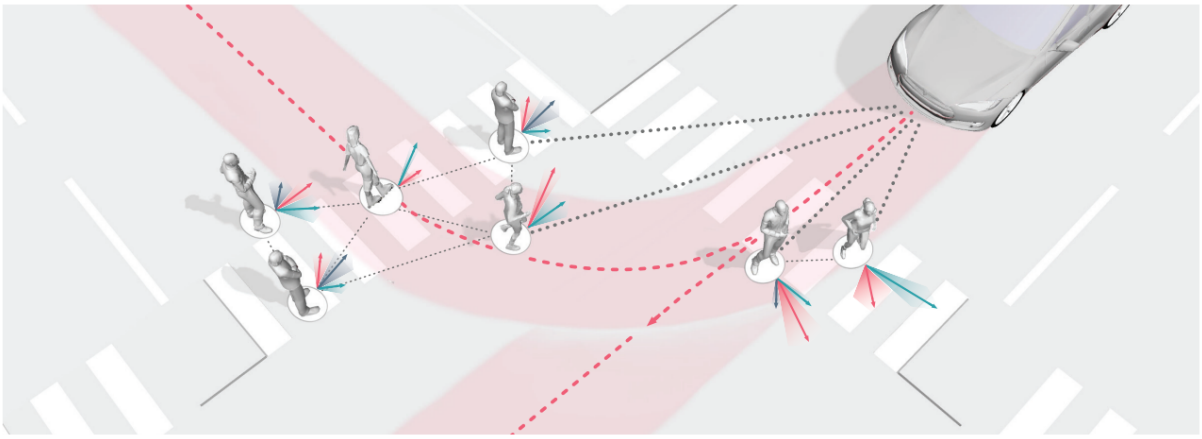


Figure 3.6: Interaction graph between road agents [Salzmann et al., 2020]

GNNs allow the model parameters not to be dependent on the number of agents while retaining agent-specific information without global aggregation as in 3.2.2 that sacrifices representative power.

3.2.4 Attention

In the interaction framework, attention can be seen as a specific instance of a fully-connected GNN [Gao et al., 2020]. The aggregation operator summarizing all neighbor agent features is then a learned attention operation as described in 3.1.5, where the query is the target agent and the keys and values are its neighbors. Distributed to every agent in the scene, this comes down to self-attention again, where every agent is looking at every other agent. As illustrated in Fig. 3.7, attention allows the target agent to select which neighbor is more important to predict its future trajectory, for example the car right in front.

[Messaoud et al., 2019, Mercat et al., 2020, Messaoud et al., 2020] apply a multi-head attention layer on top of LSTM-encoded agent features to model their interactions, enabling non local-constrained interactions for variable number of agents. Subsequent works generalize this by stacking multiple attention

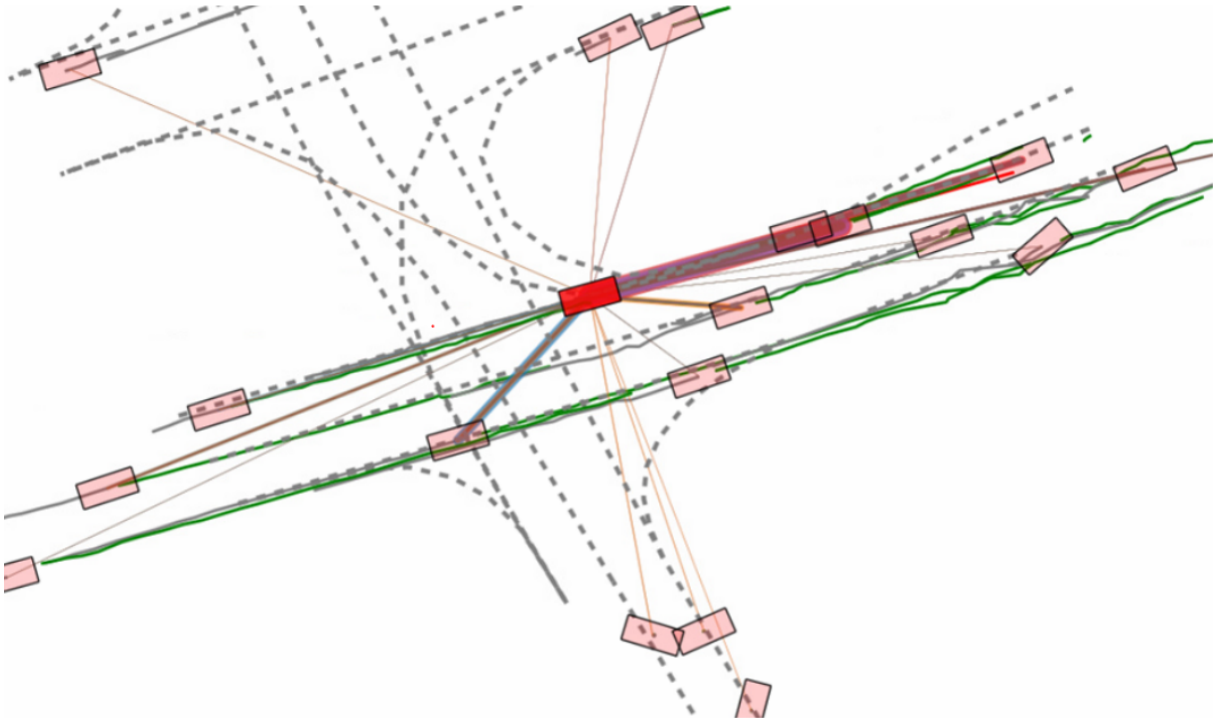


Figure 3.7: Attention between the target center agent and its neighbors. Attention is represented by straight limes, with different color according to the attention head and width scaled with attention score. Figure from [Mercat, 2021]

layers in a Transformer architecture [Liu et al., 2021b, Ngiam et al., 2021, Girgis et al., 2021, Huang et al., 2022].

Blurring the line between time and agent dimensions Recent works have tried to generalize time encoding and interaction encoding by considering that each agent at a given timestep could be a separate token interacting with other agents at other timesteps. [Yuan et al., 2021] argues that making this connection directly increases the model capability, but [Ngiam et al., 2021] factorizes the time and social attentions while using the same representation and alternates both operations, in order to improve the model complexity, while [Nayakanti et al., 2022] also evaluates the use of latent queries to further trade-off efficiency and quality.

3.3 Map context

As already mentioned in Sec. 2.3.1, while interaction encoding is a shared common part of both pedestrian and vehicle prediction, static context encoding is what differentiates them, along with physical constraints. Vehicles are more restricted by their surrounding layout, while humans evolve in a much less constrained environment.

While initial trajectory datasets were usually focused on straight highway sections with a fixed number of lanes [Colyar and Halkias, 2006, Colyar and Halkias, 2007], where there was no need for additional map information, recent urban datasets [Chang et al., 2019, Zhan et al., 2019, Caesar et al., 2020, Houston et al., 2021, Ettinger et al., 2021] contain data from intersections, roundabouts and insertions where drivable area context is needed.

Note: map context and agent interactions share similar techniques, and recently some unifying frameworks have been attempted, but remain treated/considered differently in most works, and represent different concepts: static/dynamic, environment/others.

3.3.1 Image CNN

A straightforward way is to represent the local map surrounding the agent as an image, the same way a GPS device would display it for a human as already mentioned in 3.1.2. The resulting image is a bird-eye view (BEV) of the road layout within a certain range around the vehicle, usually oriented according to the agent's yaw. This representation can contain various shapes and layers such as drivable area, centerlines optionally colored according to their heading, stop lines / traffic lights, crosswalks or satellite pictures.

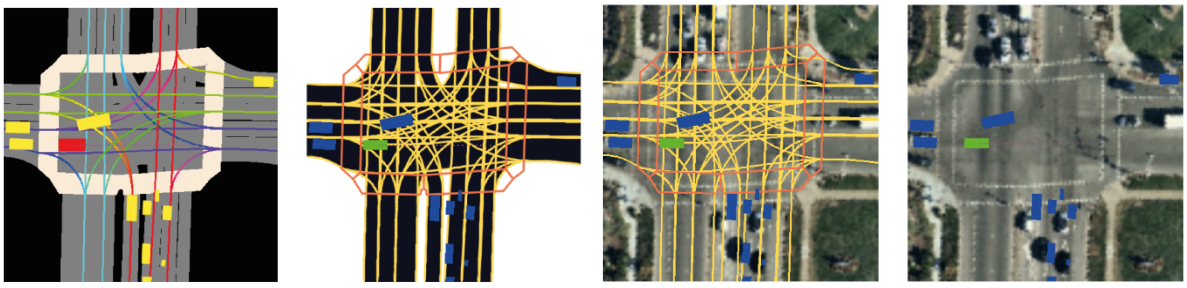


Figure 3.8: Available BEV representations in Lyft dataset [Houston et al., 2021]

ChauffeurNet [Bansal et al., 2018], Rules of the Road [Hong et al., 2019] and [Cui et al., 2019] use a CNN encoder on top of this rasterized input as their main model architecture, while DESIRE [Lee et al., 2017], Sophie [Sadeghian et al., 2019] and Trajectron++ [Salzmann et al., 2020] consider it more as an additional feature channel to complete the sequential trajectory encoding. Multipath [Chai et al., 2019] and TrafficSim [Suo et al., 2021] both encode a global non-agent centric wide map into a feature grid, and then crop agent-specific sub-grids for each agent.

CNNs allow to preserve the 2D spatial structure of the map, however to be merged with sequential coordinate inputs they are often flattened and lose most of their spatial information.

3.3.2 GNN

Most recent datasets encode their provided maps in a format similar to Lanelet2 [Poggenhans et al., 2018], where the road is segmented into lanelets. Each lanelet represents a lane continuous piece of a certain length, and is defined as a sequence of points. Lanelets are then encoded together into a graph where each lanelet possibly has a predecessor, successor, left and right neighbor. This connectivity graph contains a lot of information for the vehicle reachable paths, but is lost if transformed into an image. LaneGCN [Liang et al., 2020] therefore directly leverages the map graph as an input by encoding each individual lane segment and applying graph convolutions to incorporate connectivity information as shown in Fig. 3.9. Each agent is then connected to its local lane segments according to a distance threshhold.

LaneRCNN [Zeng et al., 2021] goes further by encoding each agent trajectory features into a separate road graph and then computing agent interactions by merging the lane graphs themselves, in order to

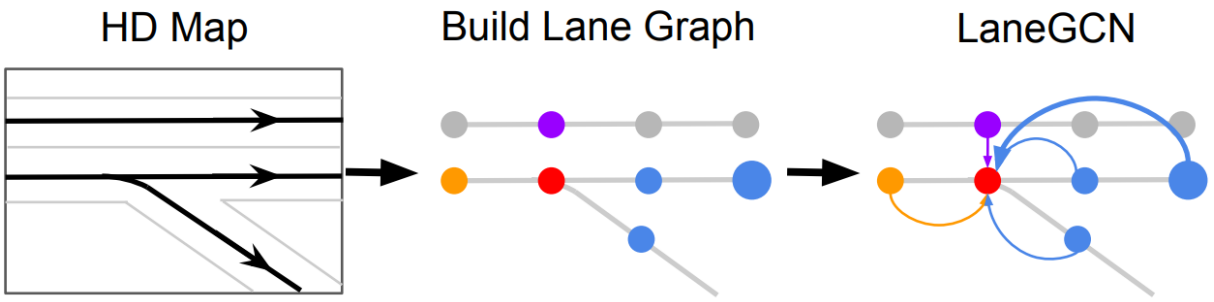


Figure 3.9: Map graph building and connectivity encoding [Liang et al., 2020]

localize the interactions. PGP [Deo et al., 2022] learns a policy for graph traversal that is then used to extract possible routes roll-outs from which trajectories are infered and scored.

3.3.3 Attention

Driving agents can query the useful information amongst the surrounding road segments by using cross-attention previoulsy described in Sec. 3.1.5. Usually a distinction is done with the attention made to other interactive agents by using separate alternative layers as in mmTransformer [Liu et al., 2021b] or SceneTransformer [Ngiam et al., 2021], but some work attempt unifying frameworks as in VectorNet [Gao et al., 2020] where global attention is applied to both static and dynamic elements indiscriminately, further re-used in TnT [Zhao et al., 2021] and DenseTNT [Gu et al., 2021].

3.4 Output representation

Once the context information has been correctly encoded comes the question of how to represent the output prediction, and how to generate it. Obviously one of the main factors for this decision is the intended use after the prediction, however many architectural choices and trade-offs remain to be done depending on the qualities we hope to find in the forecast, whether it is precision, diversity, coverage or feasibility.

Multimodal prediction While the trajectory history may contain clues about the agent future movement, some future decisions depend on hidden variables that only the driver knows, such as real destination and itinerary preferences, as illustrated in Fig. 3.10. Lane changes and giving or taking the right of way also depend on driver styles which depend on each individual, and can vary with time. Finally, the exact controls over time can never be predicted with perfect precision, leading to a growing uncertainty over time and a spread of possible coordinates at each timestep. The possible future of a road agent is therefore multimodal, meaning that a prediction should contain multiple possibilities, optionally with a matching probability score.

3.4.1 Coordinates

Here we will mostly tackle methods that try to directly output their prediction as a sequence of coordinates, opposite to predicting a distribution from which the user can then sample to obtain a point. VAEs

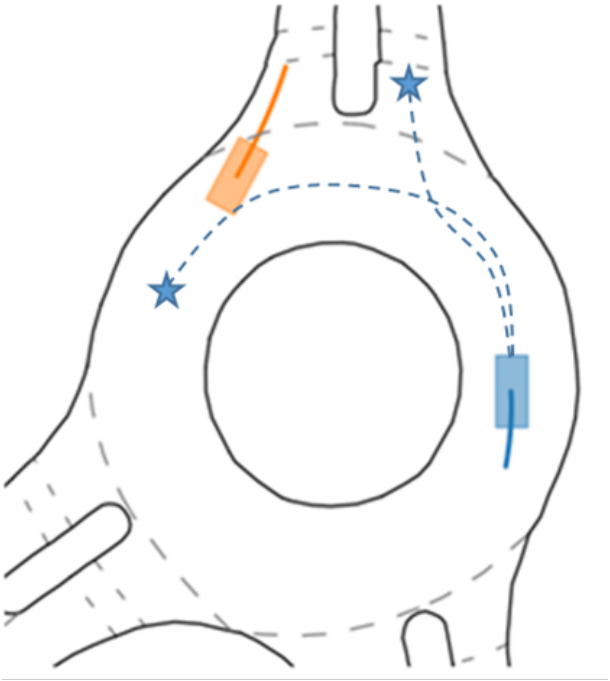


Figure 3.10: Ambiguous multiple possible futures from a single historical context

are a bit of an hybrid case in that regard, the training and inferences systematically require the model to output coordinates, but an intermediate step first predicts a distribution which is effectively sampled.

3.4.1.1 Scalar coordinates

[Altché and de La Fortelle, 2017] directly predicts the future absolute coordinates along the highway, however the large range of possible coordinates require them to output an intermediary speed value and then integrate it for longitudinal coordinates (lateral coordinates remain in a restricted range in the highway referential). In order to avoid the high range problem, [Casas et al., 2018, Bansal et al., 2018] predict relative coordinates to the initial vehicle position as a single Maximum A Posteriori (MAP) prediction.

For multimodality, most works [Cui et al., 2019, Liang et al., 2020] directly predict a set of coordinates, one for each modality. In order to train these, a Winner-Take-All (WTA) loss is commonly used, where only the prediction closest to the ground truth is trained. In order to represent aleatoric uncertainty, some methods use Gaussian [Mercat et al., 2020] or Laplace [Ngiam et al., 2021] bivariates illustrated in Fig. 3.11c where the mean is simply the predicted point coordinates, and the variance represents the estimated error of the model, which usually grows with time and speed.

Controls Trajectron++ [Salzmann et al., 2020] and Multipath++ [Varadarajan et al., 2022] predict acceleration and steering controls then integrates them to obtain physically feasible trajectories. Multipath++ observes a slight reduction in performance doing so, but choose this trade-off for better closeness to reality. Using such derivatives also enable better interpolation, as does predicting polynomial coefficients in PLOP [Buhet et al., 2021].

Auto-regressive models Instead of directly predicting a set of coordinates for each timestep, some approaches take advantages of the sequential nature of trajectories and make each subsequent timestep

depend on the previous one through a recurrent architecture [Alahi et al., 2016, Salzmann et al., 2020, Tang and Salakhutdinov, 2019, Rhinehart et al., 2019] as shown in Fig. 3.11b. At training time, these use ground truth coordinates instead of predicted ones for the conditioning over previous timesteps. However [Casas et al., 2020] compares these methods and notices compounding error problems and distributional shifts between training and test data, requiring added noise during training for better generalization.

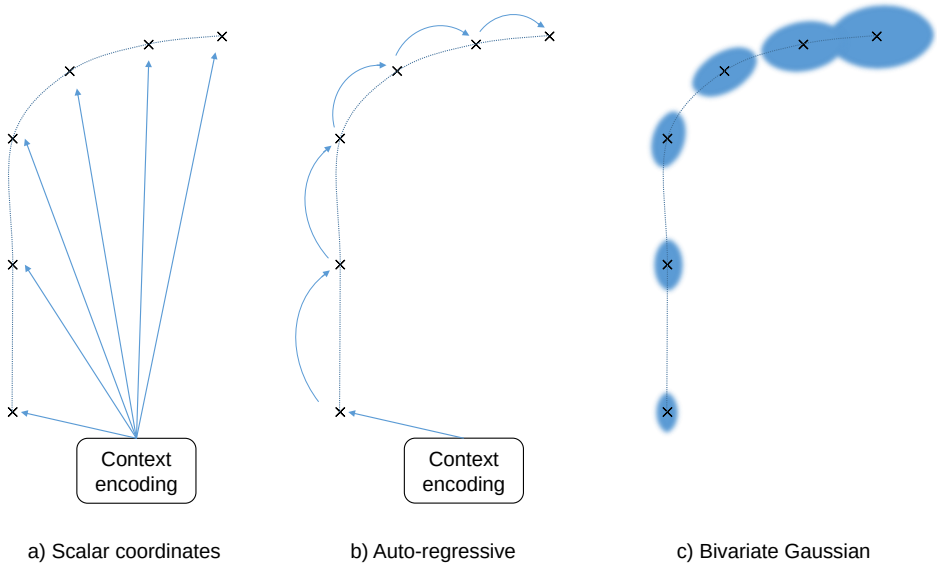


Figure 3.11: Various parametrization for trajectory output

3.4.1.2 Anchors

Models predicting set of multimodal coordinates may suffer from mode collapse if all modalities are trained at the same time, or from poor sample efficiency if only one modality is trained at a time. Furthermore, the learned modalities may lack diversity as they will focus around the means of the most likely positions over the training set. In order to help with diversity, Multipath [Chai et al., 2019] clusters training trajectories into a fixed set of predefined anchors. The model is then tasked with classifying the sample into the corresponding closest anchors and estimating regression coefficients to fit the anchor better to the real trajectory. Covernet [Phan-Minh et al., 2020] proposes a similar anchor-based prediction but adds dynamic trajectory anchors adapted to the current agent dynamics. PRANK [Biktairov et al., 2020] takes an approach more related to ranking techniques and metric learning by scoring the prediction into a large dictionary of up to 2M stored trajectories.

Later approaches leverage the prior provided by the surrounding map to select probable anchors. TnT [Zhao et al., 2021] generates end-goal proposals from the road layout and feeds then to the model to estimate probabilities and regression coeficients. LaneRCNN [Zeng et al., 2021] also leverages the map graph into which the agent features are incorporated to generate probability scores at each segment as illustrated in Fig. 3.12. PRIME [Song et al., 2022] browses reachable paths from the connected map graph and yields feasible trajectories for each of them before using a learning-based evaluator to score them. These approaches are also mentioned as goal-based methods, as they discretize possible trajectories starting from their inferred goal and then estimating the full intermediary motion.

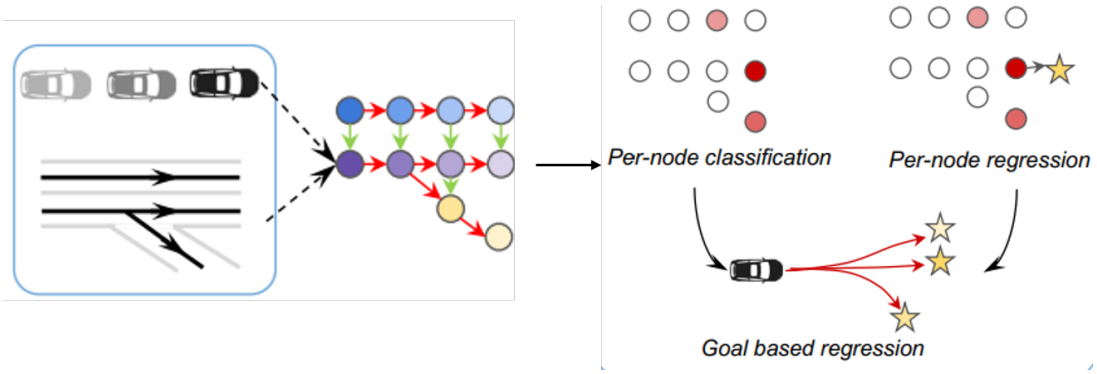


Figure 3.12: Goal-based trajectory prediction leveraging map prior [Zeng et al., 2021]

More recent approaches opt to leverage learned anchors to benefit from their diversity without having to make empirical biased decisions about where the anchors should come from. Multipath++ [Varadarajan et al., 2022] and Autobots [Girgis et al., 2021] therefore stored learned latent embeddings which correspond each to a different modality, making these a bit of a hybrid approach between anchor-based and regular direct coordinate prediction, since the model output remain a set of predicted coordinates, and the latent embeddings cannot be visualized for interpretability.

3.4.1.3 Sampling-based trajectory prediction

Sampling-based methods have an approach to multimodality different from the above described works. Instead of predicting a fixed number of coordinates or probabilities, they leverage an intrinsic random sampling that enables them to yield as many predictions as the amount of times they are sampled. Therefore, each inference only produces one trajectory, but these inferences are not deterministic, and will therefore generate a different likely trajectory when forwarded a second time.

Variational Auto-Encoders (VAE)

Variational auto-encoders [Kingma and Welling, 2013, Sohn et al., 2015] hypothesize that the future y is conditioned on a latent variable z. The parameters of the prior distribution $p(z|x)$ are sampled to obtain z values, which are then decoded by a generator $q(y|z,x)$. p and q are approximated by neural networks trained to match the resulting predicted y with the ground truth. This process is illustrated in Fig. 3.13b in contrast to the usual determinic process. During training a posterior distribution $p(z,y,x)$ is also learned and used to sample z with the knowledge of the future y in order to disambiguate the multiple possible modalities, and the posterior $p(z,y,x)$ and prior $p(z|x)$ are then constrained to have similar distributions. PRECOG [Rhinehart et al., 2019] samples a specific latent for each timestep, but [Tang and Salakhutdinov, 2019] argue that the sampled variable should be consistent at every timestep to represent a tractable intention that should be the same across all next timesteps. Trajectron [Ivanovic and Pavone, 2019] and MFP [Tang and Salakhutdinov, 2019] use a discrete latent variable to represent distinct possible futures. ILVM [Casas et al., 2020] uses a continuous latent distribution to encapsulate all possible variations in the latent, and then decodes a deterministic estimated future conditioned on the latent. Autobots [Girgis et al., 2021] learn discrete seed parameters as latent variables and use attention to decode all latents in parallel in a single forward pass.

A main drawback from these generative methods is that they don't provide probability estimation nor

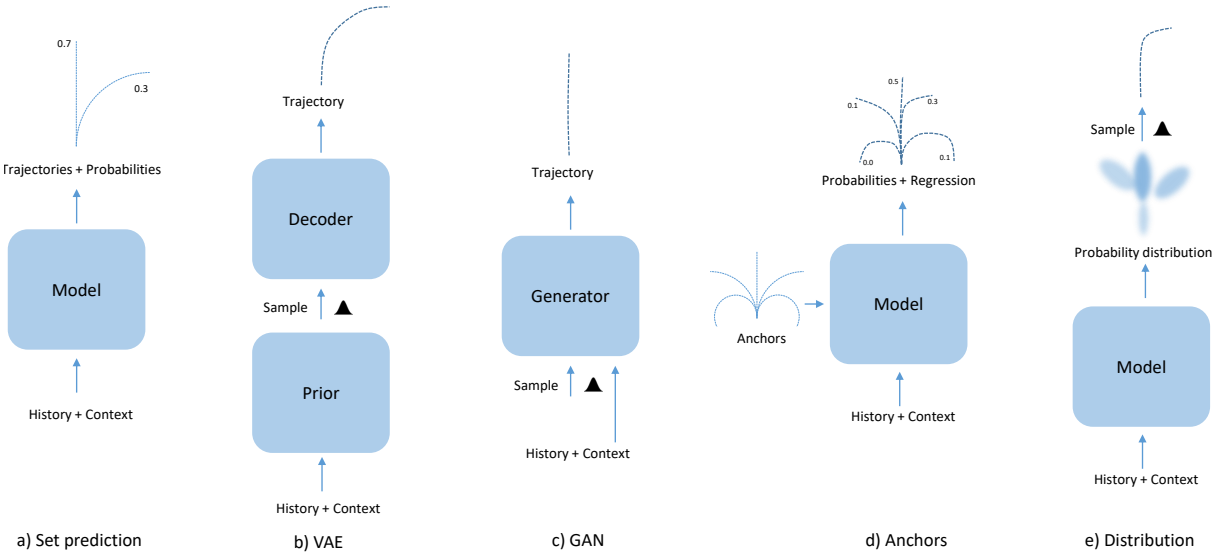


Figure 3.13: Different model architecture for multimodal and variational prediction

guarantee that the sampled modalities represent a good coverage of all the future possibilities. Lookout [Cui et al., 2021] tries to remediate to these by training a sampler to improve the coverage of the sampled modalities and a scorer to estimate their probabilities.

Generative Adversarial Networks

GANs [Goodfellow et al., 2014] are another generative techniques that train a generator to yield realistic trajectories from a sampled noise input. The generator is trained through a discriminator comparing the generated distribution to the ground truth one. The training is called adversarial because the discriminator learns to separate the scores attributed to 'fake' or 'real' trajectories, while the generator tries to trick the disciminator by bringing the two distributions close to each other, resulting in opposed losses. Compared to VAEs, the latent is directly sampled first without any context as illustrated in Fig. 3.13c, instead of being derived from a learned conditional prior. SocialGAN [Gupta et al., 2018], SoPhie [Sadeghian et al., 2019] and MATF [Zhao et al., 2019] leverage a GAN to produce multiple diverse trajectories.

Gaussian mixtures Other methods like [Ivanovic and Pavone, 2019] do not directly output coordinates, but a Gaussian mixture from which the final position has to be sampled as in Fig 3.13e. This resulting distribution has the added benefice of having an interpretable and visualizable form in the cartesian space, opposed to the initial or latent distributions of other variational methods which do not have physical equivalents and cannot be interpreted as such.

3.4.2 Maneuvers

Some works predict intention or maneuvers as a supplement or instead of direct trajectory coordinates. These maneuvers can be lane changes [Xing et al., 2019, Wang et al., 2021b], turns or decisions relative to interactions [Phillips et al., 2017]. However their definitions remain ambiguous as one intention can defer to multiple different trajectories, and these frameworks remain empirical to each paper, with a lack of a common distinct framework. Maneuvers can however be utilized as an intermediary representation

for better interpretability. IntentNet [Casas et al., 2018] produces trajectory and decision estimation in parallel, while[Deo and Trivedi, 2018b] and [Mersch et al., 2021] leverage maneuver classification as a discrete intermediary latent to predict maneuver-specific trajectories

3.5 Conclusion and motivation for our thesis

As we have described above, most existing learning methods for trajectory prediction suffer from multiple pitfalls. First and foremost, these methods need to be multimodal, but the provided ground-truth data is unimodal in nature, and the commonly used multi-head structure from multiple future prediction doesn't rely on any concrete differentiator between each head, leading to heavy dependency on the random initialization. The existing solutions of using anchors often rely on broad and bias-sensitive assumptions and may exclude important eventualities, such as the agent deviating from the drivable area. Our goal is therefore to design a prediction model that can be easily trained and produce a naturally multimodal and unconstrained prediction.

Chapter 4

Trajectory prediction with heatmaps

Contents

Limitations of the methods presented in previous chapters call for another output parametrization. Outputting an intermediary probability distribution would then enable the user to use the sampling method of their choice, depending on the intended use and performance criteria. Gaussian Mixture Models (GMMs) outputs [Ivanovic and Pavone, 2019, Messaoud et al., 2021] provide an analytical distribution but constrain the prediction to a pre-defined number of modes with a very specific ellipsoidal shape. If more modes are required than there are goals, stochastic sampling will have to be used which lacks guarantees. Moreover, the shape of the road, especially during turns, doesn't match ellipsoidal shapes. There is therefore a need for a parametrization of an analytical, unconstrained probability distribution.

This chapter presents our first contribution in this thesis: a grid-based heatmap probability distribution that enables an unconstrained bias-free multimodal prediction. We first describe what these heatmaps are and how they can be used within a trajectory prediction framework. We then propose and compare three different ways of generating and learning these heatmaps. Finally, we explain and justify the reasons for using such methods instead of the other common state-of-the-art approaches.

4.1 Heatmap representation

Following many recent works [Zhao et al., 2021, Gu et al., 2021, Zeng et al., 2021], we focus on predicting the last point of the trajectory y_T, and argue that the full trajectory can then be interpolated unequivocally from this endpoint. This allows us to simplify the problem to modelize only the probability distribution of this last point. We represent the possible futures distribution by a 2D probability heatmap that gives an unconstrained approximation of the probability of the agent position. Such a heatmap, with corresponding context and trajectories, is illustrated in Fig. 4.1.

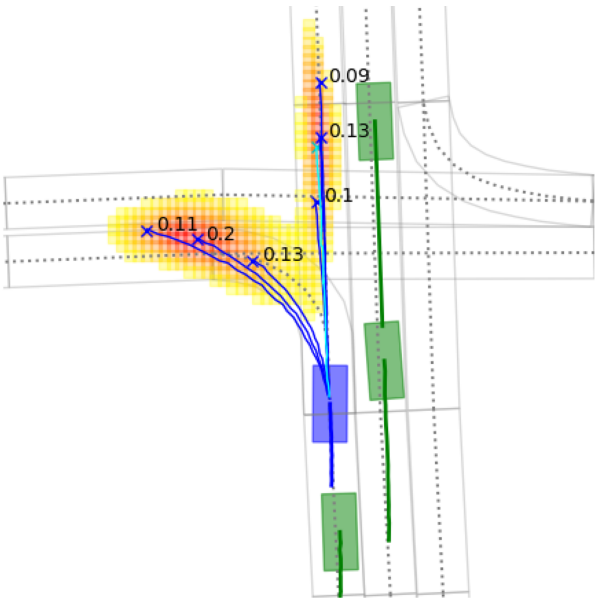


Figure 4.1: Heatmap output representing the final trajectory point prediction

This heatmap is represented as a squared image and it naturally accommodates for multimodal predictions where each pixel represent a possible future position of the target agent. It also enables to fully

describe the future uncertainty in a probability distribution, without having to choose its modes or means. In other words, we reformulate the usual regression task of trajectory prediction into a classification task, where each grid pixel is a possible class.

4.2 Related works

Other works have already leveraged heatmaps or grids as outputs. [Kim et al., 2017] models the future prediction on a highay as a rectangular occupancy grid but solely uses LSTMs and doesn't apply CNNs to exploit the natural local relationships of the grid. Consequently their grid size remains limited (36, 21) with a coarse resolution (0.875 x 5) m^2 per pixel. Moreover, as is common to many of these related grid approaches, they predict the occupancy of all scene participants in one single grid, making individual prediction and analysis impossible. [Sadat et al., 2020] also proposes an instance-free occupancy prediction focused on semantic classes with CNNs this time around, which enables them to predict a much wider and finer 140 x 80 m^2 grid of 0.2 meter / pixel resolution. MP3 [Casas et al., 2021] expands on this latter work by adding occupancy flow to provide velocity estimates. [Ridel et al., 2020] predicts individual grid predictions for each vehicle through ConvLSTMs layers [Shi et al., 2015] but they focus on pedestrian trajectories and predict one heatmap for each timestep which results in memory restrictions and long trainings (about 5 days per epoch). Furthermore, their sampling is learned and doesn't directly leverage the properties of the occupancy grids. [Mangalam et al., 2021] also focuses on pedestrians and models both long-term goals and intermediary waypoints in the two-dimensional space as an image for pedestrian trajectory forecasting, combined with random sampling and KMeans clustering [Lloyd, 1982, MacQueen, 1967].

4.3 Decoding a heatmap into a trajectory

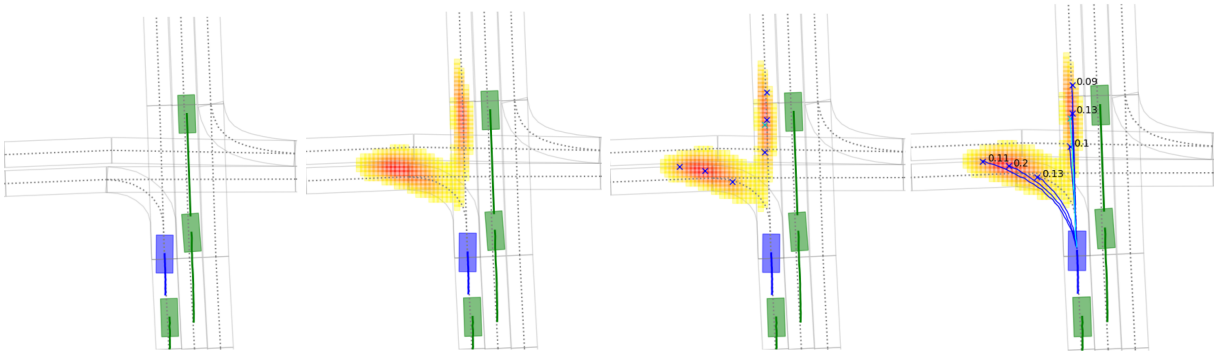


Figure 4.2: Pipeline for trajectory prediction through probability heatmap. a) Context map, target agent (blue) and neighbor (green) trajectories are given as input to the network. b) Heatmap output of the network. c) Sampled final points. d) Trajectories are built for each final point

We illustrate the process of using a heatmap for trajectory prediction in Fig. 4.2. The model first predicts a spatial heatmap given the historical past trajectory, other agents and the local map context. In a second step, we sample from the heatmap a finite number of possible future locations with the

possibility to choose which metric we want to optimize without retraining the model. Finally, we build the full trajectories based on the past history and conditioned on the sampled final points.

We will further describe how to obtain this heatmap in the later parts of this section, but for completeness we will start first with the post-processing of this probability grid and how it is used, as this is common to all the different decoding methods we will later study, and put better into context the use case of the heatmap.

4.3.1 Endpoint sampling

Our aim is here to sample the probability heatmap in order to optimize the performance metric of our choice. In most datasets such as Argoverse [Chang et al., 2019] and NuScenes [Caesar et al., 2020], two main metrics are used for the final predicted point: MissRate (MR) and Final Displacement Error (FDE). MissRate corresponds to the percentage of prediction being farther than a certain threshold to the ground truth, and FDE is simply the mean of l_2 distance between the prediction and the ground truth. When the output is multimodal, with k predictions, minimum Final Displacement Error minFDE_k and Miss Rate over the k predictions MR_k are used.

4.3.1.1 Optimizing Miss Rate

We design a sampling method in order to optimize the Miss Rate between the predicted modalities and the ground truth. A case is defined as missed if the ground truth is further than 2m from the prediction. For a given area A, the probability of the ground truth Y being in this area is equal to the integral of the probability distribution p under this area.

$$P(Y \in A) = \int_{x \in A} p(x)dx \tag{4.1}$$

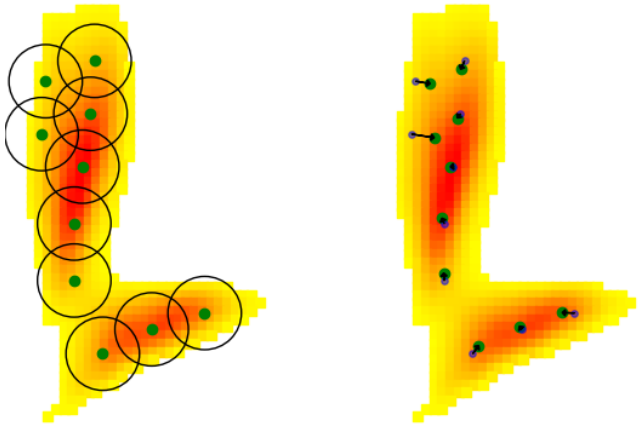
Therefore, for k predictions, given a 2D probability distribution, the sampling minimizing the expected MR is the one maximizing the integral of the future probability distribution under the area defined as 2m radius circles around the k predictions:

$$E(\mathbb{1}_{\min_k \|c_k - Y\| > 2}) = 1 - \sum_k \int_{\|c_k - x\| < 2} p(x)dx \tag{4.2}$$

We therefore process in a greedy way as described in Algo. 1, and iteratively select the location with the highest integrated probability value in its 2m circle. Once we obtain such a point, we set to zero the heatmap values under the defined circle and move on to selecting the next point with the same method.

The result is illustrated in Fig. 4.3a. We see that each sampled point can be surrounded by a circle of radius 2m that barely overlaps with other circles. Each point is sampled almost equidistant to the others, as setting the probability under previous points to zero sets a very strict limit to the minimum distance between points.

For implementation, we process the covered area for each point using a convolution layer with kernel weights fixed so to approximate a 2m circle. In practice, we don't actually use a radius of 2 meters, but a 1.8 meters one as we found out it to yield better performance. We also upscale the heatmap to 0.25 x



(a) MR sampling (b) FDE sampling

Figure 4.3: Illustration of heatmap sampling methods

Algorithm 1: MR Sampling Algorithm

input: Probability map p(x)
K number of predictions
R threshhold for Miss Rate
for $k = 1..K$ **do**
  Find c_k maximizing $\int_{\|c_k - x\| < R} p(x)dx$
  Set $p(x) = 0$ for all x such that $\|c_k - x\| < R$
end

0.25 m^2 per pixel with bilinear interpolation to have a more refined prediction location.

4.3.1.2 Optimizing Final Displacement Error

We inspire ourselves from KMeans [Lloyd, 1982, MacQueen, 1967] to optimize minFDE_k. The image output can be represented as a discrete probability distribution (x_i, p_i) where x_i represents the pixel centers and p_i the associated probability value. Optimizing the Final Displacement Error over k predictions means finding k centroids c_k that minimize the following quantity:

$$minimize_c \sum_i p_i \|c - x_i\| \tag{4.3}$$

To do so we design our sampling algorithm for FDE optimization detailed in Algo. 2.

We replace the classic weighted average $\sum_i p_i x_i$ for each centroid c_k by $\sum_i \frac{p_i}{d_i^k} x_i$ where d_i^k is the distance between point x_i and centroid c_k to be more robust to outliers and take into account the optimisation of l_2 norm instead of its square.

In essence, we update each prediction as a weighted average of its local neighborhood in a radius of 3m. The coefficient $\frac{m_i}{d_i^k}$, with m_i the distance between point x_i and its closest centroid allows for flexible

Algorithm 2: FDE Sampling Algorithm

input: Set of points x_i with probability weight p_i
L number of iterations to run the algorithm
Initialization of K centroids c_k
for $l = 1..L$ **do**
  Compute d_i^k the matrix of distance of point x_i to each centroid c_k
  Compute m_i the distance of point x_i to the closest centroid c_k
  for $k = 1..K$ **do**
    Compute new centroid coordinates :
    $c_k = \frac{1}{N} \sum_i \mathbb{1}_{d_i^k <= 3} \frac{p_i}{d_i^k} \frac{m_i}{d_i^k} x_i$
    with $N = \sum_i \mathbb{1}_{d_i^k <= 3} \frac{p_i}{d_i^k} \frac{m_i}{d_i^k}$
  end
end

partition boundaries compared to KMeans (where we would use $\mathbb{1}_{d_i^k <= m_i}$ instead): when x_i is in the partition of prediction k, its value is 1, while when it's outside it decreases, so as to be 0 when at the exact position of another prediction k', where it could never be improved by a displacement of k.

We initialize the centroids with the results of the Miss Rate optimization algorithm and use the number of iterations L as a parameter to tune the trade-off between Miss Rate and FDE: when L is zero, Miss Rate is optimized while when L increases MR is sacrificed to get better FDE. The output of the algorithm is illustrated in Fig. 4.3b, where it can be observed that centroids are brought closer together, sacrificing total coverage but getting closer to areas with high probabilities to reduce the expected distance. Results of this trade-off are illustrated further in Sec. 4.3.1.3, where we show in Fig. 4.4 that every iteration of Algo. 2 diminishes minFDE_6 and increases MR_6.

4.3.1.3 Experiments on endpoint sampling

We highlight our sampling results in Tab 4.1 and compare them to other possible sampling strategies: we try ranking pixels by probability and select them in decreasing order while removing overlapping pixels that are closer than a 1.8m radius following a classic Non-Maximum Suppression method. We also try KMeans as is used in [Mangalam et al., 2021].

Table 4.1: Ablation study on trajectory sampling
(Argoverse validation set)

Bottleneck	K=1		K=6	
	minFDE	MR	minFDE	MR
Pixel ranking with NMS	3.07	51.0	1.21	10.7
KMeans	3.06	51.6	1.23	9.3
Ours (MR)	3.02	50.7	1.28	**6.8**
Ours (FDE L=6)	**3.01**	**50.5**	**1.16**	7.4

We show in Fig. 4.4 the results of our trade-off between MR_6 and FDE_6 on the Argoverse test set thanks to the parameter L of Algo. 2. We also include points for the other top 10 methods of the

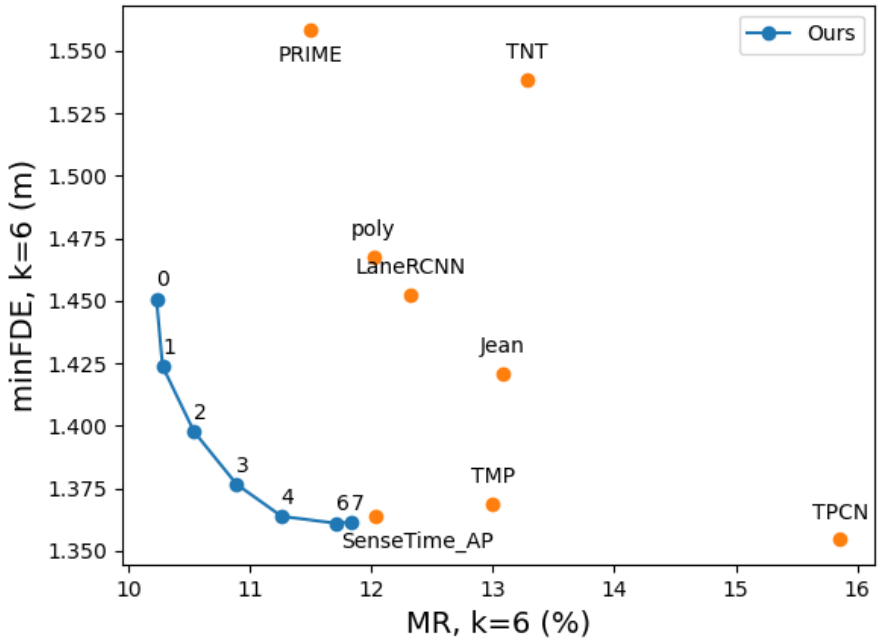


Figure 4.4: FDE_6 - MR_6 trade-off. Lower-left is better. Points of the curve (blue) are obtained increasing number of iteration L of Algorithm 2 from 0 to 7. Points for other top-10 leaderboard methods are also included (orange).

leaderboard at the date of 1st March 2021 for comparison. Our method reaches the best possible MR_6, and allows to improve FDE_6 to second-best while still being first in MR_6 (fourth curve point obtained with $L = 4$)

4.3.2 Full trajectory regression

For each sampled target a separate model is used to generate full trajectories conditioned on the final endpoint. Any non-learning-based planning method could do this task, but in our case we use a simple MLP illustrated in Fig. 4.5.

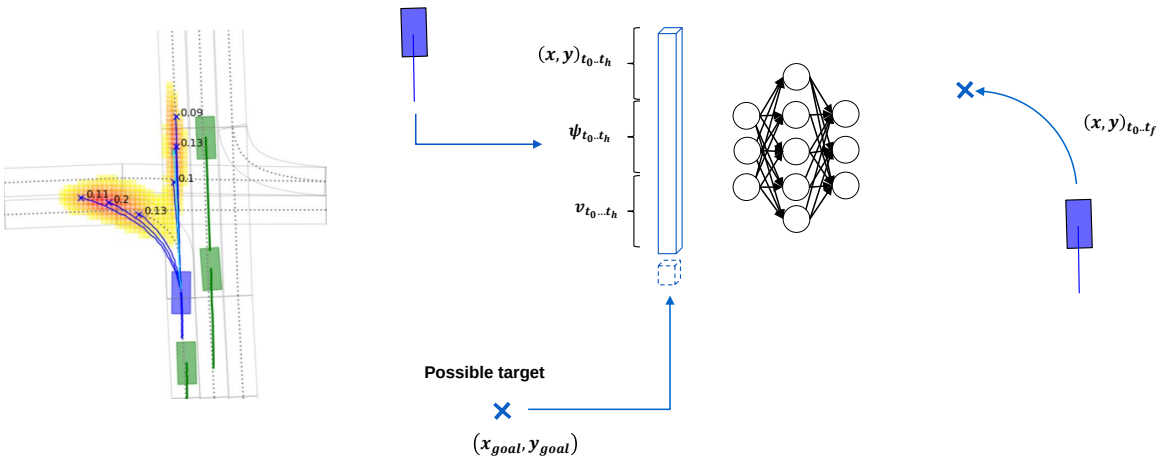


Figure 4.5: Goal conditional trajectory regression

This model applies a fully-connected layer to encode the target agent history into a vector of 32 features, which is then concatenated with the (x, y) coordinates of the target future location. Another fully-connected layer is then applied to obtain a 64 feature vector, which is then transformed through a last fully-connected layer to a set of locations representing the intermediate position of the agent in the time frame $[[1, T]]$. The probability of a trajectory is computed as the integral of the probability heatmap under the circle of radius 2m around the end point of the trajectory.

During training, this regression model is trained with the ground-truth endpoint to regress the ground-truth full trajectory in a teacher-forcing way.

4.4 How to generate such a heatmap ?

4.4.1 Convolutional neural network: HOME (Heatmap Output for Motion Estimation

The most straightforward way to yield image heatmaps is to use a CNN model that also takes images as inputs. However we also want to leverage the granular information present in the past sequence of coordinates, and therefore we design a hybrid architecture that merges rasterized and scalar information.

4.4.1.1 Map and past trajectory encoding

The local context is available as a High Definition Map centered on the target agent. We rasterize the HD-Map in 5 semantic channels: drivable area, lane boundaries and directed center-lines with their headings encoded using HSV on 3 channels. We also add the target agent trajectory as a moving rectangle on 20 history channels and the other agents history on 20 more channels. The final input is a (224, 224, 45) image with a 0.5 x 0.5 m² resolution per pixel. This image is processed by a classic CNN model alternating convolutional layers and max-pooling for downscaling to obtain a (14, 14, 512) encoding E_{raster} as illustrated in the top-left part of Fig. 4.6.

The scalar history of the agents is also taken as input to the model as a list of 2D coordinates. Missing timesteps are padded with zeros and a binary mask indicating if padding was applied or not is concatenated to the trajectory, as well as the timestamps for each step, so that we obtain a $(H, 4)$ input for each agent. Each agent trajectory goes through a 1D convolutional layer followed by a UGRU [Rozenberg et al., 2021] recurrent layer. The weights are shared for all agents except the target agent.

4.4.1.2 Inter-agent attention for interaction

Similar to [Mercat et al., 2020, Messaoud et al., 2020], we use attention [Vaswani et al., 2017] to model agent interaction. A query vector is generated for the target agent, while key and value vectors are created for the other actors. The normalized dot product of query and keys creates an attention map from the target agent to the other agent, then used to pool their value features into a context vector. The context vector is then added to the target vehicle feature vector through a residual connection followed by LayerNormalization [Ba et al., 2016]. The obtained trajectory encoding $E_{trajectory}$ is then repeated to match the context encoding E_{raster} dimensions. The final encoding $E_{context}$ is the result of the concatenation of both encodings E_{raster} and $E_{trajectory}$.

4.4.1.3 Increased output size for longer range

Due to high speed, some cars may go through a greater range in the time horizon T that is covered by the input range of 56m. However, simply increasing input size would greatly add to the computational burden while not necessarily bringing useful information. We therefore want to increase the output size while retaining the spatial correspondences through the layers. In order to do so, we apply Tranpose Convolutions with stride 1 and kernel size 3. Since 1 input pixel is connected to a grid of 3x3 output pixels, the edge pixels generate a new border of pixels around them, increasing the encoding size by 1 in each direction. We apply 2 of these layers, resulting in a (18, 18, 512) augmented encoding so that once upscaled the decoded image output will be of size (288, 288), corresponding to a 72m range.

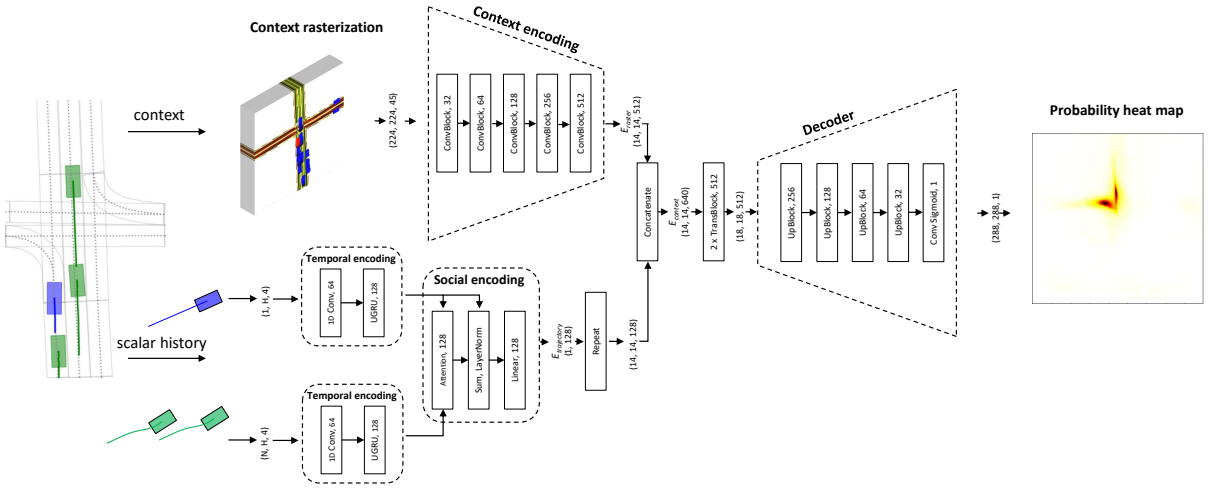


Figure 4.6: Goal conditional trajectory regression

4.4.1.4 Heatmap decoding and training

The final part of the model is a convolutional decoder alternating transpose convolutions for upscaling and classic convolutions, topped with a sigmoid activation. We output an image $\hat{Y}$ with similar resolution as the raster input (0.5 x 0.5 m² / pixel). The output target is an image Y with a Gaussian centered around the ground truth position. This image is trained with a pixel-wise focal loss inspired from [Zhou et al., 2019], averaged over the total P pixels p of the heatmap:

$$
\begin{aligned}
L &= -\frac{1}{P}\sum_{p}(Y_p - \hat{Y}_p)^2 f(Y_p, \hat{Y}_p) \\
\text{with } f(Y_p, \hat{Y}_p) &= \begin{cases} \log(\hat{Y}_p) & \text{if } Y_p{=}1 \\ (1-Y_p)^4 \log(1-\hat{Y}_p) & \text{else} \end{cases}
\end{aligned}
\tag{4.4}
$$

where the non-null pixels around the Gaussian center serve as penalty-reducing coefficients, and the square factor of error allows the gradient to focus on poorly-predicted pixels. We use a standard deviation of 0.67 pixels for the Gaussian.

4.4.1.5 Implementation details

We train all models for 16 epochs with batch size 32, using Adam optimizer initialized with a learning rate of 0.001. Each sample frame is centered on the target agent and aligned with its heading. We divide learning rate by half at epochs 3, 6, 9 and 13. We augment the training data by dropping each raster channel with a probability of 0.1 and rotating the frame by a uniform random angle in $[-\pi/4, \pi/4]$ in 50% of the samples. All convolution layers are CoordConv [Liu et al., 2018] with a kernel of size 3x3 (3 for 1D Convs) and are followed by BatchNormalization and ReLU activation.

4.4.1.6 Comparison with State-of-the-art

We show in Tab. 4.2 our results compared to other methods on the Argoverse motion forecasting test set. The benchmark is ranked by MR_6, where we rank first and significantly improve on previous results, demonstrating that having the heatmap output enables the best coverage with respect to the prior art. We also outperform other methods on p-minFDE$_6$, demonstrating superior modelling of the probability distribution between predictions. Another interesting observation is that methods performing very well on minFDE$_6$ such as LaneGCN [Liang et al., 2020] and TPCN [Ye et al., 2021] have a worse MR_6 as drawback. PRIME [Song et al., 2022] has the closest MR_6 to ours but a much higher minFDE$_6$ in comparison. We show the results of both our sampling optimized for MR and minFDE with the same trained model. Our FDE sampling with $L = 4$ sacrifices 1.1 points of MR_6 for 9 cm of minFDE$_6$, which gets us second best on minFDE$_6$ while still being good enough for 1^{st} position on the leaderboard.

Table 4.2: Results on Argoverse Motion Forecasting Leaderboard [lea,] (test set)

	K=1			K=6			
	minADE	minFDE	MR	minADE	minFDE	p-minFDE	MR
WIMP [Khandelwal et al., 2020]	1.82	4.03	62.9	0.90	1.42	3.21	16.7
LaneGCN [Liang et al., 2020]	1.71	3.78	59.1	0.87	1.36	3.16	16.3
Alibaba-ADLab	1.97	4.35	63.4	0.92	1.48	3.23	15.9
TPCN [Ye et al., 2021]	**1.64**	**3.64**	58.6	**0.85**	**1.35**	3.11	15.9
HIKVISION-ADLab-HZ	1.94	3.90	58.2	1.21	1.83	3.62	13.8
TNT [Zhao et al., 2021]	1.78	3.91	59.7	0.94	1.54	3.33	13.3
Jean [Mercat et al., 2020]	1.74	4.24	68.6	1.00	1.42	3.21	13.1
TMP [Liu et al., 2021b]	1.70	3.78	58.4	0.87	1.37	3.16	13.0
LaneRCNN [Zeng et al., 2021]	1.69	3.69	**56.9**	0.90	1.45	3.24	12.3
SenseTime_AP	1.70	3.76	58.3	0.87	1.36	3.16	12.0
poly	1.70	3.82	58.8	0.87	1.47	3.28	12.0
PRIME [Song et al., 2022]	1.91	3.82	58.7	1.22	1.56	3.04	11.5
Ours-HOME (FDE L=4)	1.72	3.73	58.4	0.92	1.36	3.08	11.3
Ours-HOME (MR)	1.73	3.73	58.4	0.94	1.45	**3.03**	**10.2**

4.4.1.7 Ablation study: Impact of heatmap output

We show the effect of output representation in Tab. 4.3 by using the same encoding backbone and replacing the image decoder with a global pooling followed by a regression head of 6 coordinate modalities. We train the regression output with a winner-takes-all l_1 regression loss similar to [Messaoud et al., 2021, Liang et al., 2020, Khandelwal et al., 2020, Ye et al., 2021, Cui et al., 2019] and a classification loss where target is obtained through a softmax on distances between predictions and ground-truth, as in [Zhao et al., 2021, Song et al., 2022]. Since the global pooling leads to loss of spatial information from the image, for fair comparison we also include a model with "scalar bottleneck" where pooling is also applied on the image encoding and is then reshaped to form an image on which the heatmap decoder is applied. For the heatmap outputs, MR sampling is used. We observe that heatmap outputs yields much better Miss Rate, and that having a scalar pooling bottleneck diminishes performance as it creates

information loss, but not significantly. Interestingly, the regression output reaches better minFDE_6 when compared to the MR-optimized sampled image output models, but is still worse than FDE-optimized model, as this scalar coordinates output doesn't leave room for any post-processing optimization.

Table 4.3: Ablation study on output representation
(Argoverse validation set)

Bottleneck	Output	K=1		K=6	
		minFDE	MR	minFDE	MR
Scalar	Regression	3.81	61.7	**1.26**	13.0
Scalar	Heatmap	3.07	51.9	1.30	8.0
Image	Heatmap	**3.02**	**50.7**	1.28	**6.8**

4.4.1.8 Qualitative results

We show supplementary qualitative results in Fig. 4.7. We highlight examples of straight line, overtaking, curve road, going outside the map and intersections. Our model heatmap output makes use and usually follows the prior from the context map, but it is also able to divert from it based on interactions, realistic observations and hints of divergence from history.

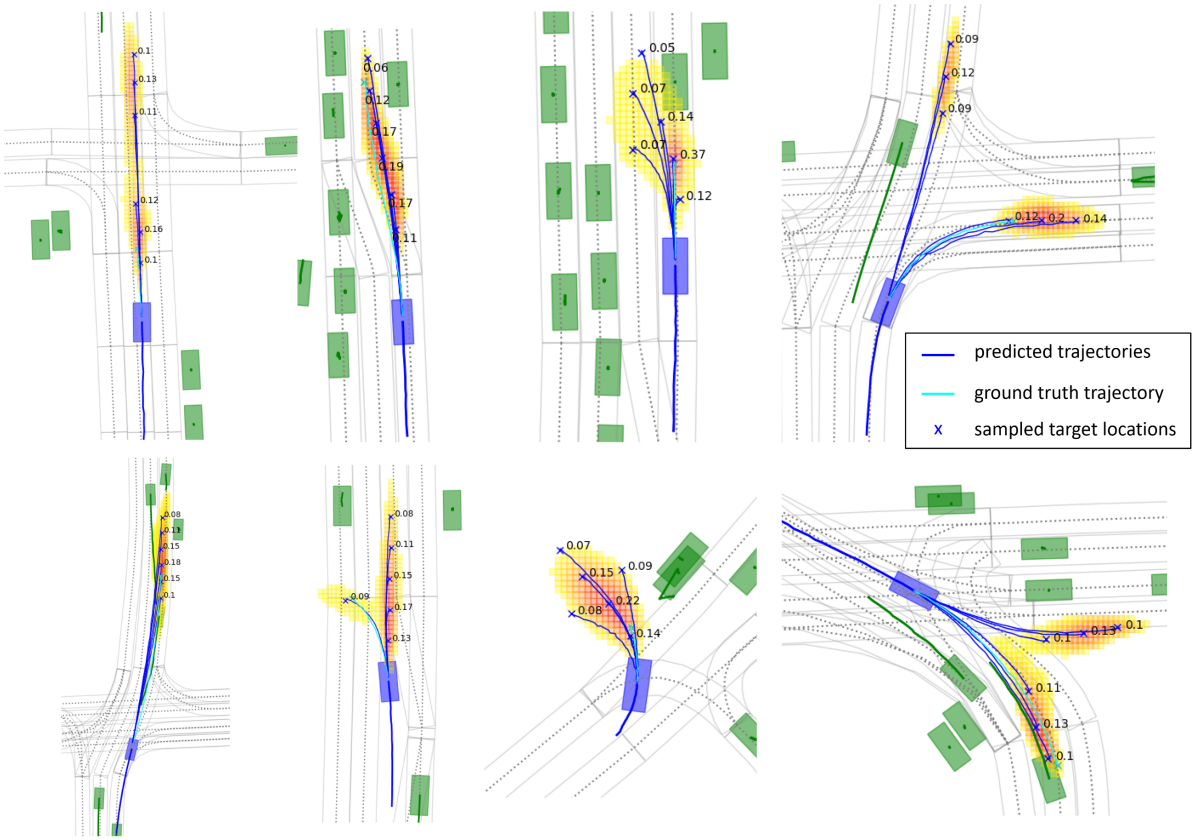


Figure 4.7: Qualitative examples. The yellow/red heatmap is our predicted probability distribution and the blue points are the sampled final point predictions. The ground truth trajectory is shown in green.

4.4.1.9 Limitations

Despite their widespread use and optimization on most deep learning frameworks, convolutional layers remain quite slow compared to simple scalar fully-connected layers, especially considering the respective sizes of processed inputs (a full image versus a few coordinates). As a consequence, we observe our training times to be a bit slower than what would be optimal for fast and efficient research (a few days per training), and ill-suited for real-time on-board inference ($\sim$ 200 ms per inference, while <100 ms would be desirable for a full real-time self-driving pipeline).

Moreover, while the original maps provided in most datasets provide connectivity information (which lane are connected together), which is especially convenient when it comes to complex intersections with multiple possible lane changes and turns as in the last bottom-right example of Fig. 4.7, our convolutional encoder doesn't consider this information, and represents the whole road map in one single layer where overlapping lanes might occlude each other.

4.4.2 Lane-based heatmaps (GOHOME)

HOME uses a fully-convolutional model and is limited to a restricted image size. GOHOME iterates on this work and presents an optimized motion forecasting framework solely based on graph operations to provide efficiently a heatmap with uncertainty measure exploiting the vectorized form of HD map.

GOHOME system focuses on lane-level operations as illustrated in Fig. 4.8. The local HD-Map is provided in the dataset as a graph of L lanelets. A lanelet represents a macro section of the road (10 to 20 meters on average), as our goal is to encode connectivity at a macro level (lane segments), and not micro level (every meter). Each lanelet is defined as a sequence of centerline points, and is connected to its predecessor, successor, left and right neighbors if they exist. We encode each lanelet into a road graph, where geometric and connectivity information are represented. Our model yields a score for each of these lanelets, that is used to identify most probable lanes. A partial heatmap is then generated for the top ranking lanelets, and projected onto a global heatmap. Afterwards, we sample a set of endpoints from the heatmap and recreate a trajectory for each.

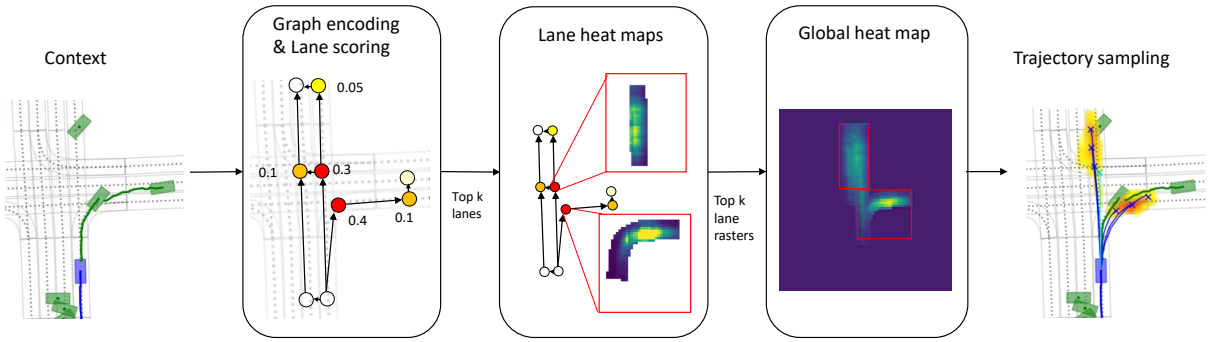


Figure 4.8: GOHOME pipeline. The lane graph extracted from the HD-Map is processed through a graph encoder. Each lane then generates a local curvilinear raster that is combined into a predicted probability distribution heatmap.

4.4.2.1 Graph neural network for HD-Map input

The model architecture is illustrated in Fig. 4.9. We encode each lanelet through a shared 1D convolution and UGRU [Rozenberg et al., 2021] recurrent $LaneEncoder$ into features F of C channels. The lanelet features F are then updated through a $GraphEncoder1$ made of a sequence of four graph convolution operations similar to [Liang et al., 2020] in order to spread connectivity information:

$$F \leftarrow FW + \sum_r A_r F W_r \tag{4.5}$$

where F is the (L,C) lane feature matrix , W is the learned (C,C) weight for ego features encoding. A_r and W_r are the respective adjacency matrix (L,L) and learned weight (C,C) for the relation $r \in \{predecessor, successor, left, right\}$ derived from the lane graph. A_r is fixed as it comes from the HD Map, while W_r enables to learn different operations for each possible relation.

Parallelly, each agent trajectory, defined as a sequence of position, speed and yaw, is encoded with a shared $TrajEncoder$ also made of a shared 1D convolution and a UGRU layer. Each agent feature is then updated with map information through a cross-attention $Lanes2Agents$ layer on the lanelet features.

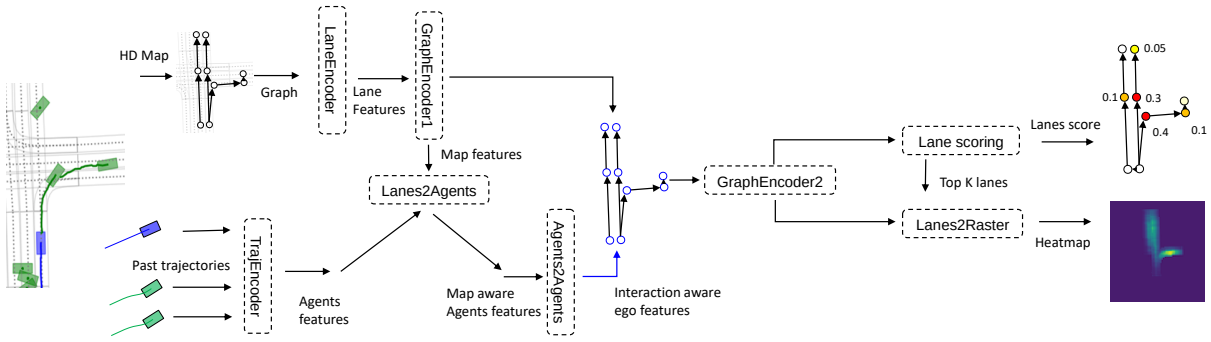


Figure 4.9: GOHOME model architecture

Interactions between agents are then taken into account through a self-attention $Agents2Agents$ layer between agents. Finally the target agent feature is concatenated to all the lanelet features by $Ego2Lanes$ and then treated through a final $GraphEncoder2$ layer, also made of 4 graph convolutions to obtain the final graph encoding that will be used to generate the different predictions.

Compared to other methods using graph neural networks, our method uses graph convolutions like LaneGCN [Liang et al., 2020] and LaneRCNN [Zeng et al., 2021], but applies them to lanelets instead of lane nodes (a lane node is a single point in the sequence of a lanelet). VectorNet [Gao et al., 2020] and TNT [Zhao et al., 2021] also use lanelets, called polylines, but connect them through global attention instead of using graph connectivity. We chose to use a GNN on the lanelets since we wanted an efficient and high-level representation allowing to spread information easily through the graph, while still leveraging connectivity. A more detailed description of the architecture is given in Appendix A.

4.4.2.2 Heatmap generation through Lane-level rasters

For the heatmap output, we wish to have a dense image in cartesian coordinates of dimensions (H, W). To do so without using any convolution on the full image, we create a raster for every lanelet in curvilinear coordinates. We use lane ranking to generate these lane rasters only for the top k lanelets and not all of them.

Lane raster generation Each of the small lane rasters of size (h, w) has a longitudinal lenght of 20m and a transversal width of 4m. These lane rasters are created as a discretization of the Frenet-Serret referential along the lane, as illustrated in Fig. 4.10. We decompose the probability distribution along a lanelet in a longitudinal component $(h, 1, 8)$ and a lateral component $(1, w, 8)$ predicted from the lanelet encoding. These components are summed together with broadcast to create a $(h, w, 8)$ $R_{features}$ volume. This way the complexity to create the volume is $(h+w)\times 8$ instead of $h\times w\times 8$. The obtained volume is then concatenated with pixelwise cartesian coordinates, heading, lane occupancy and curvature informations before a final linear layer on which is applied a sigmoid to get the R_{proba} output.

The resulting lane-level R_{proba} heatmaps are then projected onto the full cartesian heatmap $\hat{Y}$ using each pixel cartesian coordinates as illustrated in Figure 4.10. If multiple lane-level pixels fall into the same cartesian pixel, their values are averaged. The lane raster widths are set such that adjacent lane rasters overlap and can cover lane change behaviors. The target Y for this final prediction $\hat{Y}$ is a Gaussian centered around the target agent ground truth position. We use the same pixel-wise focal loss as HOME defined in Eq. 4.4.

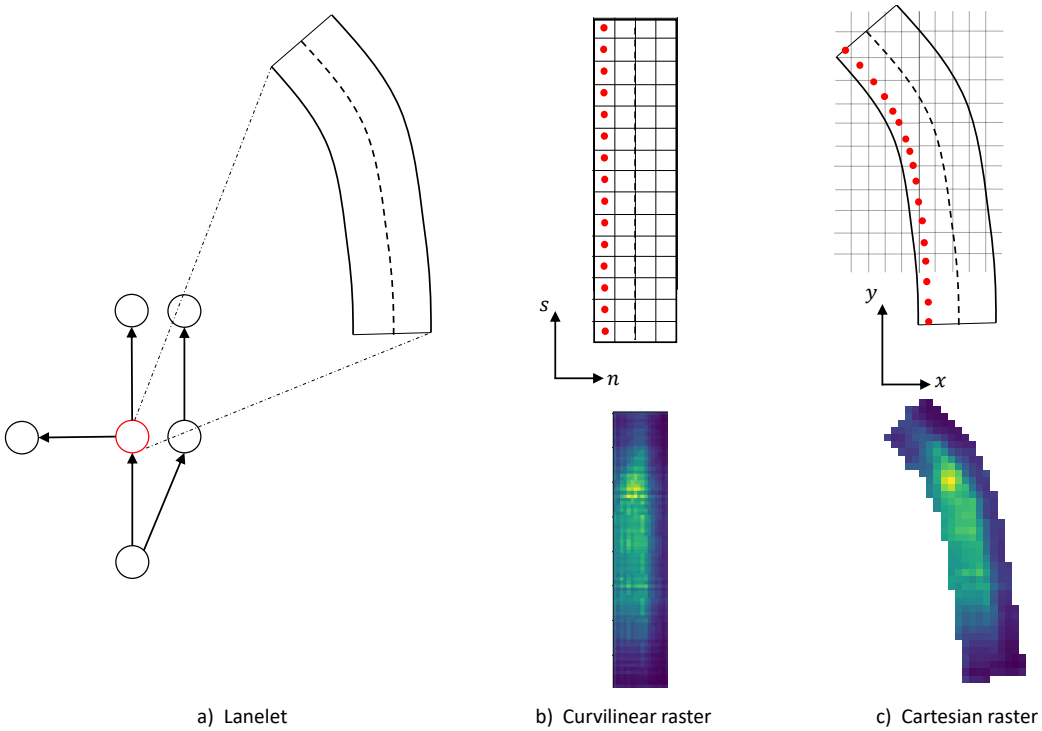


Figure 4.10: Lane raster grid projection onto cartesian coordinates. a) A single node of the graph is a lanelet and describes a road segment. b) A rectangular raster is generated along the curvilinear coordinates of the lanelet. c) The lanelet coordinates are then used to project the predicted raster back into cartesian coordinates to complete the final heatmap output.

Lane ranking The lanelet classification is obtained with a linear layer on the graph encoding, followed by a sigmoid activation. The ground-truth is defined by a 1 for all lanelets where the future car position is inside the lanelet polygon and 0 otherwise. The loss is a binary cross-entropy added to the pixel-wise loss of Eq. 4.4 with a $1e^{-2}$ coefficient. Since only a fraction of the lanelets will actually be useful to represent the future car location, we can compute the lane-level rasters only for a subselection of lanelets, saving more computation. We use the classification score c_{lane} predicted by the network to select only the top k ranking lanelets, and only compute and project the lane raster for these. Since the raster predictions for the other unselected lanelets would have been very close to zero anyway, this does not decrease performance at all, as demonstrated in Sec. 4.4.2.4.

Cartesian image connection Some lane rasters may be projected onto the same pixels and overlap, which is very difficult for the model to know of beforehand. To help the model know of overlaps and propagate location information through the lane rasters, we do a first projection of the lane rasters onto the cartesian coordinates before the final probability estimation. The $(h, w, 8)$ lane raster features $R_{features}$ are projected onto a $(H, W, 8)$ cartesian image $I_{features}$ through the same operation previously described for the probability heatmaps, with the overlaps averaged. The occupancy (number of raster pixels aggregated in each cartesian pixel) information is also concatenated to the volume $I_{features}$. A linear layer is then applied on the last dimension of $I_{features}$, which is then reprojected onto curvilinear coordinates and concatenated to the initial lane rasters $R_{features}$ before final probability estimation and projection. This way, the features of overlapping lanes are shared between them so that they can propagate information and homogenise probability.

4.4.2.3 Comparison with State-of-the-art

We report our results on the online Argoverse test set in Tab. 4.4, on the online NuScenes leaderboard in Tab. 4.5, and on the Interaction validation set in Tab. 4.6. We compare it to the published methods on each of these benchmarks. On Argoverse, our GOHOME method reaches 2^{nd} place in MR_6, with the use of a lighter and faster model than 1^{st} HOME as will be showed in Sec. 5.4.3. On NuScenes and Interaction, GOHOME ranks first in multiple metrics as well.

Table 4.4: Argoverse Leaderboard [Chang et al., 2019] [1]

	K=1		K=6		
	minFDE	MR	minADE	minFDE	MR
LaneGCN [Liang et al., 2020]	3.78	59.1	0.87	1.36	16.3
TPCN [Ye et al., 2021]	3.64	58.6	**0.85**	1.35	15.9
Jean [Mercat et al., 2020]	4.24	68.6	1.00	1.42	13.1
SceneTrans [Ngiam et al., 2021]	4.06	59.2	**0.80**	**1.23**	12.6
LaneRCNN [Zeng et al., 2021]	3.69	56.9	0.90	1.45	12.3
PRIME [Song et al., 2022]	3.82	58.7	1.22	1.56	11.5
DenseTNT [Gu et al., 2021]	3.70	59.9	0.94	1.49	10.5
HOME	3.65	57.1	0.93	1.44	9.8
GOHOME	3.65	57.2	0.94	1.45	10.5

Table 4.5: NuScenes Leaderboard [Caesar et al., 2020] [2]

	K=5		K=10		k=1
	minADE	MR	minADE	MR	minFDE
CoverNet [Phan-Minh et al., 2020]	1.96	67	1.48	_	_
Trajectron++ [Salzmann et al., 2020]	1.88	70	1.51	57	9.52
ALAN [Narayanan et al., 2021]	1.87	60	1.22	49	9.98
SG-Net [Wang et al., 2021a]	1.86	67	1.40	52	9.25
WIMP [Khandelwal et al., 2020]	1.84	55	**1.11**	43	8.49
MHA-JAM [Messaoud et al., 2021]	1.81	59	1.24	46	8.57
CXX [Luo et al., 2020]	1.63	69	1.29	60	8.86
LaPred [Kim et al., 2021]	1.53	_	1.12	_	8.12
P2T [Deo and Trivedi, 2020]	1.45	64	1.16	46	10.50
GOHOME (r=2.6m)	**1.42**	57	1.15	47	**6.99**
GOHOME (r=1.8m)	1.59	**46**	1.15	**34**	7.01

4.4.2.4 Ablation studies

We highlight the gains made by replacing convolution operations with graph operations. To measure inference time, we use a batchsize of 16, which can be considered as an average number of agents to be

[1] `https://eval.ai/web/challenges/challenge-page/454/leaderboard/1279`
[2] `https://eval.ai/web/challenges/challenge-page/591/leaderboard/1659`

Table 4.6: Interaction Validation [Zhan et al., 2019]

	minFDE_1	minFDE_6
TNT [Zhao et al., 2021]	_	0.67
HEAT-I-R [Mo et al., 2021]	0.66	_
ReCoG [Mo et al., 2020]	0.65	_
ITRA [Scibior et al., 2021]	_	0.49
GOHOME	**0.61**	**0.45**

predicted at a given time. We report only the model forward pass, omitting preprocessing and postprocessing, but notice that image preprocessing is sensibly slower, particularly because of the rasterization of the different semantic layers. All times are measured with a Nvidia 2080 TI. While we report inference time, we also notice that training times record an even greater difference. We mostly consider three different architectures: HOME, GNN-HOME which is a modified HOME model with a GNN encoder but the usual CNN decoder, and our new method GOHOME. All numbers are reported on the Argoverse validation set.

Graph operation speed-up

We evaluate the speed-up gained by using graph encoding and lane rasters instead of full convolutions in Tab. 4.7. The CNN encoding and decoding corresponds to the full HOME model, and the GNN with Lane Rasters (LR) to the GOHOME model. We also estimate the impact from the encoding separately by testing a model with GNN encoding and CNN decoding. We report FLOPs, number of parameters and Frame per Seconds. We measure an average number of 140 lanelets and 10 agents per sample to compute FLOPs.

Table 4.7: Performance/Complexity comparison

Model	K=6		#Param	FLOPs
	minFDE	MR		
HOME	1.28	6.8	5.1M	4.8G
GNN-HOME	1.28	7.2	0.43M	0.81G
GOHOME	1.26	7.1	0.40M	0.09G

Table 4.8: Lane ranking speed-up

# lanes	K=6		FPS
	minFDE	MR	
All	1.28	7.5	17
20	1.26	7.1	34
10	1.26	7.3	45

Trade-off from lane ranking

We show in Tab. 4.8 the effects of only selecting the top k lanes to extract rasters. We fix an input range of 128 and output range of 192 with 0.5m x 0.5m pixel resolution. We observe that this ranked selection doesn't decrease performance, as limiting the number of projected lanes seems to actually improve the metrics, and brings effective speed-up.

Image size and resolution scaling

While a 192m image range, which amount to a 88m reach in each direction, may be sufficient in most urban driving predictions with a time horizon of 3s, other datasets can require predictions up to 6 or 8 seconds [Caesar et al., 2020, Ettinger et al., 2021]. There is therefore a need to increase this output range, which can be done without necessarily increasing the input size, as far distances are reached with

long straight trajectories that can be easily extrapolated on highways.

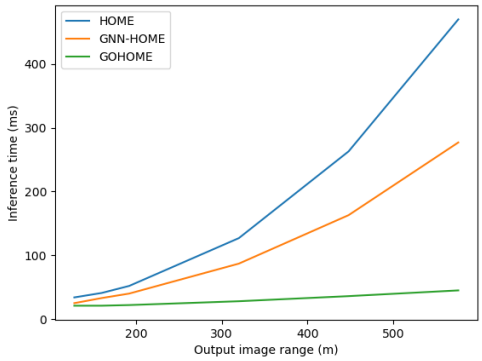


Figure 4.11: Inference time as a function of output range

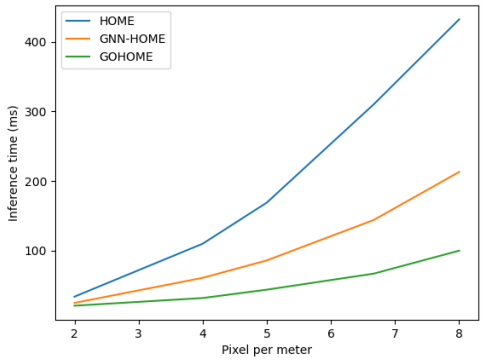


Figure 4.12: Inference time as a function of pixels per meters

We compare the scaling of our graph-based GOHOME model to the one of an image-based HOME model with regard to output range and resolution. Fig. 4.11 highlights the output range scaling, where we use a fixed input range of 128 meters, and a resolution of 0.5m per pixel. Whereas the CNN decoders of HOME and GNN-HOME lead to a quadratic scaling, the lane rasters combined with the top 20 lane ranking enable a scaling that is even less than linear.

We show in Fig. 4.12 the inference time with regard to the number of pixels per meter, which is the inverse of the resolution. While efficient optimization of convolution and constant costs lead to close inference times for the initial 2 pixels per meter, the more efficient scaling shows clearly for GNN inputs and especially Lane Rasters outputs. As the resolution of the lane rasters is also scaled with the total resolution, the quadratic complexity is still applied, but with a much lesser coefficient that allows for realistic training and inference times for finer resolutions.

4.4.2.5 Qualitative analysis

We show in Fig. 4.13 some qualitative results of our GOHOME model. The lane prediction displayed on top can be assimilated to the representation of epistemic uncertainty, as the choice of where the driver will decide to go, whereas the spread of the final heatmap modes models aleatoric uncertainty in the trajectory controls. We observe that the model assigns different modes for each lane possibility, and that each of these modes is well aligned with the corresponding lane with a spread along the curvilinear direction. More qualitative examples are given in Appendix B.

We also display in Fig. 4.14 the comparison of outputs between HOME and GOHOME on the same sample. GOHOME is more tightly correlated to centerlanes, which can help with the accuracy of the final point, but has a less exhaustive coverage of possible lane change maneuvers and going outside of the drivable area. HOME on the other hand is more diffuse, and can cover cars going out the road more easily. Their combination allows them to combine their advantages and fill their drawbacks, leading to improved performance.

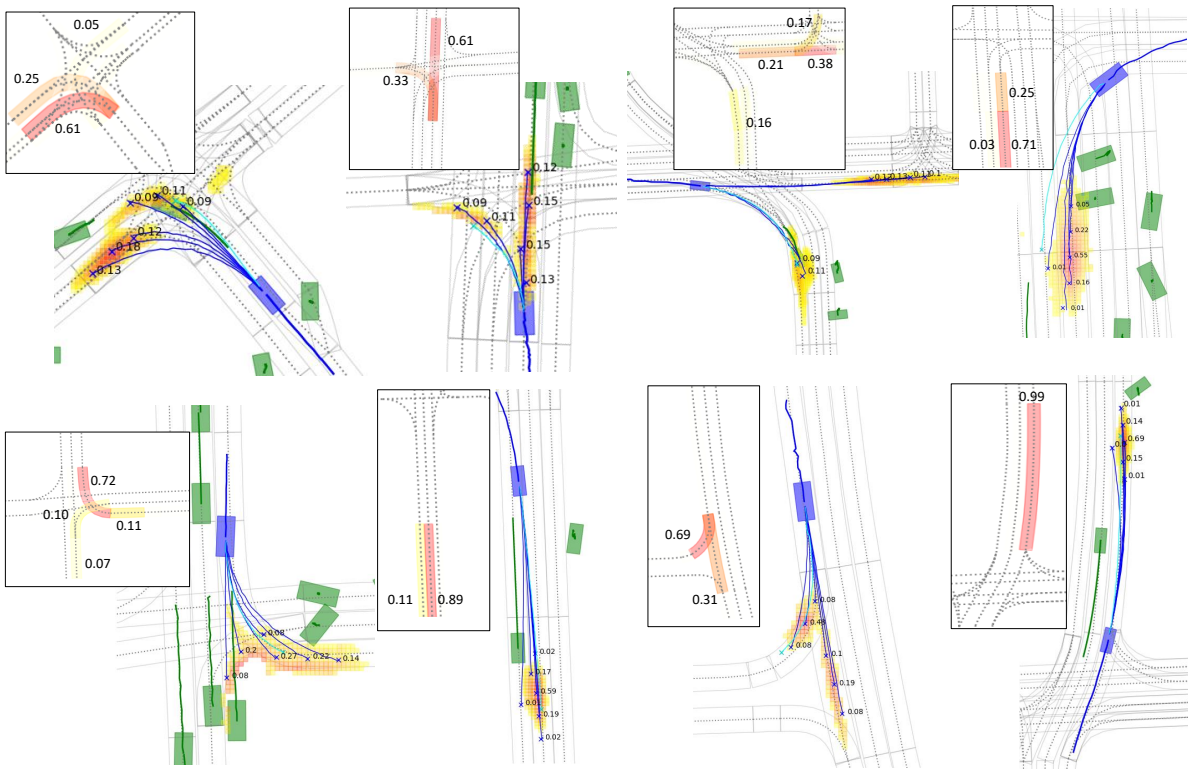


Figure 4.13: Qualitative examples of GOHOME output. Graph lane classification is shown in framed inserts

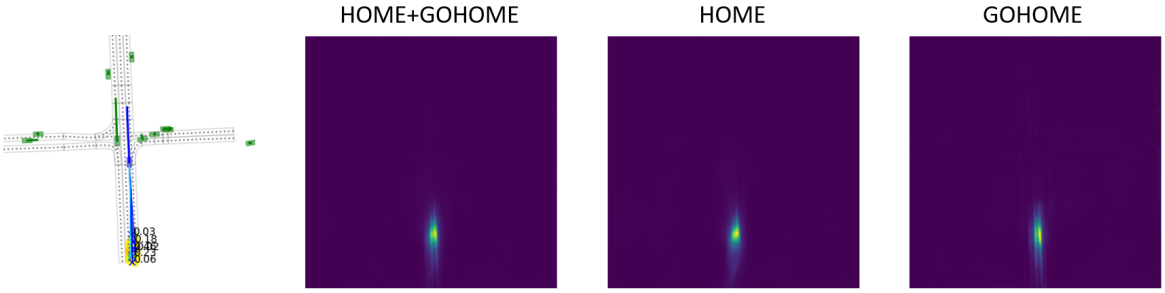


Figure 4.14: Comparison of HOME and GOHOME predictions on the same sample. The prediction of GOHOME follows the centerlanes more closely, while HOME is more spread between the lanes.

4.4.3 Cross-attention for heatmap generation (THOMAS)

The approach of GOHOME remains constrained to the drivable area and to fixed lane-widths. Moreover, a significant amount of preprocessing has to be done to prepare the lane raster coordinates. We inspire ourselves from DenseTNT [Gu et al., 2021] to simplify the sparse heatmap generation process in THOMAS, by using attention, but extend their work by not being constrained to a neighborhood of the drivable area, and implementing a hierarchical process to only decode the grid areas with high prior probability. THOMAS uses the same architecture as GOHOME for context encoding, and only differs in the heatmap decoding part. We give a precise illustration of our model architecture with each layer size in Appendix C.

4.4.3.1 Attention to grid points

We use sparse points to generate probability heatmaps as in DenseTNT [Gu et al., 2021]. Each pixel on the grid is defined by its 2D coordinates, from which features are computed by a 2-layer MLP applied on the point coordinates. These features are then concatenated to the agent encoding obtained through the GOHOME encoder, passed through a linear layer, and finally enriched by a 2-layer cross-attention on the graph lane features, before applying a linear layer with sigmoid to get the probability.

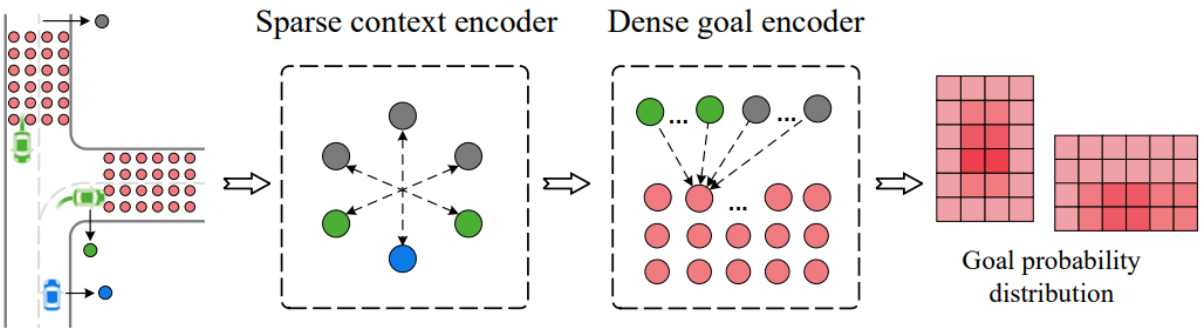


Figure 4.15: Use of attention to generate heatmap probabilities [Gu et al., 2021]. Grid points are initialized by their coordinates, and then cross-attention is processed between these grid-points as queries and the context as key/values. Once context information has been included, linear layers are applied to obtain a final probability value.

4.4.3.2 Hierarchical grid decoder

We use hierarchical predictions at various levels of resolutions so that the decoder has the possibility of predicting over the full surroundings of the agent but learns to refine with more precision only in places where the agent will end up with high probability. This hierarchical process is illustrated in Fig. 4.16.

Starting from an initial full dense grid probability at low resolution $R_0 \times R_0$ by pixels, we iteratively refine the resolution by a fixed factor f until we reach the desired final resolution $R_{final} \times R_{final}$. At each iteration i, we select only the N_i highest ranking grid points from the previous iteration, and upsample only these points to the $R_i \times R_i = {}^{R_{i-1}}/_{f} \times {}^{R_{i-1}}/_{f}$.

For a given W output range, this hierarchical process allows the model to only operate on ${}^{W}/_{R_0} \times {}^{W}/_{R_0} + \sum_i N_i \times f^2$ grid points instead of the ${}^{W}/_{R_{final}} \times {}^{W}/_{R_{final}}$ available. In practice, for a final output range

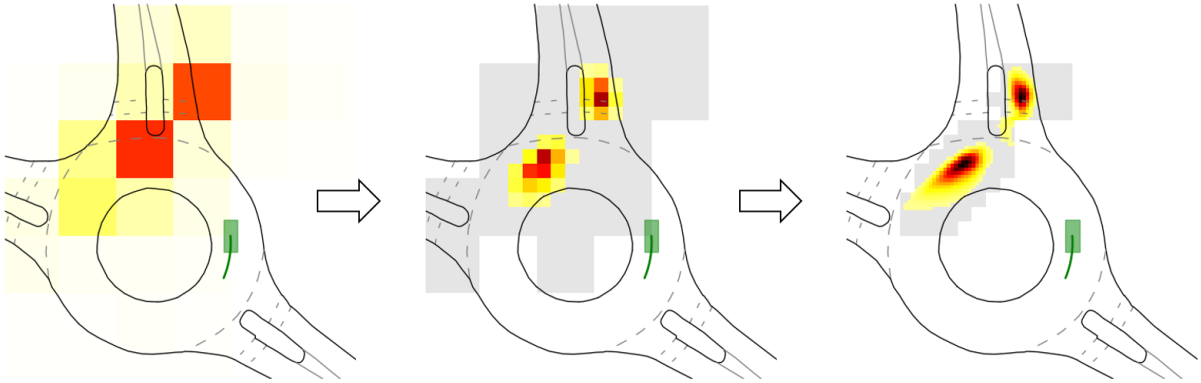


Figure 4.16: Hierarchical iterative refinement of the grid probabilities. First, the full grid is evaluated at a very low resolution, then the highest cells are up-sampled and evaluated at a higher resolution, until final resolution is reached. We highlight in grey the restricted area considered for refinement at each step.

of 192 meters with desired $R_{final} = 0.5m$ resolution, we start with an initial resolution of $R_0 = 8m$ and use two iterations of $(N_1, N_2) = (16, 64)$ points each and an upscaling factor $f = 4$. This way, we compute only 1856 grid points from the 147 456 available, with no performance loss.

The heatmap is trained on each resolution level using as pixel-wise focal loss as in Eq. 4.4, with the ground truth being a Gaussian centered at the target agent future position.

4.4.3.3 Comparison with State-of-the-art

We report the results of our THOMAS prediction model on the datasets Argoverse and nuScenes in Tab. 4.9 and 4.10 respectively.

Table 4.9: Argoverse Leaderboard [Chang et al., 2019] [1]

	K=1		K=6		
	minFDE	MR	minADE	minFDE	MR
Autobot [Girgis et al., 2021]	_	_	0.89	1.41	16
TPCN [Ye et al., 2021]	3.64	58.6	0.85	1.35	15.9
SceneTrans [Ngiam et al., 2021]	4.06	59.2	**0.80**	**1.23**	12.6
DenseTNT [Gu et al., 2021]	3.70	59.9	0.94	1.49	10.5
GOHOME	3.65	57.2	0.94	1.45	10.5
HOME	3.65	57.1	0.93	1.44	**9.8**
THOMAS	**3.59**	**56.1**	0.94	1.44	10.4

THOMAS reaches similar or better performance than both HOME and GOHOME, and also fares very well with other state-of-the-art methods, notable on the minFDE_1 metric, while reducing considerably its memory and computing footprint.

[1] https://eval.ai/web/challenges/challenge-page/454/leaderboard/1279
[2] https://eval.ai/web/challenges/challenge-page/591/leaderboard/1659

Table 4.10: NuScenes Leaderboard [Caesar et al., 2020] [2]

	K=5		K=10		k=1
	minADE	MR	minADE	MR	minFDE
P2T [Deo and Trivedi, 2020]	1.45	64	1.16	46	10.50
GOHOME	1.42	57	1.15	47	6.99
Autobot [Girgis et al., 2021]	1.37	62	1.03	44	8.19
PGP [Deo and Trivedi, 2020]	**1.30**	57	**0.98**	**37**	7.72
THOMAS	1.33	55	1.04	42	**6.71**

4.4.3.4 Ablation study

We use the Interaction v1.2 dataset that has recently opened a new multi-agent track in the context of its Interpret challenge. It contains 47 584 training cases, 11 794 validation cases and 2 644 testing cases, with each case containing between 1 and 40 agents to predict simultaneously.

We assess the speed gain of our proposed hierarchical decoder compared to the lane rasters of GOHOME. In Tab. 4.11. We report training time for 16 epochs with a batchsize of 16, and inference time for 32 and 128 simultaneous agents (if one scene contrains less than the infered number of agents, the lacking agents are padded with zeros and therefore still processed by the model in a similar time budget).

Table 4.11: Comparison of consistent solutions on Interpret multi-agent validation track

	Training	Inference 32 agents	Inference 128 agents
GOHOME	12.8 hours	36 ms	90 ms
THOMAS	7.5 hours	20 ms	31 ms

For additional comparison, the other existing dense prediction method DenseTNT [Gu et al., 2021] reports an inference speed of 100ms per sample for their model.

Speed / Performance trade-off with hierarchical refinement We also display the trade-off between inference speed and coverage from hierarchical refinement in Fig. 4.17, with marginal MissRate_6 as the coverage metric.

The curve is obtained setting the number N of upsampled points at the last refinement iteration from 2 to 128. From $N = 16$ and lower, coverage performance starts to diminish while little speed gains are made. We still kept a relatively high N=64 in our model as we wanted to insure a wide coverage, and the time loss between 41 ms and 46 ms remains acceptable.

4.4.3.5 Qualitative examples

We display examples of multi-agent heatmap prediction from out THOMAS model in Fig. A.3.

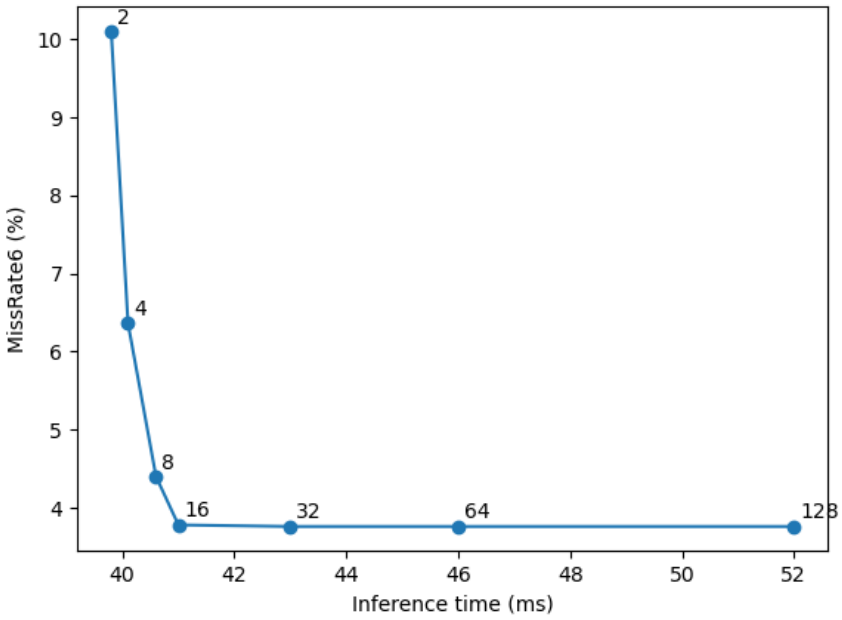


Figure 4.17: Curve of MissRate_6 with regard to inference time with varying number of points upsampled at the last hierarchical refinement iteration.

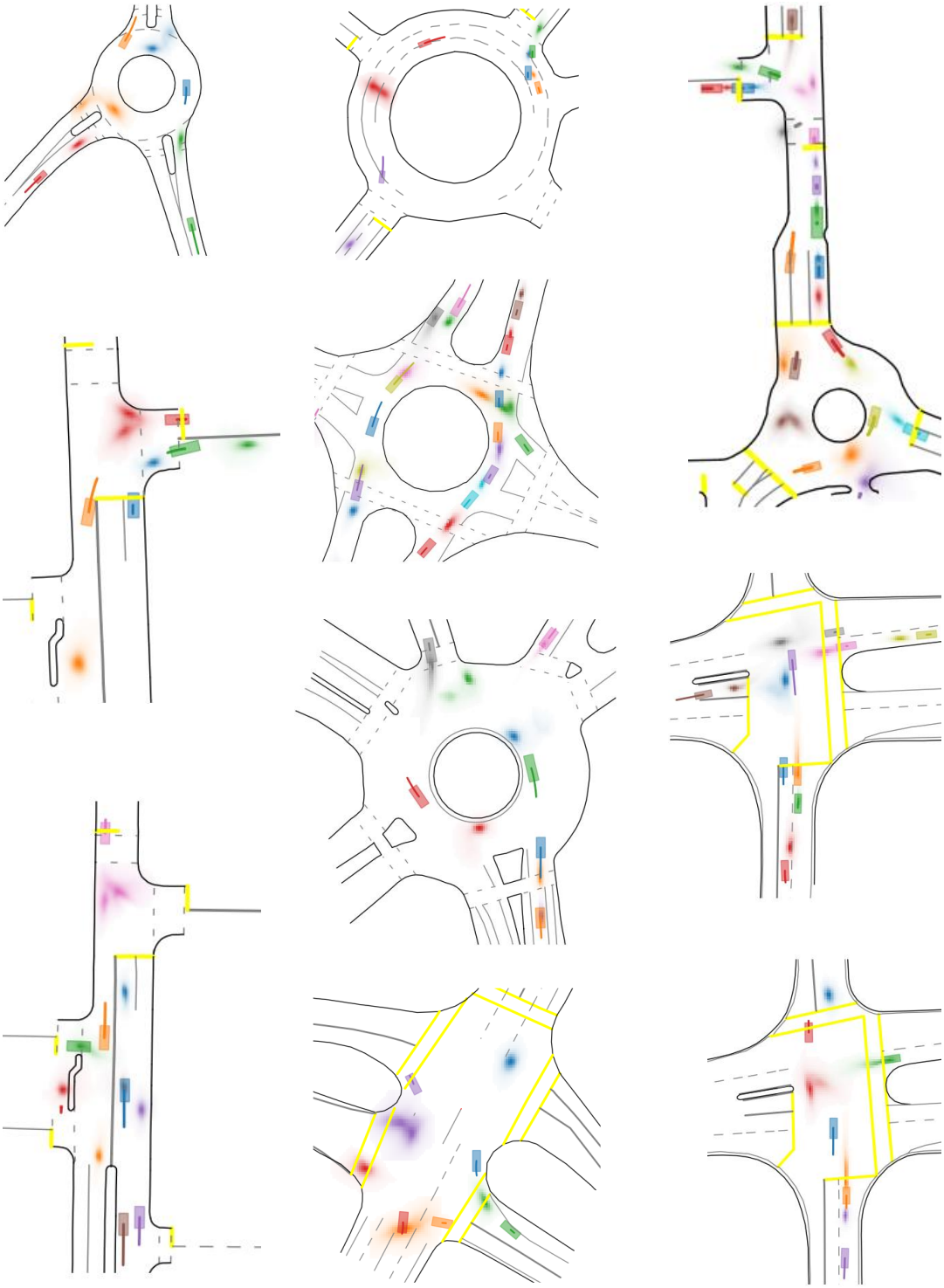


Figure 4.18: Qualitative examples of heatmap output from our multi-agent model. All the heatmaps from one scene are resulting from one single forward pass in our model predicting all agents at once. We use matching colors for the agent history, current location and their future predicted heatmap (best viewed in color).

4.5 Advantages of heatmaps

After presenting different ways of generating a heatmap, we now interest ourselves to the general concept of heatmap-based trajectory prediction and investigate their pros compared to more common motion forecasting methods.

4.5.1 Performance

As we have reported in multiple benchmarks [Chang et al., 2019, Zhan et al., 2019, Caesar et al., 2020] on Tables 4.2, 4.4, 4.5, 4.6 and 4.9, heatmap models obtain state-of-the-art performance, notably in the minFDE_1 and MR_6, which we can translate in the following ways:

The resulting per-pixel multimodality provides a very good ranking of the most likely places where the agents will go, resulting in an accurate ordering of the predicted modes (low minFDE_1. Compared to most Winner-Takes-All [Liang et al., 2020, Ngiam et al., 2021] models which have troubles estimating which of their multiple modes is the most probable, the probability values our model attributes to each location translate to a globally precise and comparable estimate, showcasing a good predicted probabilistic distribution.

The final sampled modes give a very good coverage of the possible reachable positions, reaching MR_6 values unequalled in any other method. Since our model prediction is unconstrained and trained in a framework that strongly encourages to not disregard any probable pixel, and through the use of a sampling algorithm designed with coverage performance in mind, we are able to provide a prediction that, while not necessarily the most accurate to the nearest centimeter, will almost always have at least one predicted mode in the local vicinity of the ground truth.

4.5.2 Multi-modality

We compare the effect of adding more modalities obtained by HOME to the one derived from a regression output in Fig 4.19. Even if the MR_k improves for the total number of modalities as k increases, the performance for a fixed k such as 1 or 6 worsens. [Khandelwal et al., 2020] and [Zhang et al., 2020] notice a similar trend, obtaining much better results for $k = 1$ metrics when training less total modalities. Furthermore, for a regression output model a new training is required each time to accommodate the maximum number of modalities, whereas with heatmap output any number of modalities can be obtained at will with the same training, and the lower k numbers are not impacted by the total number of modalities extracted, as showed by the dashed horizontal lines displayed for MR_1, MR_3 and MR_6. Finally, our model heatmap output scales better with the number of k modalities, converging to a 0% MR faster that the regression output model.

4.5.3 Ensembling for increased performance and highlight of model differences

Because of the multimodal nature of the predictions, model ensembling is usually tedious in trajectory prediction, as it is not possible to determine which modality should be averaged with which, and even shortest distance matching doesn't guarantee that two predictions highlight the same decision and would make sense averaged together. On the other hand, probability heatmap are a great way of representing

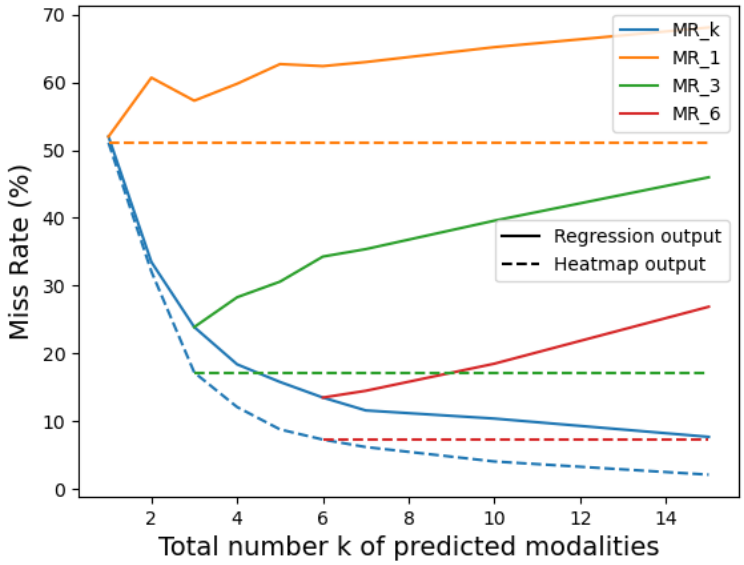


Figure 4.19: Effect of maximum number k of predicted modalities trained on metrics of lower fixed modality numbers. Full lines are results of regression output model. Dashed lines are result of our heatmap output model. We show the Miss Rate for total number of predicted modalities k (blue) and fixed number of modalities 1 (orange), 3 (green) and 6 (red). In the regression output case, since the training of each individual modality is dependent on the total number k of modality trained, the metrics $MR_{1,3,6}$ are not fixed and worsen when k increases.

information coming from different sources or models in a common system of reference and can be averaged together without any assumption nor risk of mode collapsing.

We report the results of our ensembled models on the test set in Tab. 4.12. We first highlight that the ensemble of two similar HOME models (HO+HO) brings significant improvement compared to HOME alone. As a general rule, the more different and complementary two models are, the greater the performance increase will be. We notice that the combination of HOME and GOHOME models (HO+GO) brings a greater improvement than HO+HO, despite each HOME model being better in single performance than the GOHOME model. Our best ensembling, a weighted combination of 9 HOME and GOHOME models, allows us to improve on the existing state-of-the-art by a significant margin, with a more than 15% MR_6 decrease.

Table 4.12: Argoverse Leaderboard [Chang et al., 2019] [1]

	K=1		K=6		
	minFDE	MR	minADE	minFDE	MR
HOME	3.65	57.1	0.93	1.44	9.8
GOHOME	3.65	57.2	0.94	1.45	10.5
HO+HO	3.57	56	0.92	1.41	9.4
HO+GO	**3.53**	**55.8**	0.92	1.40	9.1
Best ensemble	3.68	57.2	0.89	1.29	**8.5**

We also show in Fig. 4.20 the heatmap output of two model sharing the same architecture but

[1] https://eval.ai/web/challenges/challenge-page/454/leaderboard/1279

different training seeds. Each model chooses a separate lane, highlighting the benefits of the ensemble that covers all possible lane changes.

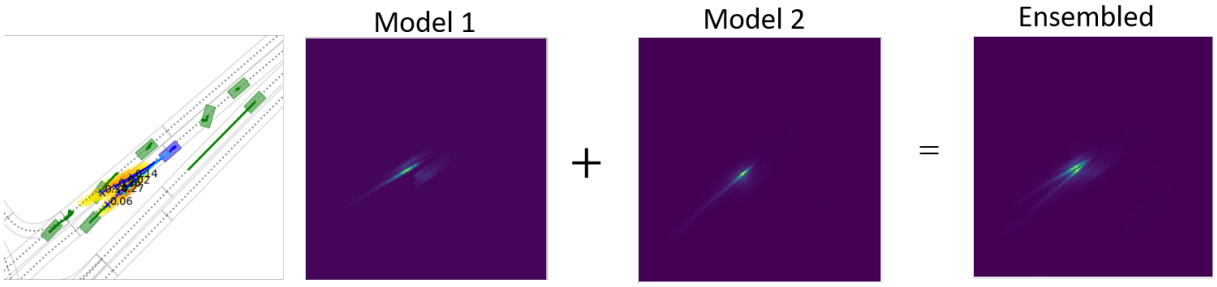


Figure 4.20: Complementarity of models in an ensemble. Model 1 predicts a lane-change to the right while Model 2 predicts keeping the same lane, and the Ensemble covers both case.

4.6 Conclusion

We have demonstrated in this chapter the advantages of heatmap-based trajectory prediction. We have also identified their weaknesses, notably their slower compute time, and tried to remediate to this problem by designing new model architectures enabling a more efficient heatmap generation, with inference times close to the non-heatmap based ones. We have highlighted the very significant gains in metrics such as MR_6, but also note that these gains also come at a cost on other metrics like $minFDE_6$. We notice that a trade-off usually exists, not only with our methods but in almost all state-of-the-art approaches, between these two metrics. While this trade-off is not absolute, as a more powerful model can gain performance on both fronts, for a fixed model design choices will usually lead to preferences in one quality (coverage) or another (accuracy). We note that an advantage of our heatmap approaches is that we can explicitly tune this trade-off after training by setting a single parameter in the sampling step during inference time.

Chapter 5

Multi-agent consistent prediction

Contents

The methods presented in the previous chapters focus on individual agent forecasts, however this prediction always takes place in a multi-agent interactive setting. In this chapter we now focus on how to obtain scene-level multi-agent consistent predictions.

5.1 Scene-level consistent prediction

In the context of autonomous driving, the self-driving stack must detect, track and predict not one, but A multiple surrounding agents at any given time. Following what we have said in earlier chapters, this prediction must be K multimodal for every agent. However, if no additional constraint is applied, the k^{th} modality of the first agent will not be consistent with the k^{th} modality of the second agent, and may even collide with it. This means that the planning step cannot use the agent modalities in unison with each other, and therefore has to explore all K^A possible combinations, growing exponentially with the number of agents. Such a strategy is not viable, hence a need for consistency constraints on the multi-agent multimodal prediction.

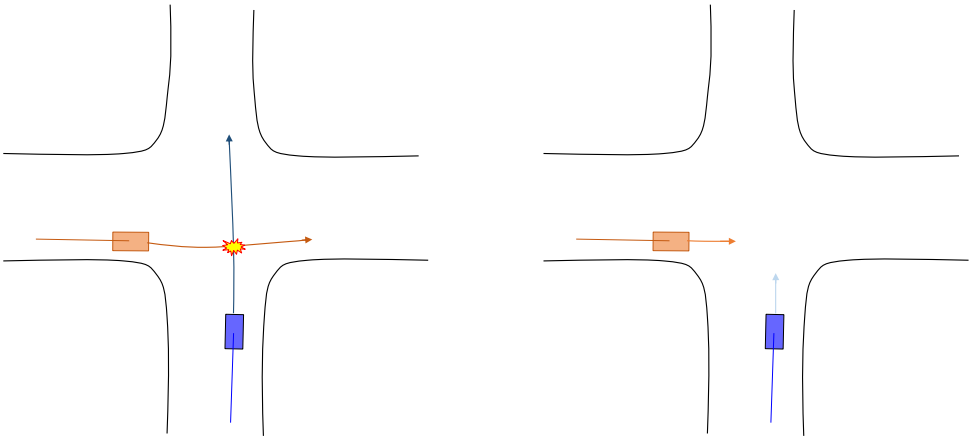


Figure 5.1: Two failure cases of multi-agent consistency. **Left**: Both agents are predicted to cross the intersection at the same time. **Right**: Both agent yield and nobody crosses.

Fig. 5.1 illustrates two possible errors in consistent multi-agent prediction. Both cases can be considered as the first (left) and second (right) predicted modality for each agent. The right modalities showcase that the model has encoded and taken into account interactions between agents: both see another car at the crossing and infer that one of the modalities should slow down and stop for the other one. However, since the multimodal decoding has no consistency constraint, both agents predict crossing first as their first modality, and yielding as their second one. The usual minFDE_k loss will still be minimal for this prediction, no matter which agent actually crosses first.

Our goal is therefore to output a multimodal prediction for each agent, where the first modalities of each agents are consistent with each other, and so is the second, and so for every k_{th} modality. The definition for the performance metric of such consistency is proposed by ILVM [Casas et al., 2020] scene-level displacement errors minSFDE and SMR. To define these, we recall the usual marginal definition of the metrics minFDE_k and MR_k, which is averaged over agents after the minimum operation, which

means that the best modality of each agent is selected independently for each and then averaged:

$$\begin{aligned} minFDE_k &= \frac{1}{A}\sum_a min_k \|p_k^a - \hat{p}^a\|_2 \\ MR_k &= \frac{1}{A}\sum_a min_k \mathbb{1}^a_{miss\,k} \end{aligned} \tag{5.1}$$

For consistent scene multi-agent prediction, we report the joint scene-level metrics, where the average operation over the agents is done before the minimum operator. In this formulation, the minimum is taken over scene modalities, meaning that only the best scene (joint over agents) modality is taken into account:

$$\begin{aligned} minSFDE_k &= min_k \frac{1}{A}\sum_a \|p_k^a - \hat{p}^a\|_2 \\ SMR_k &= min_k \frac{1}{A}\sum_a \mathbb{1}^a_{miss\,k} \end{aligned} \tag{5.2}$$

In other words, the marginal metrics pick their optimal solutions in a pool of K to the power of A predicted solutions, while the joint metrics restrict this pool to only K possibilities, making it a much more complex problem.

5.2 Related works

While very little work has directly tackled multi-agent prediction and evaluation so far, multiple methods hint at the ability to predict multiple agents at the same time [Liang et al., 2020, Ivanovic et al., 2020, Zeng et al., 2021] even if they then focus on a more single-agent oriented framework. Other works [Alahi et al., 2016, Tang and Salakhutdinov, 2019, Rhinehart et al., 2019, Girgis et al., 2021] use autoregressive roll-outs to condition the future step of an agent on the previous steps of all the other agents. SceneTransformer [Ngiam et al., 2021] repeats each agent features across possible modalities, and performs self-attention operations inside each modality before using a loss computed jointly among agents to train a model and evaluate on the WOMD dataset [Ettinger et al., 2021] interaction track. ILVM [Casas et al., 2020] uses scene latent representations conditioned on all agents to generate scene-consistent samples, but its variational inference does not provide a confidence score for each modality, hence LookOut [Cui et al., 2021] proposes a scenario scoring function and a diverse sampler to improve sample efficiency. AIR^2 [Wu and Wu, 2021] extends Multipath [Chai et al., 2019] and produces a cross-distribution for two agents along all possible trajectory anchors, but it scales exponentially with the number of agents, making impractical for a real-time implementation that could encounter more than 10 agents at the same time.

The main other works proposing an actual solution to the joint multi-agent prediction problem are ILVM [Casas et al., 2020] and SceneTransformer [Ngiam et al., 2021].

ILVM [Casas et al., 2020] shown in Fig. 5.2 uses variational inference to learn a latent representation of the scene conditioned on each agent with a Scene Interaction Module, and decodes it with a similar Scene Interaction Module. The required number of modalities is obtained by sampling the latent space as many times as required. Even though the sampling is independent for each agent, the latent space is generated conditionally on all the agents. Since each sampled inference is evaluated and trained over all

the agents together, with likelihood of the whole scene according to the model, the network is effectively trained to provide scene-consistent samples.

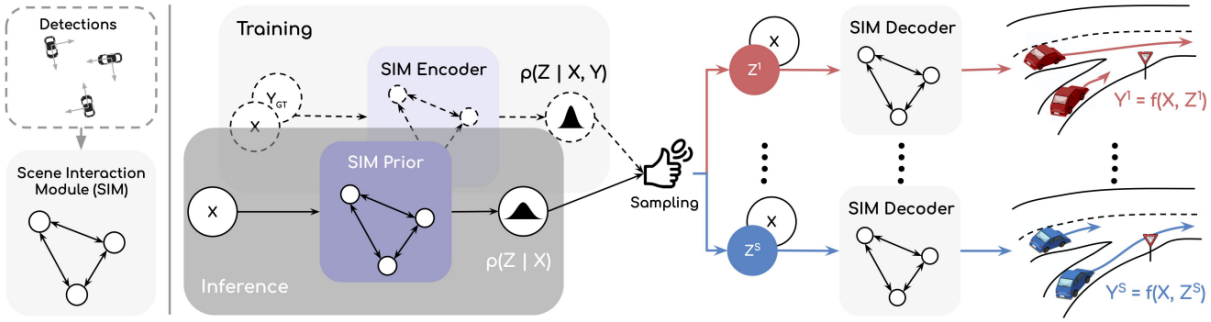


Figure 5.2: ILVM workflow. Latent values are sampled conditionally on the whole scene, and decoded taking into account the decoding of other agents.

SceneTransformer [Ngiam et al., 2021] duplicates the agent encoding with the number of modalities required and adds a one-hot encoding specific to each modality. They then apply a shared transformer architecture on all modalities to encode intra-modality interactions between agents and generate the final trajectories. The scene-consistency is obtained by simply using the joint version of minSFDE_k loss as descrived in Eq. 5.2 instead of Eq. 5.1 commonly used.

5.3 Consistent multi-agent heatmap-based forecasting

Since our heatmap output is a dense prediction and doesn't have direct discrete multi-modalities, we cannot add multi-agent consistency as a simple constraint loss on the heatmap output, as SceneTransformer did on their scalar output. Indeed, the heatmap represents all the possible positions of the agent with probabilities, and there is no notion of k^{th} modality we could synchronize in-between-agents. There is therefore a need for a post-processing operation applied during or after the endpoint sampling to insure multi-agent coherence.

5.3.1 Collision-free endpoint sampling

We first try to design a deterministic sampling algorithm based on the heatmaps generated in previous Chap. 4 in order to sample endpoints for each agent in a collision aware manner. We use the same sampling algorithm as in Sec. 4.3.1.1 based on MR optimization, but add a sequential iteration over the agents for each modality.

For a single modality k, we predict the possible endpoint of a first agent a by taking the maximum accumulated predicted probability under an area of radius r. We then not only set to zero the heatmap values of this agent heatmap $\mathcal{I}_{k'}^{a}$ around the sampled location (so not to sample it in the next modalities k'), but we also set to zero the same area on the heatmaps $\mathcal{I}_{k}^{a'}$ of the other agents a' on the same modality k, so that these other agents cannot be sampled at the same position for this modality. We describe formally this collision-free sampling in Algorithm 3.

This way, we try to enforce collision-free endpoints, and expect that considering collisions brings logic to improve the overall consistency of the predictions. However, as will be highlighted in Sec. 5.4.3,

Algorithm 3: MR Sampling Algorithm

input: Probability map $\mathrm{p}^a(x) foreachagent$a
K number of predictions
A number of agents R threshhold for Miss Rate
for k = *1..K* **do**
  for a = *1..A* **do**
    Find c_k^a maximizing $\int_{\|c_k^a - x\| < R} p^a(x)dx$
    for *a'* = *1..A* **do**
      Set $p^{a'}(x) = 0$ for all x such that $\|c_k^{a'} - x\| < R$
    end
  end
end

this methods significantly lowers the collision rate without the need for any additional learned model but it does barely improve the multi agent consistency. Indeed, considering the example cases in Fig 5.1 again, this sampling method takes care of the collision in the left example, but doesn't bring any improvement to avoid cases like the right one where there is no collision but the scene-level prediction remains inconsistent.

5.3.2 Modality recombination (THOMAS)

5.3.2.1 Reminder of the graph encoding and heatmap generation steps

In this part, we will first assume that heatmaps have been predicted marginally for each agent. To do so in an efficient and distributed way, we leverage the encoding from GOHOME and the hierarchical decoding from Sec. 4.4.3, but optimize it by sharing the scene encoding between all agents present in the scene and only having a separate flow for the heatmap decoding part as illustrated in Fig. 5.3. The resulting heatmaps are then sampled independently (not using the proposed modified collision-free algorithm from the previous section Sec. 5.3.1).

We use the same encoder as the GOHOME model [Gilles et al., 2021a]. The agent trajectories are encoded though $TrajEncoder$ using a 1D CNN followed by a UGRU recurrent layer [Rozenberg et al., 2021], and the HD-Map is encoded as a lanelet graph using a GNN $GraphEncoder$ made of graph convolutions. We then run cross-attention $Lanes2Agents$ to add context information to the agent features, followed by self-attention $Agents2Agents$ to observe interaction between agents. The final result is an encoding F_a for each agent, where history, context and interactions have been summarized. This encoding F_a is used in the next decoder operations and is also stored to be potentially used in modality recombination described in the next section 5.3.2.2. The resulting architecture of these encoding operations is illustrated in the first half of Fig. 5.3.

5.3.2.2 Modality recombination for multi-agent consistent prediction

We address the scene consistency after the initial endpoint deterministic sampling. As single agents metrics are usually noticeably better than joint ones, this hints us that a potential solution already exists in the individually sampled modalities, and only a re-ordering of these is needed.

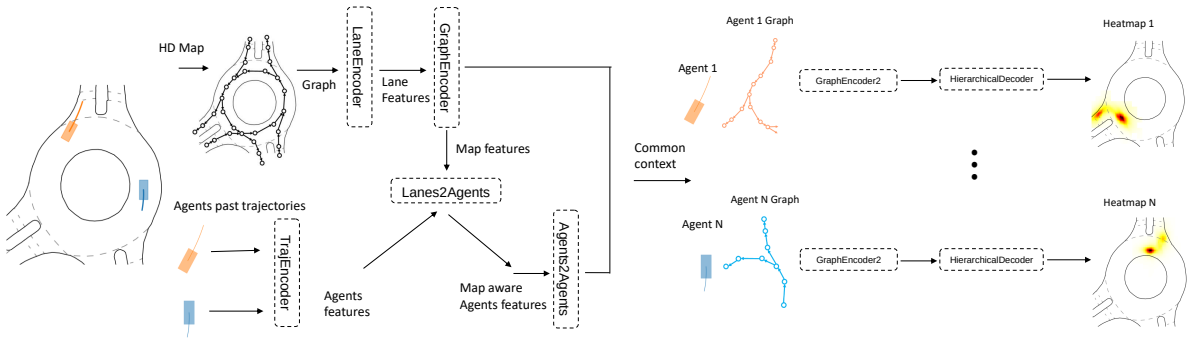


Figure 5.3: Model architecture for multi-agent prediction with shared backbone

Initially, the prediction output of the model can be defined as marginal, as all A agents have been predicted and sampled independently in order to get K possible endpoint each. Our goal is to output a set of scene predictions $\mathcal{J} = (L, A)$ from the marginal prediction $\mathcal{M} = (A, K)$, where each scene modality l belonging to $\mathcal{J}$ is an association of endpoints p_l^a for each agent a. To achieve this, our main hypothesis lays in the fact that good trajectory proposals are already present in the marginal predictions, but they need to be coherently aligned among agents to achieve a set of overall consistent scene predictions. For a given agent a, the modality selected for the scene l would be a combination of this agent available marginal modalities $p_k^a \in \mathcal{M}$. Each scene modality l would select a different association between the agents. In practice we are claiming that a consistent multi-agent prediction $\mathcal{J}$ could be achieved by smartly re-ordering $\mathcal{M}$ among agents. The resulting pipeline is detailed in Fig. 5.4.

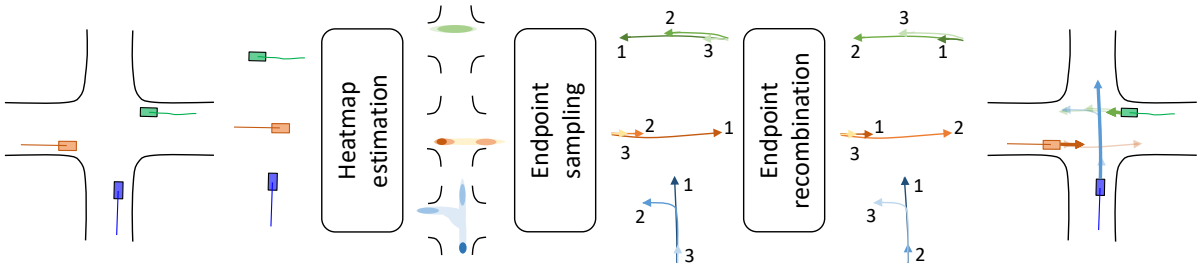


Figure 5.4: Illustration of the THOMAS multi-agent prediction pipeline

We illustrate our scene modality generation process in Fig. 5.5. The model learns L scene modality embeddings S_l of D features each. $K \times A$ agent modality vectors A_k^a are also derived from each agent modality endpoint. These vectors are obtained through a 2-layer MLP applied on the agent modality coordinates p_k^a , to which the stored agent encoding F_a (previously described in Sec. 5.3.2.1) is concatenated in order to help the model recognise modalities from the same agent. The scene modality vectors S_l are 'specialized' by querying the available modality proposals A_k^a of each agent through cross-attention layers. Subsequently, a matching score $c_l^{k,a}$ between each scene modality S_l and each agent modality A_k^a is computed. This matching score can be intuitively interpreted as the selection of the role (maneuver k) that the agent a would play in the overall traffic scene l:

$$c_l^{k,a} = S_l . A_k^{aT}$$

Since argmax is non-differentiable, we employ a soft argmax as a weighted linear combination of the agent modalities p_k^a using a softmax on the $c_l^{k,a}$ scores:

$$p_l^a = \sum_k softmax(c_l^{k,a}) p_k^a$$

The recombination module is trained with the scene-level minimum displacement error, where for each modality inside a single scene all predicted agent displacement losses are averaged before taking the minimum (winner-takes-all loss) over the L scene modalities, as defined in Eq. 5.2 (minSFDE$_k$).

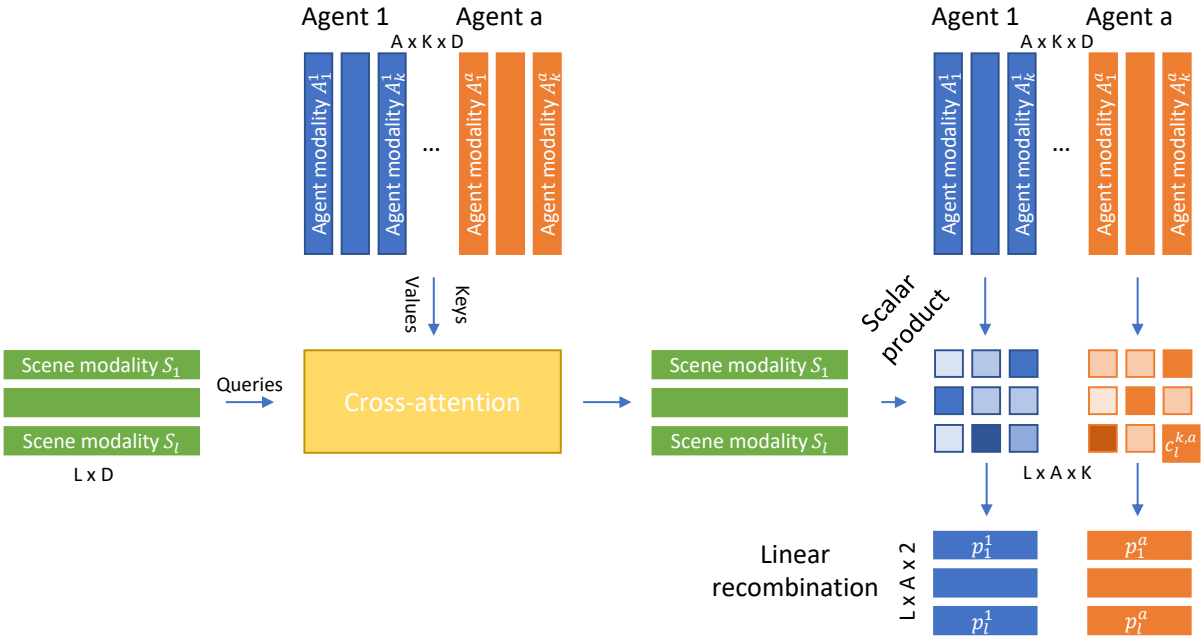


Figure 5.5: Illustration of THOMAS methods for generation of scene-consistent agent modalities

5.4 Experiments

5.4.1 Dataset

We use the Interaction v1.2 dataset that has recently opened a new multi-agent track in the context of its Interpret challenge. It contains 47 584 training cases, 11 794 validation cases and 2 644 testing cases, with each case containing between 1 and 40 agents to predict simultaneously.

5.4.2 Comparison with State-of-the-art

We compare our THOMAS model performance with other joint predictions methods ILVM [Casas et al., 2020] and SceneTransformer [Ngiam et al., 2021]. For fair comparison we use a GOHOME encoder for each of the method, and adapt them accordingly so that they predict only endpoints similar to our method. For each method, we focus on implementing the key idea meant to solve scene consistency and keep the remaining part of the model as close as possible to our approach for fair comparison:

Implicit Latent Variable Model We use a GOHOME encoder for the prior, posterior and decoder

Scene Interaction Modules. We weight the KL term with $\beta = 1$ which worked best according to our experiments.

Scene Transformer The initial paper applies a transformer architecture on a $[F, A, T, D]$ tensor where F is the potential modality dimension, A the agent dimension and T the time dimension, with D the feature embedding, with factorized self-attention to the agent and time dimensions separately, so that agents can look at each-other inside a specific scene modality. The resulting output is optimized using a jointly formalized loss. For our implementation, we get rid of the T dimension as we focus on endpoint prediction and coherence between the A agents. The initial encoded $[A, D]$ tensor is obtained with a GOHOME encoder, multiplied across the F futures and concatenated with a modality-specific one-hot encoding as in [Ngiam et al., 2021] to obtain the $[F, A, D]$ tensor. We then apply two layers of agent self-attention similar to the original paper, before decoding the endpoints through a MLP.

The results are reported in Tab. 5.1. While having comparable marginal distance performance (demonstrating that our model is not inherently more powerful or leveraging more information), THOMAS significantly outperforms other methods on every joint metric. SMR is improved by about 25% and SCR by almost 30%, leading to a combined cSMR decreased by also more than 25%.

Table 5.1: Comparison of consistent solutions on Interpret multi-agent validation track

	Marginal metrics			Joint metrics			
	mADE	mFDE	MR	mFDE	MR	SCR	cMR
ILVM [Casas et al., 2020]	0.30	0.62	10.8	0.84	19.8	5.7	21.3
SceneTranformer [Ngiam et al., 2021]	**0.29**	**0.59**	10.5	0.84	15.7	3.4	17.3
THOMAS	0.31	0.60	**8.2**	**0.76**	**11.8**	**2.4**	**12.7**

We also report our numbers from the Interpret multi-agent online test set leaderboard in Tab. 5.2 and Tab. 5.3. It is to be noted that the DenseTNT solution explicitely checks for collisions in the search for its proposed predictions, which we don't, hence their 0% collision rate (SCR) and its direct impact on consistent collision-free joint Miss Rate (cSMR).

Table 5.2: Results on Interpret multi-agent regular scene leaderboard (test set)

	minSADE	minSFDE	SMR	SCR	cSMR
MoliNet	0.73	2.55	44.4	7.5	47.4
ReCoG2 [Mo et al., 2020]	0.47	1.16	23.8	6.9	26.8
DenseTNT [Gu et al., 2021]	0.42	1.13	22.4	**0.0**	**22.4**
THOMAS	**0.42**	**0.97**	**17.9**	12.8	25.2

5.4.3 Ablation studies

5.4.3.1 Recombination module

We establish the following baselines to assess the effects of our THOMAS recombination.

Table 5.3: Results on Interpret multi-agent conditional scene leaderboard (test set)

	minSADE	minSFDE	SMR	SCR	cSMR
ReCoG2 [Mo et al., 2020]	0.33	0.87	14.98	0.09	15.12
DenseTNT [Gu et al., 2021]	0.28	0.89	15.02	**0.0**	15.02
THOMAS	**0.31**	**0.72**	**10.67**	0.84	**11.63**

Scalar output: we train a model with the GOHOME graph encoder and a multimodal scalar regression head similar to [Liang et al., 2020]. We optimize it with either marginal and joint loss formulation.

Heatmap output with deterministic sampling: we try various sampling methods applied on the heatmap, with either the deterministic sampling as described in Sec. 4.3.1.1 or a learned sampling trained to directly regress the sampled modalities from the input heatmap.

Compared to these baselines, THOMAS can be seen as a hybrid sampling method that takes the result of deterministic sampling as input and learns to recombine it into a more coherent solution.

We report the comparison between the aforementioned baselines and THOMAS in Tab. 5.4, where our approach THOMAS corresponds to the bottom line with combinatorial learned sampling.

With regard to the joint algorithmic sampling that only tackled collisions but has little to no effect on actual consistency, as highlighted by the big drop in collision rate from 7.2% to 2.6% in Tab. 5.4 but a similar joint SMR, THOMAS actually brings a lot of consistency in the multi-agent prediction and drops the joint SMR from 14.8% to 11.8% in Tab. 5.5. We also note that initializing the scalar joint training with the results of the scalar marginal training in order to correctly initialize multi-modality solves this problem and improves significantly the results (jointMR_6=15%), but choose not the report these numbers on the Table since they would be the result of a training twice longer and therefore unfair comparison to other methods and baselines.

Table 5.4: Comparison of consistent solutions on Interpret multi-agent validation track

			Marginal metrics			Joint metrics			
Output	Sampling	Objective	mADE	mFDE	MR	mFDE	MR	Col	cMR
Scalar	_	Marg	0.28	0.59	10.4	1.04	23.7	6.4	24.9
Scalar	_	Joint	0.34	0.77	16.2	0.90	19.9	49	21.7
Heat	Learned	Marg	**0.26**	**0.46**	4.9	0.98	20.9	4.1	21.9
Heat	Learned	Joint	0.29	0.58	9.8	0.88	15.2	3.0	16.4
Heat	Algo	Marg	0.29	0.54	**3.8**	0.83	14.8	7.2	15.9
Heat	Algo	Joint	0.29	0.54	**3.8**	0.83	14.8	2.6	15.6
Heat	Combi	Joint	0.31	0.60	8.2	**0.76**	**11.8**	**2.4**	**12.7**

Usually, scalar marginal models already suffer from learning difficulties as only one output modality, the closest one to ground-truth, can be trained at a time. Some modalities may therefore converge faster to acceptable solutions, and benefit from a much increased number of training samples compared to the others. This problem is aggravated in the joint training case, since the modality selected is the same for all agents in a training sample. The joint scalar model therefore actually fails to learn multi-

modality as illustrated by a higher marginal minFDE_6 than any other model, and an extremely high crossCollisionRate since some modalities never train and always point to the same coordinates regardless of the queried agent. Note that, despite a similar training loss, SceneTransformer doesn't suffer of the same pitfalls in Tab. 5.1 as is shares the same weights between all modalities and only differentiates them in the initialization of the features.

We isolate more closely the effects of our proposed recombination module in Tab. 5.5, where the first line is the raw marginal trained model without joint recombination, and the second line the result of our THOMAS learned recombination applied on top of this exact same prediction model.

Table 5.5: Ablation study of the recombination module

	Marginal metrics			Joint metrics			
	mADE	mFDE	MR	mFDE	MR	Col	cMR
No recombination	0.29	0.54	**3.8**	0.83	14.8	7.2	15.9
With recombination	0.31	0.60	8.2	**0.76**	**11.8**	**2.4**	**12.7**

5.4.4 Qualitative examples

In this section, we will mostly compare the model before recombination, which we will refer by the $Before$ model, to the model after recombination, referenced as the $After$ model. We display four qualitative examples in Fig. 5.6 with colliding modalities in the $Before$ model (in dashed orange) and the solved modality (in full line orange) after recombination. For each model ($Before$-dashed or $After$-full), the highlighted modality in orange is the best modality according to the SMR_6 metric among the 6 available modalities. We also display in dashed grey the other 5 predicted $Before$ modalities, and highlight that the recombination model indeed selects modalities already available in the vanilla set and reorders them so that non-colliding modalities are aligned together.

We also show more qualitative examples in Fig. 5.7, where we highlight the comparison in modality diversity between the $Before$ model (in dashed lines) and the $After$ model (in full lines). While the $Before$ model tries to spread the modalities for all agents to minimize marginal miss-rate, the recombined model presents much less spread compared to the original model, maintaining a multimodal behavior only in presence of very different possible agent intentions such as different possible exits or turn choices. For most other agents, almost all modalities are located at the same position, that is, the one deemed the most likely by the model. Thus, if the truly uncertain agents have to select the second or third most likely modality, the other agents still have their own most likely modality

5.5 Conclusion

We have presented THOMAS, a recombination module that can be added after any trajectory prediction module outputing multi-modal predictions. By design, THOMAS allows to generate scene-consistent modalities across all agents by making the scene modalities select coherent agent modalities and restricting the modality budget on the agents that truly need it. We show significant performance increase

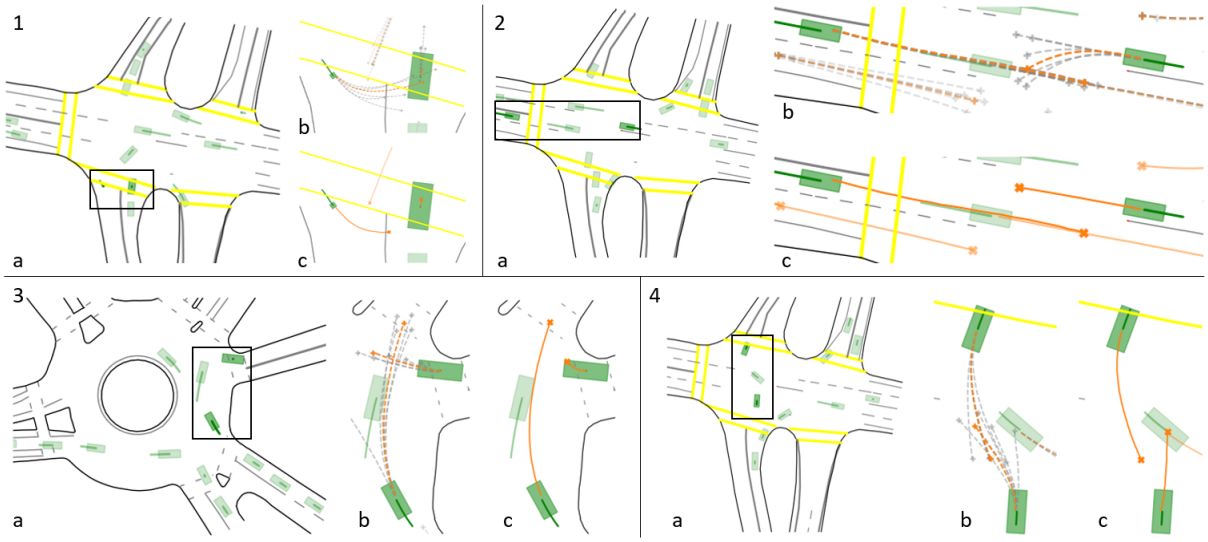


Figure 5.6: Qualitative examples of recombination model assembling collision-free modalities together compared to initial colliding modalities. For each example we display the general context with highlighted agents and area of interest, then two zooms in on the agents, one displaying the initial best modality before recombination in dashed orange and all the other available modalities in grey. The second zooms shows the best modality after recombination in full line orange.

when adding the THOMAS module compared to the vanilla model and achieve state-of-the-art results compared to already existing methods tackling scene-consistent predictions.

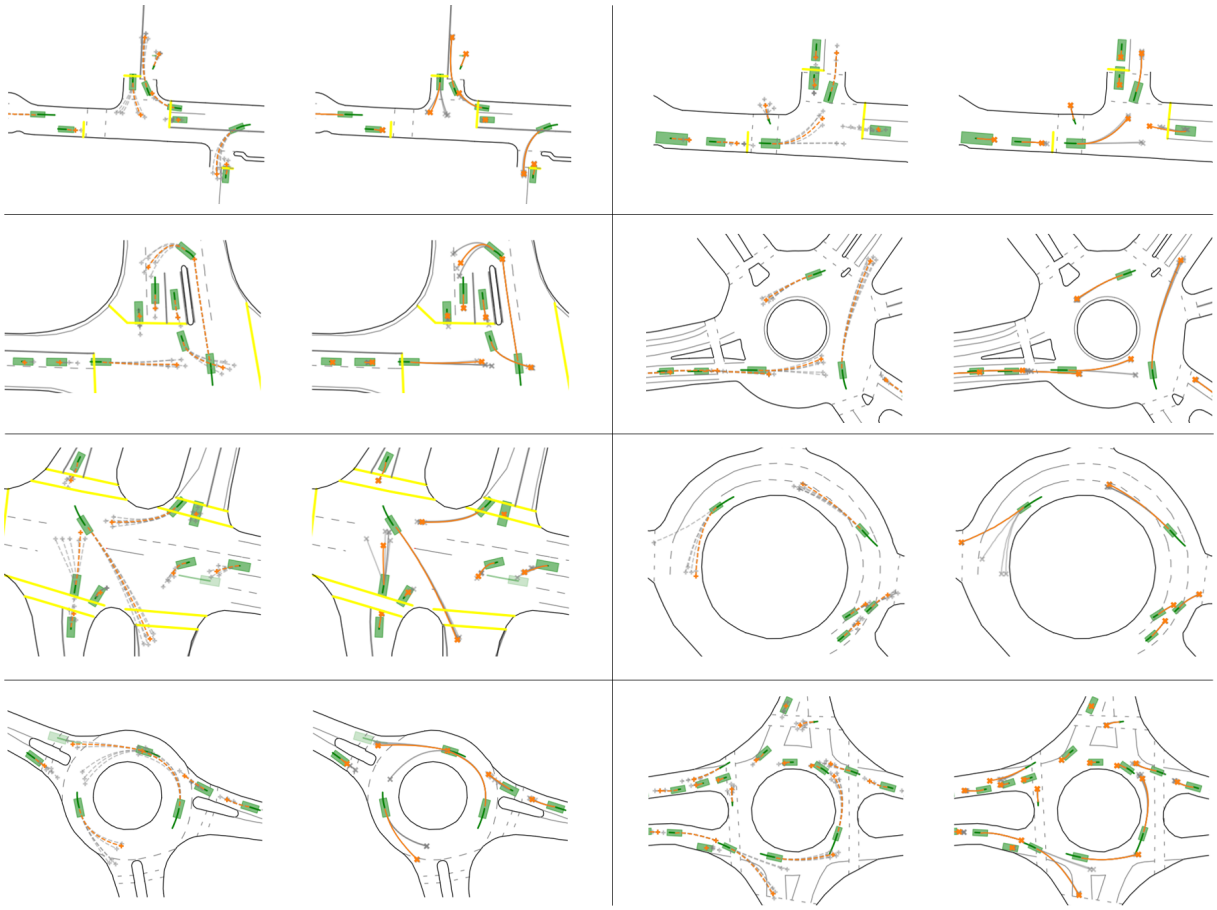


Figure 5.7: Qualitative examples of recombination model selecting fewer but more pronounced and impactful agent modalities compared to initial colliding modalities. For each example we display on the left the vanilla model modalities with dashed lines, with the initial best modality in dashed orange and all the other available modalities in grey. On the right we display the selected modalities after recombination, where the model focuses on the most likely occurrence in most agents.

Chapter 6

Expanding the evaluation of trajectory prediction

Contents

While the pursuit of performance and the adaptability to high complexity problems are important, both remain focused on very specific tasks, and very rarely is the relevancy of the evaluation process questioned.

In this chapter, we now study the robustness and interpretability of trajectory prediction models. We first conduct a cross-dataset analysis of generalizable performance, to observe which methods, but also which datasets, perform best in new unseen environments. We then tackle the topic of uncertainty estimation, how to calculate this uncertainty but also how to evaluate and use it in practice. Finally the check the calibration of our heatmap models to assess its interpretability.

6.1 Cross-dataset Generalizability

As learned methods grow in performance and popularity [Liu et al., 2021a, Gomes and Wolf, 2022, Karle et al., 2022] by either extending existing traditional methods [Jouaber et al., 2021] or replacing them completely [Mercat, 2021], so does the dependency to the data coverage these models are trained on. Such methods may encounter distributional shift due to changing geographical or weather conditions [Malinin et al., 2021]. It becomes therefore crucial to study the adaptability and performance of these methods across varying distributions.

Multiple trajectory prediction datasets [Zhan et al., 2019, Chang et al., 2019, Caesar et al., 2020, Malinin et al., 2021] have been used separately to train and evaluate motion estimation models, but few works actually study the performance of their models on more than one of these datasets at a time, and even more importantly, no study has yet been done to evaluate the representative coverage and generalization potential of the datasets across each other.

6.1.1 Related work

Recently, more reflection has been carried out on the ways of evaluating these trajectory prediction methods. Some argue that motion estimation should be evaluated with regards to its downstream effect on the planner [Ivanovic and Pavone, 2021b, Ivanovic and Pavone, 2021a, McAllister et al., 2022], while others focus on their lack of generalization to new scenarii [Bahari et al., 2022]. Similar cross-datasets studies have been conducted for fields related to autonomous driving such as human intention [Gesnouin et al., 2022] or detection [Hasan et al., 2022].

6.1.2 Trajectory prediction methods

In order to be representative of the wide scope of existing trajectory prediction methods, we compare two state-of-the-art baselines both representative of the different possibilities for output formulation, i.e. scalar coordinates output or probability heatmap output.

SceneTransformer [Ngiam et al., 2021] is one of the most recent trajectory prediction model regressing multiple scalar trajectories using a transformer architecture. In its encoding phase, it retains the time dimension across all agents present in the scene, and applies factorized self-attention either across agents or time, as well as cross-attention onto the map context. It uses modality one-hot embeddings and a transformer decoder to predict multiple modalities, so that it can share the decoding weights of the multiple futures that are trained using a Winner-Take-All loss as in most scalar output methods [Cui

et al., 2019, Chai et al., 2019, Liang et al., 2020, Narayanan et al., 2021, Varadarajan et al., 2022]. We reimplement a similar architecture with the same number of layers as in the original paper but with a smaller hidden dimension D=128 to make it fit on a single GPU and be more comparable to our second baseline in parameter size and training time.

GOHOME is part of a growing class of methods using occupancy grids [Kim et al., 2017, Park et al., 2018, Hong et al., 2019, Kurbiel et al., 2020, Ridel et al., 2020, Mangalam et al., 2021, Gilles et al., 2021b, Casas et al., 2021, Gu et al., 2021, Gilles et al., 2022, Schäfer et al., 2022, Mahjourian et al., 2022]. The occupancy grid usually represents a probability distribution in the form of a heatmap describing the possible future locations of the vehicle at the end of the prediction horizon T. Given the predicted heatmap, a set of final future positions is sampled. In a final step, for each sampled locations, the full trajectory is regressed [Gilles et al., 2021a]. In order to sample the possible future locations from the heatmap, usually a Non-Maximum Suppression (NMS) method is applied [Gu et al., 2021, Gilles et al., 2021b, Schäfer et al., 2022]. This NMS requires a sampling radius parameter r to determine how far apart the sampled endpoints should be from each other.

We apply some slight modifications to the GOHOME architecture to adapt it to our case analysis. First, since some datasets don't provide connectivity information between lanes [Malinin et al., 2021], we replace the graph convolutions with global attention, in a VectorNet-like manner as in [Gu et al., 2021, Ngiam et al., 2021, Pustynnikov and Eremeev, 2021]. We also replace the lane-based heatmap decoder with the hierarchical sparse grid decoder from [Gilles et al., 2022] for faster inference and once again independence from the HD-Map connectivity information

6.1.3 Datasets and Metrics

We evaluate performance on the widely used trajectory datasets Argoverse [Chang et al., 2019], Interaction [Zhan et al., 2019], NuScenes [Caesar et al., 2020] and Shifts [Malinin et al., 2021], all focusing on car trajectories. These benchmarks have slightly different initial settings as described in Tab. 6.1. Namely, the history and prediction horizons are not always the same, and can be sampled at different rates. For fair evaluation and transferability, we standardize these datasets to always use 1s of history and predict 3s in the future. We also interpolate the trajectories to resample them at 10Hz each.

Table 6.1: Dataset settings

Dataset	Argoverse	Interaction	NuScenes	Shifts
History (s)	2	1	2	5
Prediction horizon (s)	3	3	6	5
Frequency (Hz)	10	10	2	5
Training size	200k	400k	30k	5M

In our analysis we consider the well established multimodal metrics minFDE_l and MR_l [Zhan et al., 2019, Chang et al., 2019]. minFDE_l represents the minimum final displacement error at time horizon T over the l top-ranked trajectories. MR_l represents the percentage of samples in the dataset on which the ground truth future position of the target agent at time horizon T is farther than 2m from any of the l top-ranked predicted trajectories.

6.1.4 Cross-dataset evaluation

We analyze here the trajectory prediction performance of both models presented in section 6.1.2 when they are trained on the training split of one dataset and tested on the validation splits of all datasets.

6.1.4.1 Training details

We trained each model for 50 epochs of 2000 iterations each, with a batch size of 64. The GOHOME hyper-parameter r related to the sampling radius has been optimized on the training-split of the training dataset and kept unchanged for the test datasets. Few data augmentation schemes were employed to optimize the generalizability performance. First, all models are trained with random rotations to prevent overfitting on the current car heading measurement. Furthermore, we noticed that the Argoverse dataset does not present any case where the target agent to be predicted has a speed lower than 1 m/s. Contrarily, other datasets include vehicles to be predicted that stand still or with very low speed values for the whole prediction horizon. For this reason, it has been necessary to augment the Argoverse dataset with prediction samples related to vehicles present in the scenes that move slowly or are parked other than the predefined target. Without this augmentation procedure, models trained on the plain Argoverse ends up with poor generalization performance on other datasets.

6.1.4.2 Results

We report onto Fig. 6.1 the cross-dataset performance matrices for minFDE_6 and MR_6 for both prediction models. The label on the rows indicates the dataset used for training while the label on the columns represent the target test datasets. The numbers in the matrix corresponds to the performance measured on the validation split of the corresponding target dataset. We use a correlation matrices visualization with unified colormap for better highlighting of the comparison with good and poor performances, but also provide the same tabular results for minFDE_6 in Tab. 6.2.

As expected, the best performance are visible on the diagonals, since both models perform better when tested on data coming from the same distribution as the training.

We observe that Argoverse training exhibits the smallest loss of performance when tested on other datasets. We also observe that despite its relatively short size (only 30k samples), the training on NuScenes also performs reasonably on other datasets, whereas when trained on Interaction, the models performs poorly on every other domain. We attribute the poor performance of Interaction to its different data collection and processing, done with top-view images from drones instead of the usual perception pipeline from the autonomous vehicle. Therefore Interaction is trained on an almost perfect object detection and tracking, and does poorly on other datasets filled with detection inaccuracies and tracking jumps caused by occlusions. Surprisingly, despite its superior sample size, training on Shifts doesn't provide better transferability performance compared Argoverse and NuScenes.

Data Augmentation on Argoverse We mentioned in Sec. 6.1.4.1 the need to include non-target agents in the training data of Argoverse to correctly generalize to other datasets. We illustrate here this distributional gap in Fig. 6.2, where we display the average speed of the target agent during the future to be predicted. We can therefore observe that Argoverse has little to no agent that stay stationary, compared to other datasets. This mostly comes from the sample selection in Argoverse [Chang et al., 2019],

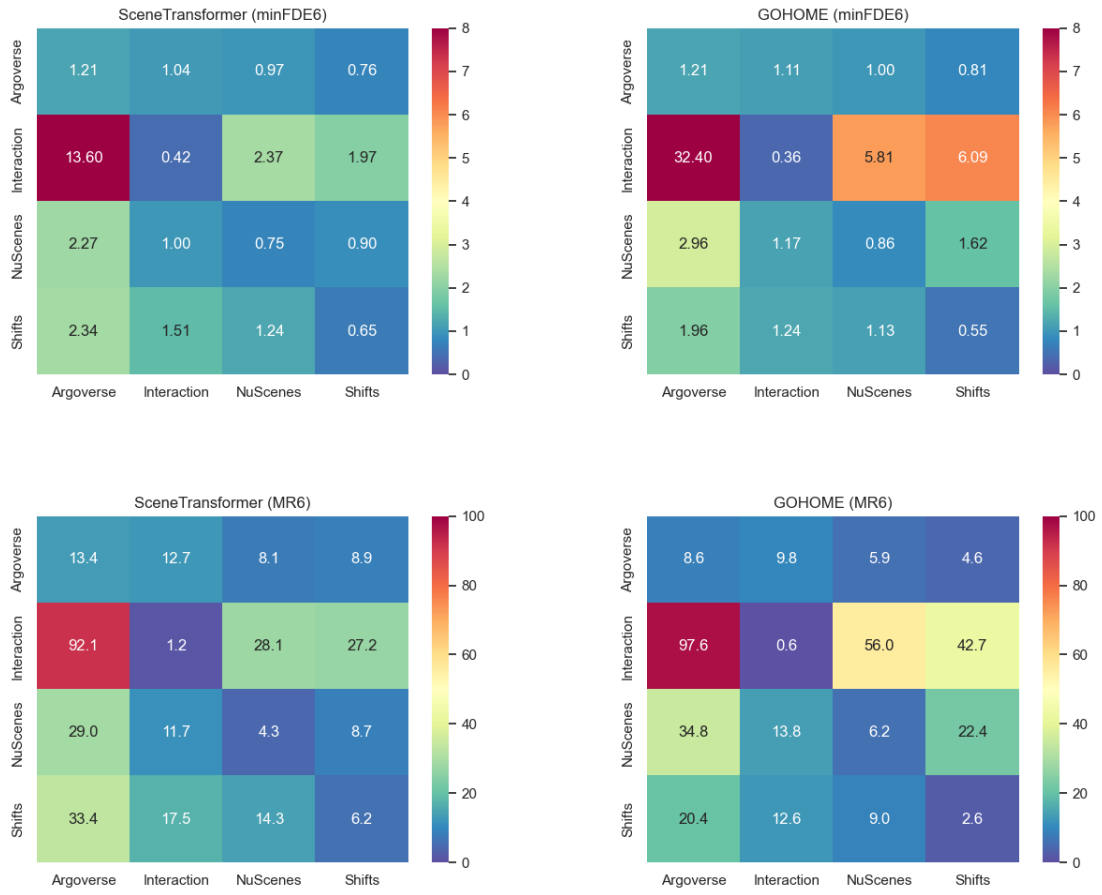


Figure 6.1: Prediction performance in a cross-datasets evaluation setting. The rows (left label) refer to the TRAIN dataset, while the columns (bottom label) refer to the TEST dataset.

where "interesting" samples have been selected according to some hand-designed criteria. Understandably, stationary cars were not deemed to have interesting trajectories and were therefore not selected. However, in a real life application, parked cars need to be predicted as such, otherwise models trained on exclusively non-stationary cars as in Argoverse will systematically predict stopped cars to start moving again at a certain minimum speed. Since the Argoverse data also contains non-target surrounding agents that were not subjected to this selection, we use the surrounding agents with sufficient historical data to enrich the model training.

This leads to the performance gap observed in Fig. 6.3a, where the model trained strictly on Argoverse has way higher errors on the other datasets, and notably on the Shifts dataset which has a very high proportion of stationary samples. However, when we include a random sampling of 30% of agents other than the predefined target, the resulting speed distribution reported in Fig. 6.3b is much more representative of lower speed cases, and transfers much better onto the other datasets without losing performance on Argoverse itself.

Ideal mixed training In order to assess ideal cross-dataset performance, for completeness we also present the results that are obtained when training the models on all the available datasets at the same time. To achieve this, each sample loaded during training is drawn randomly from one of the 4 datasets,

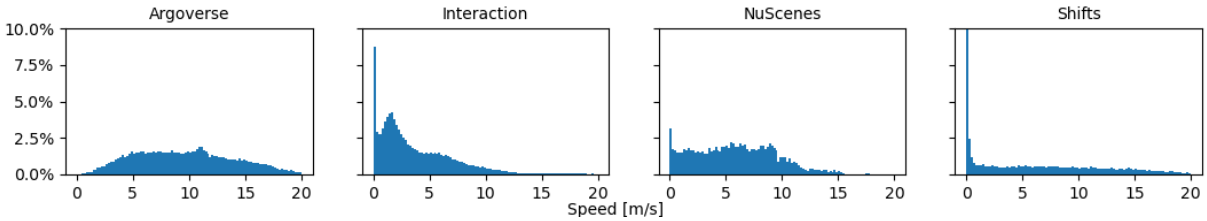
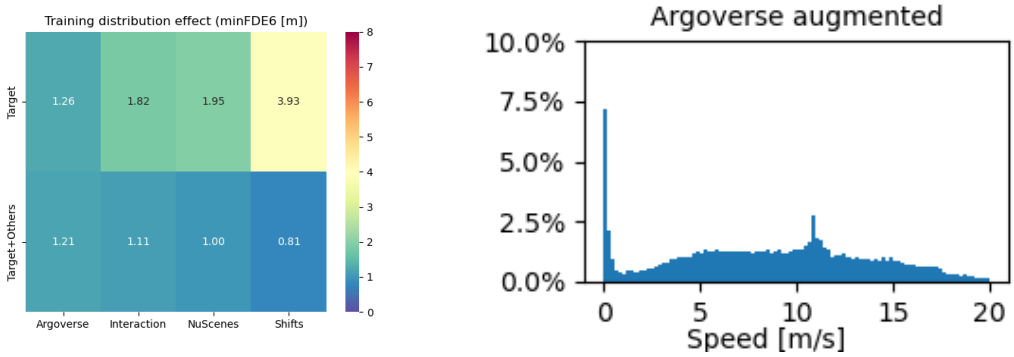


Figure 6.2: Distribution of average speed between initial agent position and last future position. Shifts reaches more than 40% samples of 0 m/s average speed, this bin is therefore out of scale for easier cross-dataset comparison.



(a) Performance difference

(b) Speed distribution after augmentation

Figure 6.3: Impact of incorporating non-target agents in Argoverse to demonstrate slow-moving behaviors

with equal probability. We report the results of this training across all datasets in Tab. 6.2, as well as the performance of the individual dataset trainings from Fig. 6.1 for better comparison.

The resulting mixed trained model has very close to best performance on each of the dataset compared to single dataset models evaluated on the same dataset. The mixed model sacrifices a bit of specificity but gains a lot of generalizability across all datasets. In some cases, training on all the datasets even improves the best performance trained only on this dataset: SceneTransformer trained on all datasets performs better on Shifts than SceneTransformer trained only on the Shifts dataset.

The main first conclusion we can draw from this cross-dataset performance is that it is not so much the size of the data that matters, rather than its ability to faithfully represent real conditions.

When comparing the performance between the heatmap-based and the scalar-based models, we can notice how the heatmap output provides the best MR on the training datasets (with the exception of NuScenes) while scalar output provide the best minFDE. Regarding the transferability performance, SceneTransformer present the smallest performance loss compared to GOHOME when tested on other datasets.

Finally, such an evaluation allows us to have a first impression of the relative difficulty of each dataset. The main conclusion seems to be that Argoverse dataset is the more difficult to predict, and Interaction the simplest. Many interpretations can be submitted as to the reason for this difficulty difference, going from the different type of intersection layout, to the amount of interactions in the recorded scenes and the perception quality varying in-between datasets.

Noise distribution

Table 6.2: Prediction performance minFDE_6 in a mixed dataset training setting

	Training	Argoverse	Interaction	NuScenes	Shifts
GOHOME	Argoverse	1.21	1.11	1.00	0.81
GOHOME	Interaction	32.40	0.36	5.81	6.09
GOHOME	NuScenes	2.96	1.17	0.86	1.62
GOHOME	Shifts	1.96	1.24	1.13	0.55
SceneTransformer	Argoverse	1.21	1.04	0.97	0.76
SceneTransformer	Interaction	13.60	0.42	2.37	1.97
SceneTransformer	NuScenes	2.27	1.00	0.75	0.90
SceneTransformer	Shifts	2.34	1.51	1.24	0.65
GOHOME	Mixed	1.34	0.66	0.88	0.70
SceneTransformer	Mixed	1.33	0.58	0.81	0.58

We hypothesize that one of the non-exhaustive reason for this difficulty reason comes from the perception noise present in each dataset. In order to estimate perception noise in each dataset, we filter each trajectory with a Kalman filter and report the maximum displacement between the raw trajectory and the filtered one. We report the resulting noise distribution in Fig. 6.4 and notice that the Interaction distribution is shifted towards lower noises than the other datasets, while Argoverse reaches higher noise values. These differences may explain the poor performance of the Interaction-trained model on other datasets, as well as the higher difficulty on the Argoverse dataset

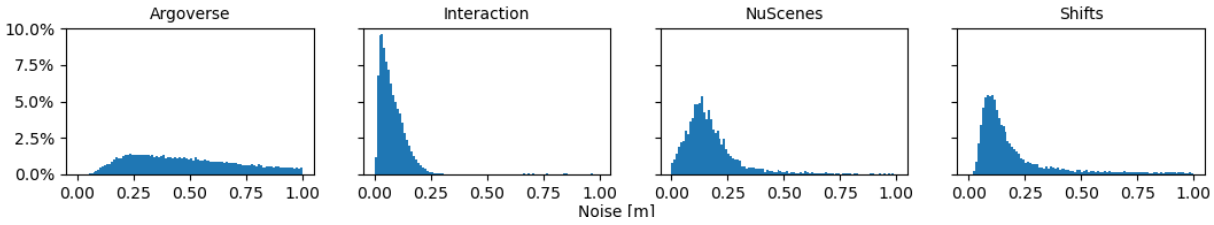


Figure 6.4: Distribution of perception noise across each dataset

6.1.5 Unsuccessful trials

Following the noise observations made in Sec. 6.1.4.2, we tried to augment the training data on Interaction with synthetic perception noise to bridge the gap to the other datasets. We were however not able to gain any kind of significant performance this way. This failure may be due to the way we modelled perception noise (independent Gaussian noise at every timestep) that could be inappropriate, or to the fact that the performance gap is due to other factors other than input noise.

We also noticed a difference in speed distribution in Fig. 6.2 that reaches a lower upper limit (approx. 12.5 m/s) in Interaction compared to other datasets (although NuScenes also has a similarly limited distribution), and tried global random scaling (multiplying all coordinates by a global factor between 0.75 and 1.25 as in [Ye et al., 2021]) to simulate higher speed, but this didn't bring much improvement either.

We hypothesize that the remaining performance gap when training on Interaction may be due to

overfitting on the limited number of maps, as Interaction has a discrete set of relatively small intersection maps compared to other dataset maps that scale closer to city sizes, but didn't explore this hypothesis further.

6.2 Uncertainty estimation

In this section, we present a method to leverage the heatmap output formulation of models like GOHOME in order to estimate how much the model is uncertain when performing a trajectory prediction. We first present the formulation of the uncertainty estimation and in a second step we show how the uncertainty can be utilized to improve prediction performance.

6.2.1 Evaluation of uncertainty prediction

We first define the framework within which we can evaluate the quality of our uncertainty prediction. We will define here the different methods we will use to evaluate the uncertainty estimation, as this will also enable us to define better the actual task and justify some design choices made.

6.2.1.1 Error correlation

Classification uncertainty is relatively straightforward to evaluate, through checking that for a given confidence bin value, the amount of predictions being correct is close in percentage to the confidence value, as illustrated in Fig. 6.5, ie the confidence ($confidence = 1 - uncertainty$) is representative of the expected failure percentage.

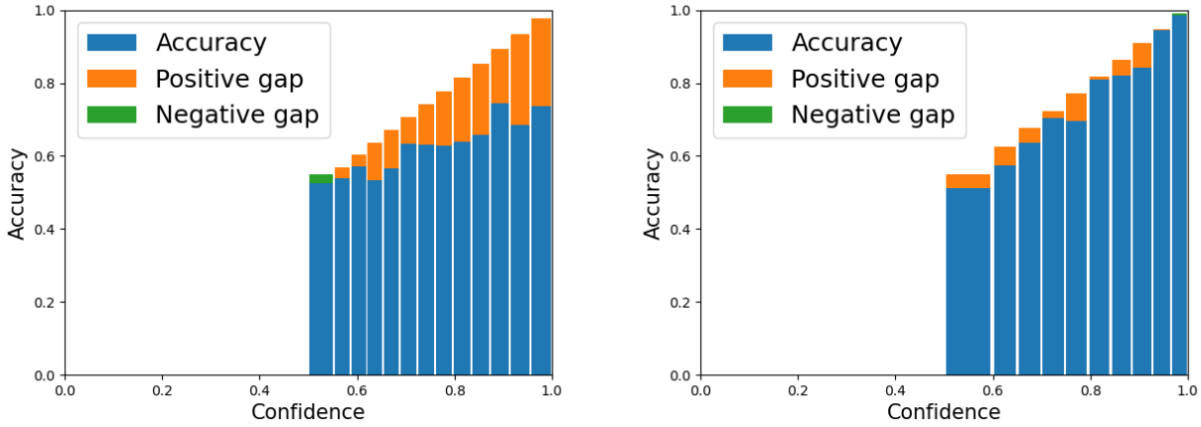


Figure 6.5: Reliability diagram from [Gesnouin, 2022]. For each x confidence bin, the accuracy y value is the percentage of cases correct amongst cases within that confidence interval. The closer the diagram is to the identity function, the better (here right is much better than left).

However in our case, regression uncertainty has no direct connection between uncertainty and percentage of failure, as opposed to classification uncertainty. The main goal of uncertainty estimation is to provide an appraising of the prediction quality, so that complementary measures can be taken when the uncertainty goes above a certain threshold. The intended evaluation should then check that higher error cases also have higher uncertainty than lower error cases, effectively checking the ranking ability of the uncertainty. [Lakshminarayanan et al., 2017, Malinin, 2019] therefore propose to look at *error-retention curves*, which ranks predicted samples by uncertainty, and then replaces a fraction of the dataset with ground truth instead of the prediction. The retained fraction of predicted samples is called retention fraction. For this retention fraction, the error metric (eg minFDE or CNLL) can be measured, and a curve called *error-retention curve* can then be drawn for every parsed value of this retention fraction, as

illustrated in Fig. 6.6. This means that the curve point at 0% will be at 0 error (only ground truth) and the point at 100% will be at the error usually measured on the dataset by the prediction method.

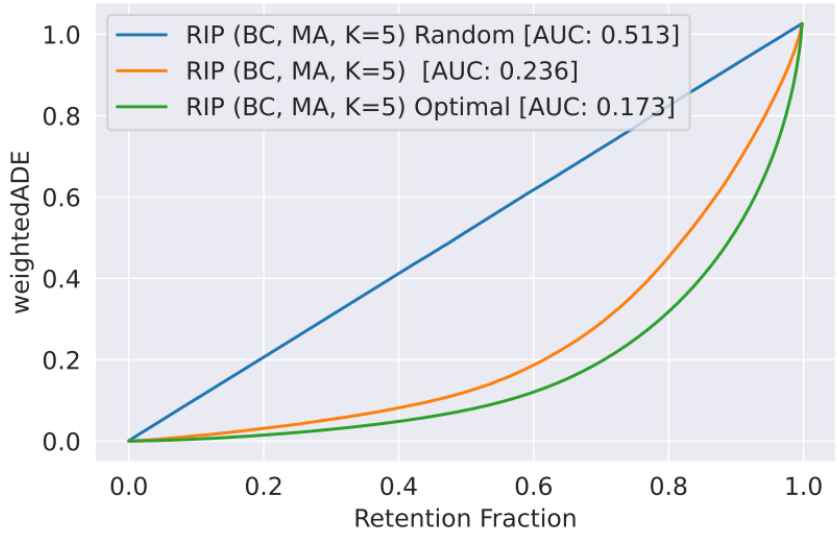


Figure 6.6: Error-retention curves from [Malinin et al., 2021]. For a given retention fraction, the selected wADE metric is calculated on the predicted set by replacing the higher uncertainty values with the ground truth. The worst case uncertainty prediction (random) and the best optimal case (perfect ranking) are illustrated in blue and green respectively.

As a result, the lower the curve the better it is. The Area Under the Curve (AUC) is therefore used as an aggregated metric to characterize the global estimation performance.

For uncertainty evaluation in this framework, we will use the dataset Shifts [Malinin et al., 2021] designed specifically for that purpose and for the analysis of distributional shift between training and testing data, where uncertainty can be used to detect these shifts and take appropriate counter-measures. Shifts uses R-AUCcCNLL as their main scoring metric, which is the Area Under the Retention Curve of the corrected Negative Log-Likelihood error. The CNLL error is defined as:

$$CNLL = -\log(\sum_k w_k \exp(-\|p_k - p_{gt}\|^2)) \tag{6.1}$$

Shifts [Malinin et al., 2021] uses this cNLL metrics as they argue it to be more suited to multimodal predictions than wADE_k which suffers from mode collapse as its optimal solution when hesitating between two equally possible modes where the metric minimum is reached by averaging the two possble modes, while cNLL optimal solution consists in two separate modes as expected.

Another more qualitative analysis of uncertainty correlation with error correlation can be to simply display the plot of error with regard to uncertainty, which can help identifying correlation and highlighting domains where the uncertainty stops being relevant, as we will illustrate later in Fig. 6.10.

6.2.1.2 Error improvement

While providing a direct analysis of the performance of our uncertainty prediction, the correlation described in the above section doesn't provide any suggestions as to how the uncertainty should actually

be used in practice, namely in order to improve the prediction. In the next sections, after detailing our uncertainty estimation method, we will propose in Sec. 6.2.3 an approach for improving our trajectory prediction with this uncertainty, which will enable us to simply evaluate in Sec. 6.2.4.2 our uncertainty performance through the final performance improvement it brings to the full motion forecasting pipeline.

6.2.2 Uncertainty formulation

As argued in Sec. 6.2.1.1, our goal is to design a good error estimator. We formalize the predicted heatmap $H(p)$, where H is the function that for the pixel/position p returns the predicted probability value of the agent being at this position in the future. Given an error function $\mathcal{E}$, that for a prediction p_{pred} and a ground truth p_{gt} returns the error $\mathcal{E}(p_{pred}, p_{gt})$, and the aforementioned prediction p_{pred}, we can then compute the expected error value:

$$\mathbb{E}(\mathcal{E}) = \sum_{p} H(p)\mathcal{E}(p_{pred}, p) \tag{6.2}$$

The whole process is illustrated in Fig. 6.7. This can then also apply to multimodal predictions p^k_{pred} with adapted multimodal metrics $\mathcal{E}$.

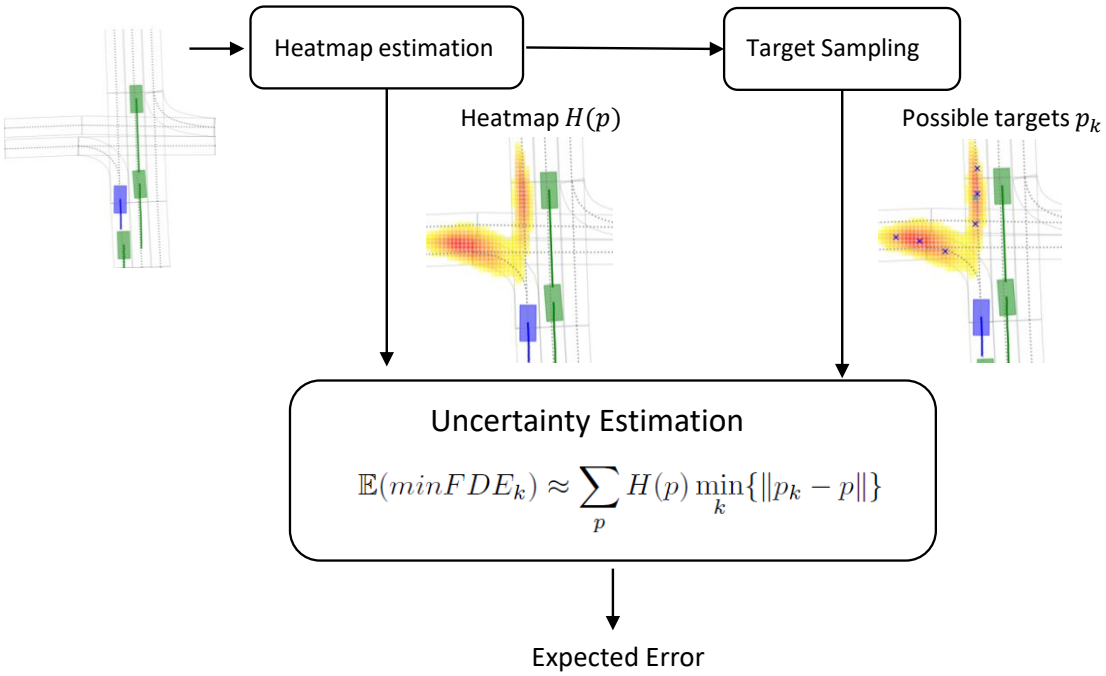


Figure 6.7: Uncertainty estimation from heatmap using expected error estimation

Given the future probability heatmap, we are therefore able to compute the expected values of any desired metric according to the probability distribution from the heatmap. We study here a few possible expected metrics we could compute to estimate the uncertainty of a prediction.

Variance As a general rule, the more spread the predicted heatmap will be, and the higher the corresponding uncertainty should be. Therefore the most straightforward metric to evaluate the uncertainty

would be the variance V of this heatmap:

$$\begin{aligned} E &= \sum_p H(p)p \\ V &= \sum_p H(p)\|p - E\|^2 \end{aligned} \tag{6.3}$$

where we iterate over the positions p of the heatmap, with the probability heatmap value $H(p)$ for a given position. Despite being a bit outlier-sensitive due to the square distance, and not taking multi-modality nor the actual prediction into account, this metric gives already a remarkably good and simple proxy for uncertainty, and has the advantage of not depending on the sampling result, which we will use later.

Corrected negative log-likelihood (CNLL) Another intuitive metric to estimate would be the CNLL itself, since it is the actual metric used for evaluation in Shifts [Malinin et al., 2021].

$$\begin{aligned} CNLL &= -\log(\sum_k w_k \exp(-\|p_k - p_{gt}\|^2)) \\ \mathbb{E}(CNLL) &= -\sum_p H(p) \log(\sum_k w_k \exp(-\|p_k - p\|^2)) \end{aligned} \tag{6.4}$$

Note that we only evaluate it on the last trajectory point at T=5s instead of the full trajectory, as we approximate that the error on the full trajectory is proportional to the error at the last point. However, if the variance was already a bit outlier sensitive, CNLL is extremely sensitive, as it applies an exponential on the square distance, before multiplying by the probability and applying a log function.

A bit more insight into the CNLL metric Let's consider a set of sampled endpoints p_k with their corresponding modality probabilities w_k. Assuming that the endpoints were sampled with a radius R of 2m, the closest modality k_{min} to the ground truth would be about 2m closer than the other modalities at minimum. As a consequence, after exponential inside the CNLL metric, the modalities other than k_{min} would be a coefficient e^{-4} at least smaller than k_{min}, making them non-significant. Therefore the CNLL can be approximated as:

$$CNLL \approx \|p_{k_{min}} - p_{gt}\|^2 - \log(w_{k_{min}}))$$

which will already be a bit less outlier-sensitive. Since $\|\hat{p}_{k_{min}} - p\|^2$ and $\|\hat{p}_{k_{min}} - p\|$ share the same monotonicity, this motivates our next expected metric.

minFDE$_k$ is a more simple and robust estimator than its square equivalent, and matches well will trajectory prediction metrics used in traditional benchmarks:

$$\mathbb{E}(minFDE_k) \approx \sum_p H(p) \min_k\{\|p_k - p\|\} \tag{6.5}$$

The result is a very simple and robust expected metric that gives a very good uncertainty estimation.

The proposed uncertainty formulation is based on the fact that prediction methods designed to produce an heatmap provide a natural intrinsic uncertainty estimator in the spread of their output. We use the variance of the predicted spatial probability distribution as an indicator of the model uncertainty U:

$$U = \sum_p H(p)\|p - E\|^2 \quad \text{with} \quad E = \sum_p H(p)p \tag{6.6}$$

where we iterate over the positions p of the heatmap and we indicate with $H(p)$ the probability value for the given position. E corresponds to the expected value of the probability distribution described by the heatmap. With this formulation, we claim that the heatmap provides for free an unconstrained and non-parametric measure of uncertainty without the need of adding and training a model part specific to uncertainty prediction as in [Pustynnikov and Eremeev, 2021] and [Postnikov et al., 2021].

6.2.3 Controlling prediction diversity with uncertainty

GOHOME outputs a heatmap estimating the probability distribution of the position of the target agent, onto which we apply a Non-maximum Suppression (NMS) method to sample the desired number of endpoint modalities. This NMS requires a sampling radius parameter r to determine how far apart the sampled endpoints should be from each other. We illustrate in Fig. 6.8 the effect of this radius on the sampling.

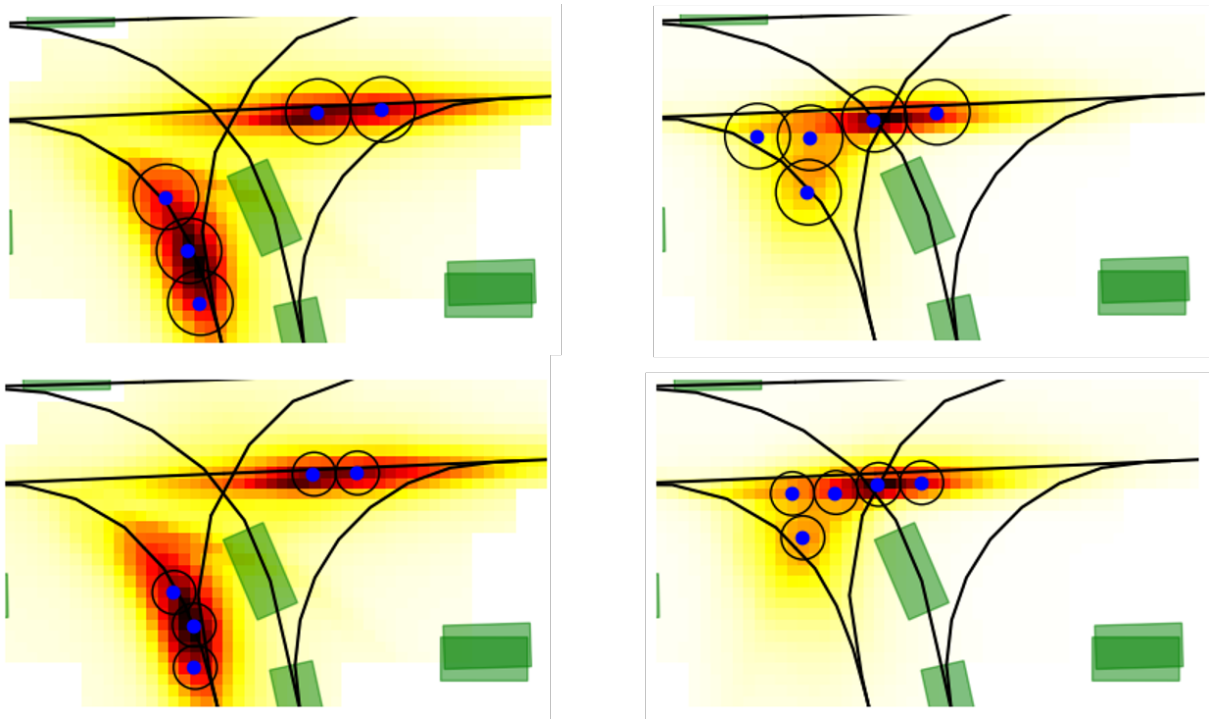


Figure 6.8: **Left column**: High uncertainty heatmap. **Right column**: Low uncertainty heatmap. **Top line**: high sampling radius. **Bottom line**: low sampling radius. As seen in the bottom left, using a low radius for a very spread heatmap leads to uncovered areas that may account for missed predictions. On the other hand, setting a high radius on a very focused heatmap spreads the sampled endpoints more than necessary and may generate a higher error if the ground truth is in-between two sampled points.

As seen in the Fig. 6.8 above, given a fixed number of future modalities, the distance between these future points should be adapted with regards to how spread the heatmap is, which correlates with the uncertainty of the model. We demonstrate this correlation further in Fig. 6.12, where we plot for each dataset, for the model trained on this dataset, the average optimal radius (according to the minFDE_6 metric) for uncertainty values grouped into integer bins.

We leverage the presented uncertainty estimation to control the diversity of the predicted future locations at the prediction horizon T. Intuitively, when the network is more uncertain, in order to minimize

the prediction error it is required to increase the diversity of the predictions to cover a wider span of possibilities. In practice we control this behavior, by adapting the sampling radius r presented in section 4.3.1.1 to adjust the diversity of the sampled locations to the spread of the heatmap. This behavior is depicted in Fig. 6.9 where a bigger sampling radius is employed on the left example to cope with a more uncertain prediction testified by the bigger spread in the heatmap.

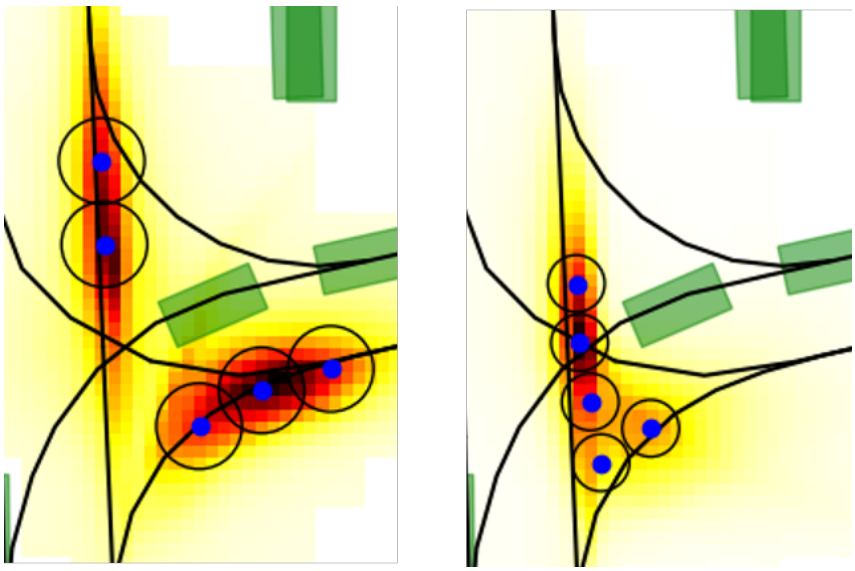


Figure 6.9: Endpoint sampling with different radii adapted to the uncertainty of the model

This method also enables us to illustrate the uncertainty evaluation approach hinted at in Sec. 6.2.1.2, where the uncertainty quality can be tested by recording the performance improvement it brings.

6.2.4 Results

6.2.4.1 Uncertainty as prediction error estimator

To motivate our uncertainty definition, we first show how the uncertainty estimated through Equation 6.6 is correlated with the prediction error that the model ends up making. We start these experiments using only the Variance proxy from Eq. 6.3 for simplicity. Figure 6.10 shows the average prediction error minFDE_1 of the GOHOME model for uncertainty values grouped into integer bins. In the case of GOHOME, minFDE_1 represents the error made by a single prediction on the most probable location highlighted by the heatmap. The prediction error is calculated on the validation split of each dataset in consideration.

We can clearly see a strong correlation between uncertainty and prediction error testifying that heatmap based methods intrinsically carry a notion of their performance when making a prediction inference. It is interesting to notice how a similar trend is maintained even when the analysis is done cross-dataset, i.e. when the model is evaluated on a dataset different from the training dataset.

As we use the same x scale for every cross-dataset combination, we can observe quite various uncertainty distribution from one dataset to another.

A more precise comparison is done in Tab. 6.3, where we compare the R-AUC from the error-retention curves performance between the 3 uncertainty proxys defined in Sec. 6.2.2.

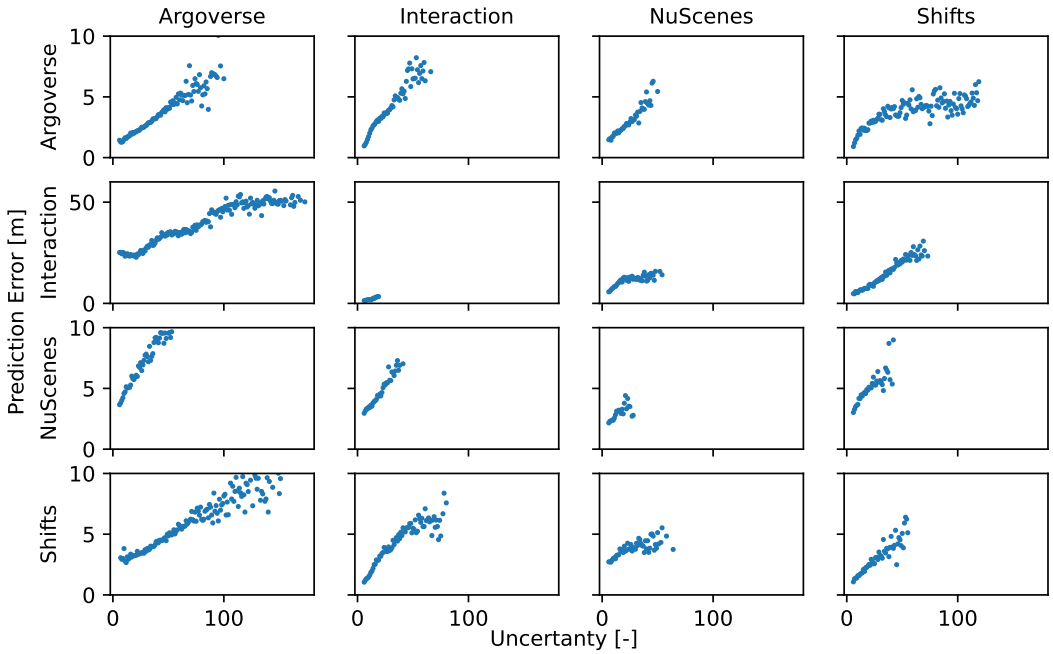


Figure 6.10: Analysis of correlation between uncertainty and prediction error in a cross-dataset setting. Line: dataset onto which the model has been trained. Column: dataset onto which the model has been tested.

Table 6.3: Uncertainty estimation with different expected metrics

	Variance	CNLL	minFDE_5
R-AUC CNLL	2.31	2.38	**2.20**

The results that $minFDE_5$ provides the best proxy for the estimated uncertainty according to the error-retention of cNLL, despite being a different metric, as it actually performs better than the actual cNLL proxy, which we attribute to $minFDE_5$ being less outlier-sensitive for a proxy that has to be computed on every possible pixel, even ones very unlikely and afar from the predicted modalities. Interestingly, while very simple in definition and computation, the variance proxy also does very well, out-performing the CNLL proxy. Since the variance proxy presents the advantage of not requiring a sampled multi-modal prediction for its calculation, and not being biased towards a specific metric, we will use it in the following experiments.

6.2.4.2 Uncertainty enhanced performance

In this section we show the benefit of using the presented uncertainty to adapt the diversity of the predicted locations through the sampling radius r. First, we experimentally show that the optimal sampling radius r_{opt} that minimize the prediction error follows a linear trend with respect the estimated uncertainty. Figure 6.11 depicts the average optimal sampling radius calculated over the Argoverse dataset versus the estimated uncertainty grouped in bins of integer values. Note that this is different from the

previous plots in Fig. 6.10, where the y axis was the measured error, while here in Fig 6.11 we sampled multiple sets of multimodal predictions for each case, using sampling radii in a grid from 0.6 to 2.4 with 0.1 increments. Then for each uncertainty bin, we report average of the radius with optimal performance (minFDE_6) for each case. Indeed, since our goal is to show that the uncertainty can be used to adapt the sampling radius for better performance, our aim is to show an actual linear relationship between this radius and the uncertainty, that we could then use during inference.

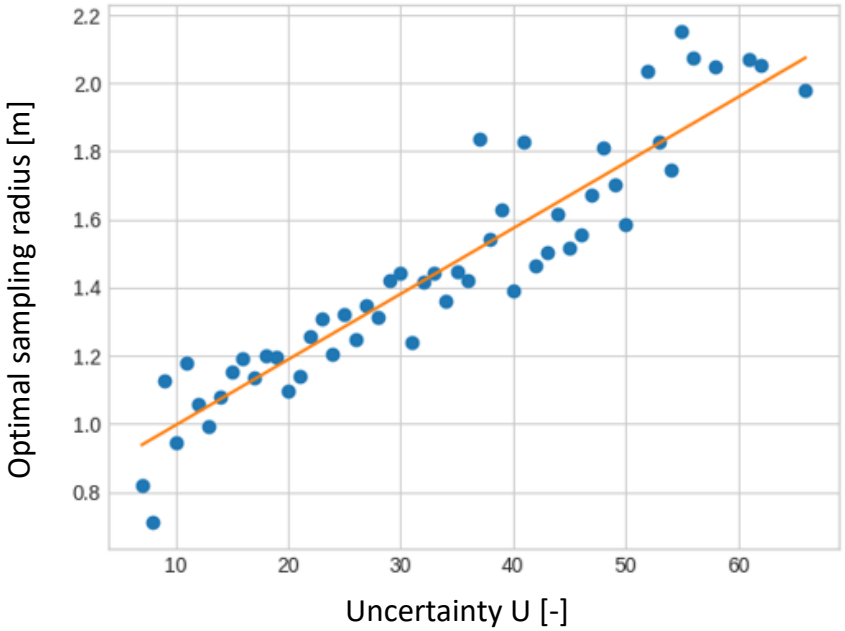


Figure 6.11: Average optimal radius with regard to uncertainty. We bin uncertainty values per equal integer values, and average the optimal radius for each of the cases in the bin. We show in orange the plot resulting from a linear regression

This plot justifies well our motivation to infer an adaptive sampling radius from uncertainty, as it shows an almost linear correlation between uncertainty and the best sampling radius for a fiven prediction case.

Fig. 6.12 highlights that this intuition for adaptive sampling is present across many datasets, for most of the uncertainty values range. We therefore apply ordinary least squares to find regression coefficients between our estimated uncertainty and the optimal radius for a given case. The resulting curves are plotted in Fig. 6.12 and we report the resulting regression coefficients in Tab. 6.4, as well as the optimal fixed radii without adaptation for each dataset.

Table 6.4: Optimal radii and linear regression parameters per dataset

Dataset	Argoverse	Interaction	NuScenes	Shifts
Radius	1.5	0.6	1.1	1.5
Affine	0.020x+0.78	0.026x+0.96	0.014+1.32	0.022x+0.91

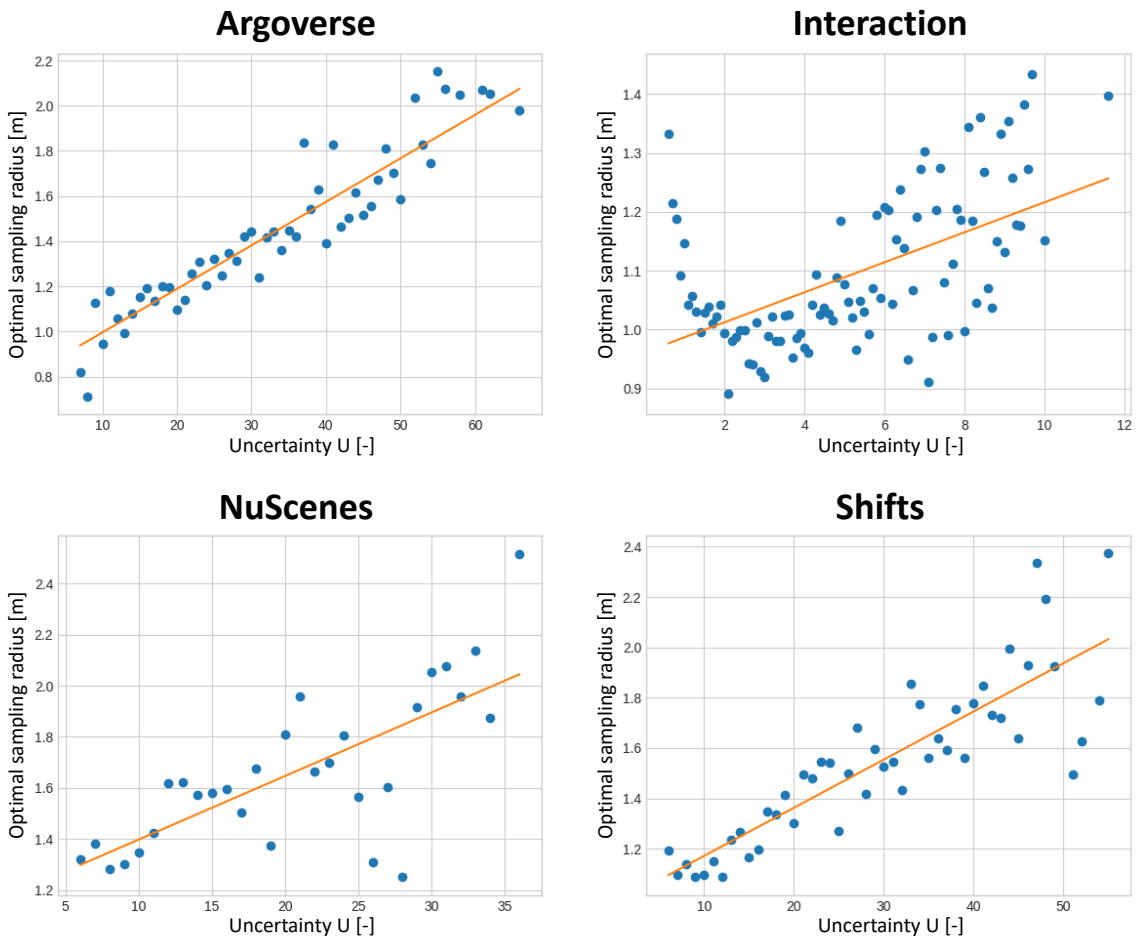


Figure 6.12: Average optimal radius with regard to uncertainty. We bin uncertainty values per equal integer values, and average the optimal radius for each of the cases in the bin. We plot in orange the linear curve obtained from applying least square regression on the points.

Results of performance improvements

Furthermore, we report cross-datasets results using an adaptive sampling radius to adjust the prediction diversity depending on uncertainty. Left image in Fig. 6.13 shows the minFDE_6 cross-dataset performance when the model is trained on the dataset denoted in the row label and evaluated on the datasets denoted by the column label. In each one of this experiment the radius is adapted following the linear model calibrated on the dataset used for training and kept unchanged for evaluation on target datasets. We can see in the middle image of 6.13 how the adaptive sampling strategy is significantly better in almost all cases compared to the constant radius sampling presented in Fig. 6.1.

We benchmark our method of computing the prediction uncertainty by comparing it to a learned variance V of a Gaussian distribution as in [Kendall and Cipolla, 2017, Meyer and Thakurdesai, 2020, Ngiam et al., 2021, Moreau et al., 2022]. As in [Kendall and Cipolla, 2017] we directly predict $s = \log(V)$ for numerical stability with the following loss:

$$L(s) = E \cdot exp(-s) + s \quad \text{with} \quad E = minFDE_6 \tag{6.7}$$

We report on the right image of Fig. 6.13, the improvement in minFDE_6 when using our uncertainty

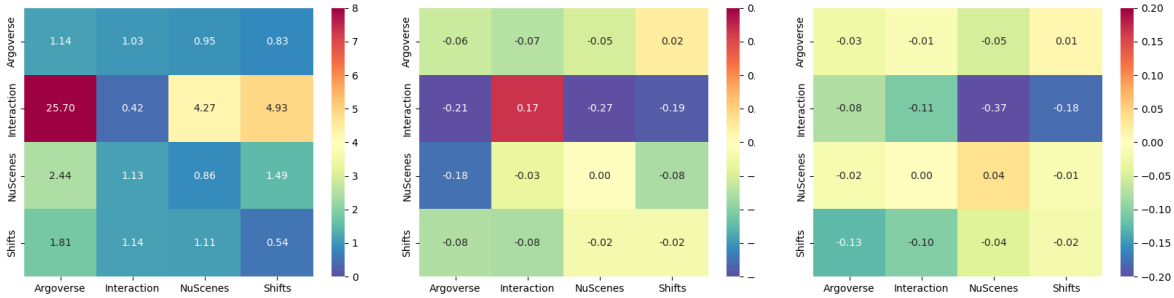


Figure 6.13: **On the Left:** Absolute cross-dataset performance when using a sampling strategy based on uncertainty **On the Middle:** Relative improvement in cross-dataset minFDE_6 when using uncertainty compared to fixed radius sampling **On the right:** Relative improvement in cross-dataset minFDE_6 when using our heatmap-based uncertainty compared to a learned uncertainty baseline

definition of Equation 6.6 compared with the learned baseline of Equation 6.7 to adapt the sampling radius. While it yields similar results on the same train-test diagonal, the learned uncertainty tends to overfit on its training data and doesn't perform as well on out-of-distribution data.

Qualitative analysis We display qualitative prediction examples in Fig. A.2. Each line is an example sample of one dataset, and each column the prediction result of the model trained on the corresponding dataset. We also report uncertainty numbers for each sample to observe how uncertainty matches the heatmap spread and the resulting adapted endpoint sampling.

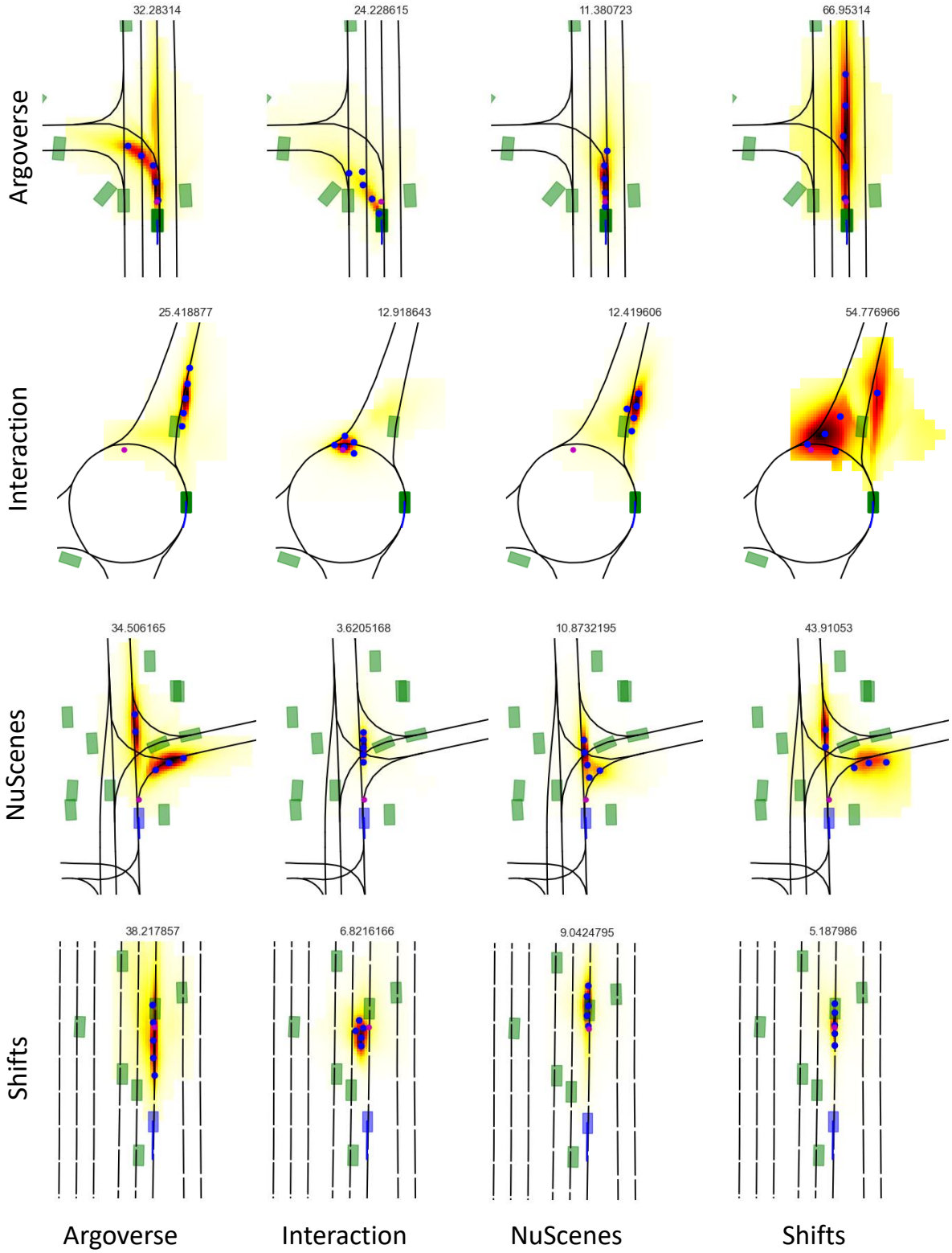


Figure 6.14: Qualitative results across datasets. Sampled endpoints are displayed in blue and ground truth in magenta. Heatmap variance is displayed on top of each example

6.3 Heatmap Calibration

Other than post-sampling metrics and uncertainty estimation, another path to evaluate the quality of a predicted heatmap can be to check its calibration. Our heatmap prediction can be interpreted as a classification task, where each pixel is a possible class. We can then observe our model calibration on this classification task. We follow the same approach as in [Naeini et al., 2015, Guo et al., 2017, Heo et al., 2018, Ovadia et al., 2019, Gesnouin et al., 2022] by plotting reliability diagrams. However one difference compared to these usually evaluated classification tasks is their focus on a relatively limited number of classes (they usually apply to binary classification, ie 2 classes [Naeini et al., 2015, Heo et al., 2018, Gesnouin et al., 2022], up to rarely 100 or 1000 [Guo et al., 2017, Ovadia et al., 2019]) compared to our $256 \times 256 = 65536$ classes, which remains a few orders of magnitude above, albeit with very similar classes, most of which being easy to eliminate (outside of drivable area locations, opposite lanes, etc...).

6.3.1 Reliability diagram

Reliability diagrams compare model confidence against measured accuracy. In our case confidence is the given probability p_i of a pixel i, and accuracy defined by the fraction of cases where the ground truth is actually inside the pixel. As a reminder, a pixel is defined as a $0.5 \times 0.5 m^2$ cell in the xy plane. We divide possible confidence values into bins B_m and compute the average accuracy $\text{acc}(B_m)$ within the bin:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i) \tag{6.8}$$

We then plot the histogram of accuracy values $\text{acc}(B_m)$ for each bin. Ideally, the diagram should be close to the identity line where the average accuracy is equal to the average confidence within each bin. We first display such diagram in Fig. 6.15 the reliability diagram for a THOMAS heatmap with sigmoid activation, as is during training.

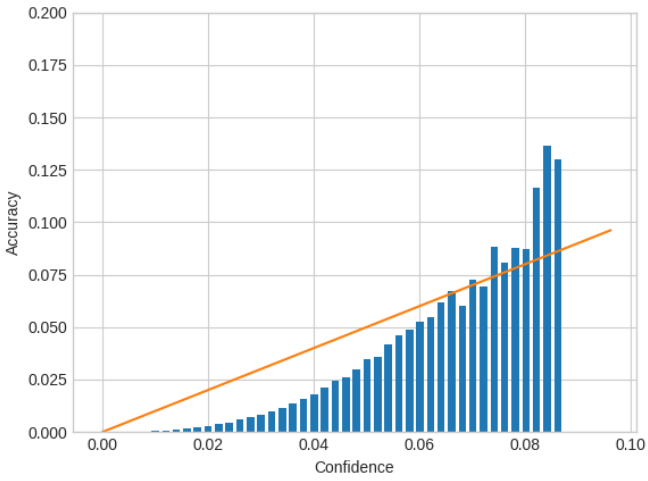


Figure 6.15: Reliability diagram of raw predicted heatmap with simple sigmoid activation

In this diagram and the following ones, we adapt the bin size to have a sufficient amount of sample per

bin so that the average computed accuracy is significant enough. Notably, this model is trained without normalization on the outputed heatmap, ie the sum of probabilities is not necessarily equal to 1, because of the huge imbalance of label distribution (only one positive pixel and all the other are negatives). Additionally, since this model is trained with hard negative mining by selecting and back-propagating on only the top-k predicted pixels (only the top k = 8192 predicted locations are estimated and therefore back-propagated), we can observe on the diagram that the model is over-confident on the low probability values, as the measured accuracy is lower than the predicted confidence. This can be explained by the fact that the model is only trained on a fraction of the negative samples, which are the most likely to be positive, and therefore doesn't have the additional regularization of multiple orders of magnitude more negative pixels in the loss. However, for high confidence pixels, we observe that the measured accuracy is actually higher than the predicted confidence, which means our model does a good job of increasing its confidence on the most likely positions.

6.3.2 Output normalization

Since the final model output should match a probability distribution that sums up to 1, we normalize the output after training by switching the activation from a sigmoid to a softmax on the final layer:

$$\begin{aligned} \text{sigmoid}(x_p) &= \frac{\exp(x_p)}{1+\exp(x_p)} \\ \text{softmax}(x) &= \frac{\exp(x_p)}{\sum_{p'} \exp(x_{p'})} \end{aligned} \tag{6.9}$$

The observed distribution then changes when we actually normalize the predicted heatmap distribution, and gives a different reliability diagram displayed in Fig. 6.16.

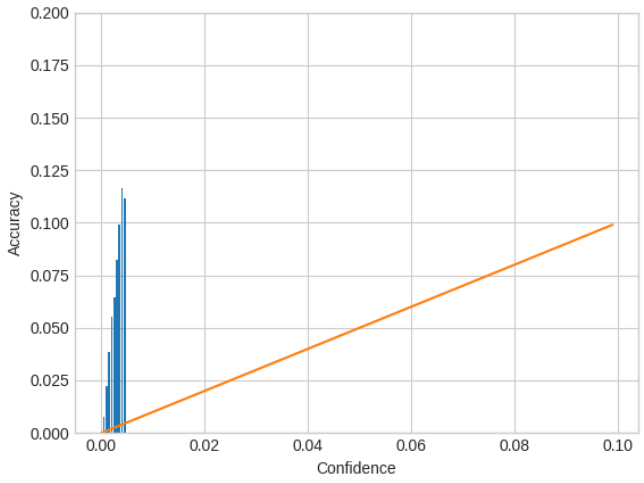


Figure 6.16: Reliability diagram of the model heatmap after softmax normalization

We notice that our model calibration is now shifted above the identity line, which means that predictions have accuracies usually above their predicted probability value. However we notice again a good

global trend where the actual accuracy increases very linearly with the predicted probability.

6.3.3 Heatmap recalibration

We can confirm this correlated trend by applying re-calibration, as explored in [Guo et al., 2017], through a simple temperature re-scaling method: we add a temperature coefficient T inside the softmax to control the contrast between high and low predicted confidence values. A temperature below 1 would make the model more confident, which is our aim in this case, by spreading high confidence values further apart from the low confidence cases.

$$\text{softmax}(x) = \frac{\exp(x_p/T)}{\sum_{p'} \exp(x_{p'}/T)} \tag{6.10}$$

We find a temperature of $T = 0.01$ to yield the best reliability on our case, as illustrated in Fig. 6.17. We observe that the model remains slightly under-confident in low probability values, but still correlates quite well with its effective accuracy.

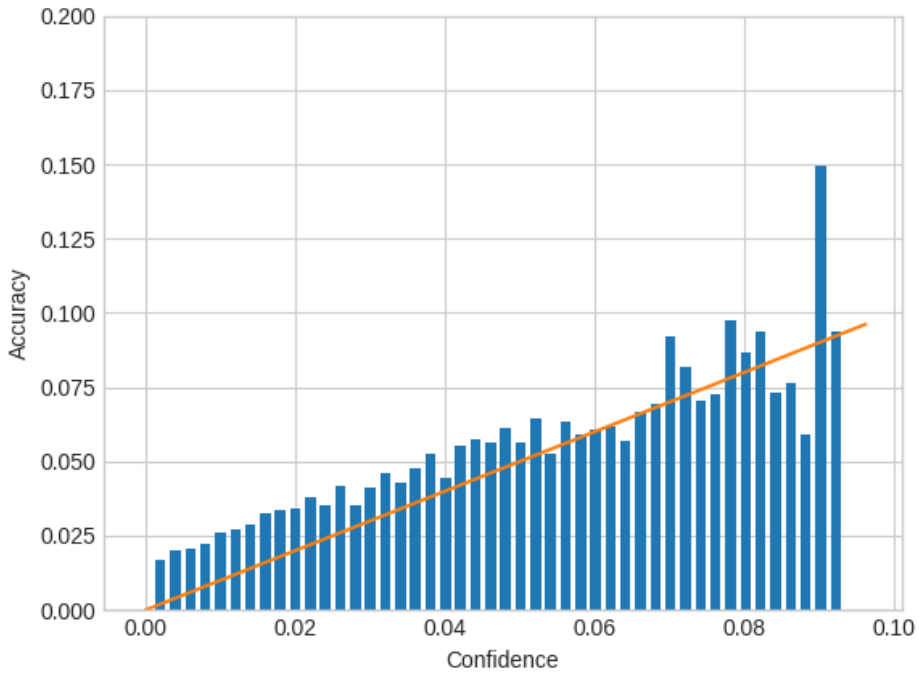


Figure 6.17: Reliability diagram after temperature recalibration on the softmax activation

We can then conclude that, while the initial heatmap output of our model is not perfectly calibrated, overconfident on low probability values and under-confident on high probability values due to its hard-mining training and un-normalized output, it can easily be re-normalized to a pretty well calibrated representation by simply adapting its activation for inference through a softmax with a suite temperature.

6.4 Conclusion and Limitations

In this work we have tried to expand the traditional benchmarks of trajectory prediction by assessing the first cross-dataset analysis in the field of vehicle trajectory prediction for autonomous driving. We have analyzed the cross-dataset transferability performance of two state-of-the-art trajectory prediction models. We have also proposed a new way to estimate uncertainty for heatmap-based trajectory prediction methods that doesn't require any further training and works better than classically learned uncertainties. We showed how uncertainty can be used to boost the trajectory prediction performance of heatmap-based methods in a cross-dataset setting. Finally, we have checked the quality of our predicted heatmaps by verifying their calibration and adjusting in a simple and efficient way.

Limitations This analysis has been limited to car trajectory prediction, a similar analysis across different type of traffic participants such as bicycles and pedestrians would also be of interest. Furthermore, while this study demonstrates that the use of heatmap variance for uncertainty estimation and sampling radius adaptation brings a significant performance improvement to heatmap output methods, the comparison to scalar outputs methods like SceneTransformer shows a less clear trend. The heatmap based method enriched with uncertainty has similar transferability performance to SceneTransformer. SceneTransformer is trained end-to-end to directly predict a set of multimodal coordinates and somehow internally learns to adapt the diversity of the predictions without explicitly outputting an uncertainty value. On the other hand we strongly believe that having an explicit uncertainty output can be useful also for other downstream tasks in the autonomous driving stack.

Chapter 7

Conclusion

Contents

7.1 Summary of contributions and limitations

Overall, this thesis focused on three main research directions:

- **Design and Comparison of Fast Heatmap-based trajectory prediction models** Focusing on the multimodal aspect of the forecast, we proposed a heatmap representation quite different from most existing scalar output methods. Starting from the most simple convolution-based architecture HOME, we observed unachieved coverage performance compared to the best existing methods, but noticed inference speed limitations. We therefore move towards a fully vectorized graph-based architecture GOHOME that retained the same performance gains but attained way faster training and prediction times. We then simplified this fully-vectorized heatmap generation by leveraging cross-attention in THOMAS, which enabled us to incorporate a hierarchical efficient heatmap decoder unlocking further speed gains.

- **Scene-level multi-agent consistent prediction** Starting from the observation that it is much easier to train an independent trajectory predictor than a joint consistent one for all agents, and from the limitation that a heatmap is inherently agent-specific, we designed a second-step module in THOMAS, able to take an independent multimodal prediction as input and to recombine it into a coherent scene prediction. Such a module can be trained on the output of any marginally trained prediction model to provide a joint forecast.

- **Further exploring trajectory prediction evaluation** After reaching top spots on multiple motion forecasting benchmarks, one can wonder if incremental improvements of a few centimeters matter anymore. Instead, we shift towards the actual application of a prediction module. First we study how stable the prediction performance is in places and situations it has never seen before, by training it on one dataset and evaluating it on another. We observe that data augmentation is sometimes necessary to reach satisfactory cross-dataset performance, but that the main factors remain data quality and diversity, and often cannot be fixed by simple data augmentation techniques. Interestingly, we observe that dataset size does not necessarily correlate with model transferable performance, but that the most important factor is the difficulty of the dataset. The harder the training dataset (lower validation performance when trained and tested on the same dataset), the more generalizable to other testing datasets. We also investigate the growing importance of providing uncertainty estimates along with our prediction, and propose a way of estimating this uncertainty directly from heatmap. While most uncertainty works are usually limited to observing the uncertainty ability to correlate with error, we incorporate it into our multimodal sampling to improve final performance.

We also conducted real world experiments by implementing our HOME algorithm on a prototype to train it on home-collected data and run real-time on a few kilometers of open-roads in Shanghai (the prediction was however not used to plan and control the car, it was mostly for visualization, evaluation and demonstration purposes).

7.2 Perspectives and Future Work

While an essential part of the autonomous pipeline, the prediction module cannot be simply tackled as a single independent entity. It is inherently reliant on the inputs that are provided to it, and its design must follow the constraints brought by the planning algorithm that will use its prediction.

Strengthening the connection between perception and prediction As human drivers, we use multiple cues that do not exist in most commonly used trajectory datasets. A striking example are turn signals, which are our prime information to know if a car is going to turn, but are not provided in many datasets. Some works [Casas et al., 2020, Cui et al., 2021] try to bypass the limitations of having to list every required feature in a tabular way by directly using perception features (eg. a direct back-propagated layer input to the perception network). At the same time, interpretability should exist as to what information is used by prediction in order to not overfit or generate unwanted biases. This perception data can also be flawed, and could therefore be provided with caveats to the prediction model, such as multiple hypothesis [Weng et al., 2021] or uncertainty estimates [Ivanovic et al., 2022c]

Leveraging more self or unsupervised training methods for improved performance and/or physics feasibility The task we usually perform on trajectories data is to predict the future using the past. However the trajectory data is way richer than this simple application. Why wouldn't neural networks gain as much understanding of physics or driving behaviors by reconstructing the past using the future, or re-interpolating large sections of trajectories only knowing their first and last point extremities ? Modern transformer architectures [Vaswani et al., 2017, Dosovitskiy et al., 2021] can indifferently reconstruct any piece of their inputs by knowing their position in the sequence/image, and masked auto-encoders have demonstrated great pre-training improvements on both NLP and image tasks [Radford et al., 2018, He et al., 2022]. These masked auto-encoders pretraining could be adapted for trajectory prediction, by adding further motion specific constraints, such as constraining the prediction to be kinematically feasible/differentiable with an added loss to denoise the raw trajectory data, or using the existing road map to generate artificial trajectories [Azevedo et al., 2022].

Not limiting the prediction to already observed entities Predicting where the cars we see might go is one thing, but as drivers we also identify dangerous and unseen intersections, and try to anticipate if an unseen car could pop up from some blind spot or transversal street. A good example of this behavior is when we slow down for intersections even if we don't see immediate other cars on it. Translated into a prediction task, we could train a model to predict the spawning of new road agents from occluded areas, by identifying when tracked vehicles are first seen (first data point on their trajectory). Since we cannot provide agent-specific representations for this task, a probabilistic heatmap would be very well suited to predict the joint spawning probability of a new agent on each xy pixel in the top-view plan.

Evaluating prediction through planning A plethora of metrics are available on each trajectory benchmark: minFDE_1, minFDE_6, MissRate_1, MissRate_6, ... However, we do not really know which of these metrics, if any, is the most relevant for the use that the planning step will make of it. An extensive evaluation of a prediction system could be done by choosing or designing a planning method, and evaluating the final driving performance, as in collision rate, comfort, miles covered without intervention or incident, for a given prediction model. [Ivanovic and Pavone, 2021a, McAllister et al., 2022] already take into account planning to either propose new metrics or identify important agents for planning, but they remain quite general and don't investigate every aspect of prediction performance.This would also

enable further analysis to know the actual importance of multimodality, how many modes are really needed, which metric is the most relevant (precision/minFDE or coverage/MissRate), and if consistent scene-level predictions are necessary.

Chapter 8

Conclusion

Contents

8.1 Summary of contributions and limitations

Design and Comparison of Fast Heatmap-based trajectory prediction models Focusing on the multimodal aspect of the forecast, we proposed a heatmap representation quite different from most existing scalar output methods. Starting from the most simple convolution-based architecture HOME, we observed unachieved coverage performance compared to the best existing methods, but noticed inference speed limitations. We therefore move towards a fully vectorized graph-based architecture GOHOME that retained the same performance gains but attained way faster training and prediction times. We then simplified this fully-vectorized heatmap generation by leveraging cross-attention in THOMAS, which enabled us to incorporate a hierarchical efficient heatmap decoder unlocking further speed gains.

Scene-level multi-agent consistent prediction Starting from the obseration that it is much easier to train an independent trajectory predictor than a joint consistent one for all agents, and from the limitation that a heatmap is inherently agent-specific, we designed a second-step module in THOMAS, able to take any independent multimodal prediction as input and to recombine it into a coherent scene prediction.

8.2 Perspectives and Future Work

Appendix A

Appendix

A Detailed architecture of GOHOME

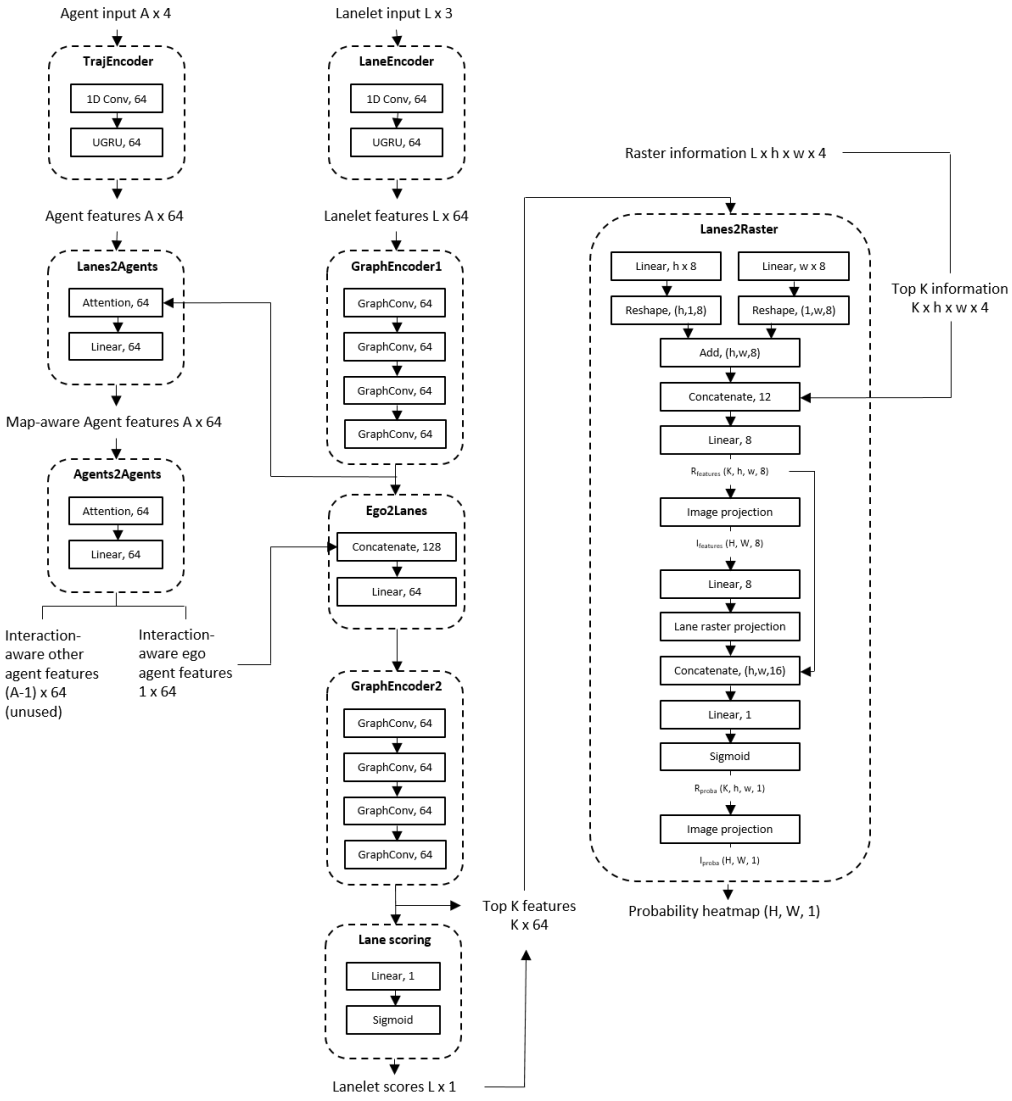


Figure A.1: Detailed illustration of the GOHOME model architecture

B Additional qualitative results for GOHOME

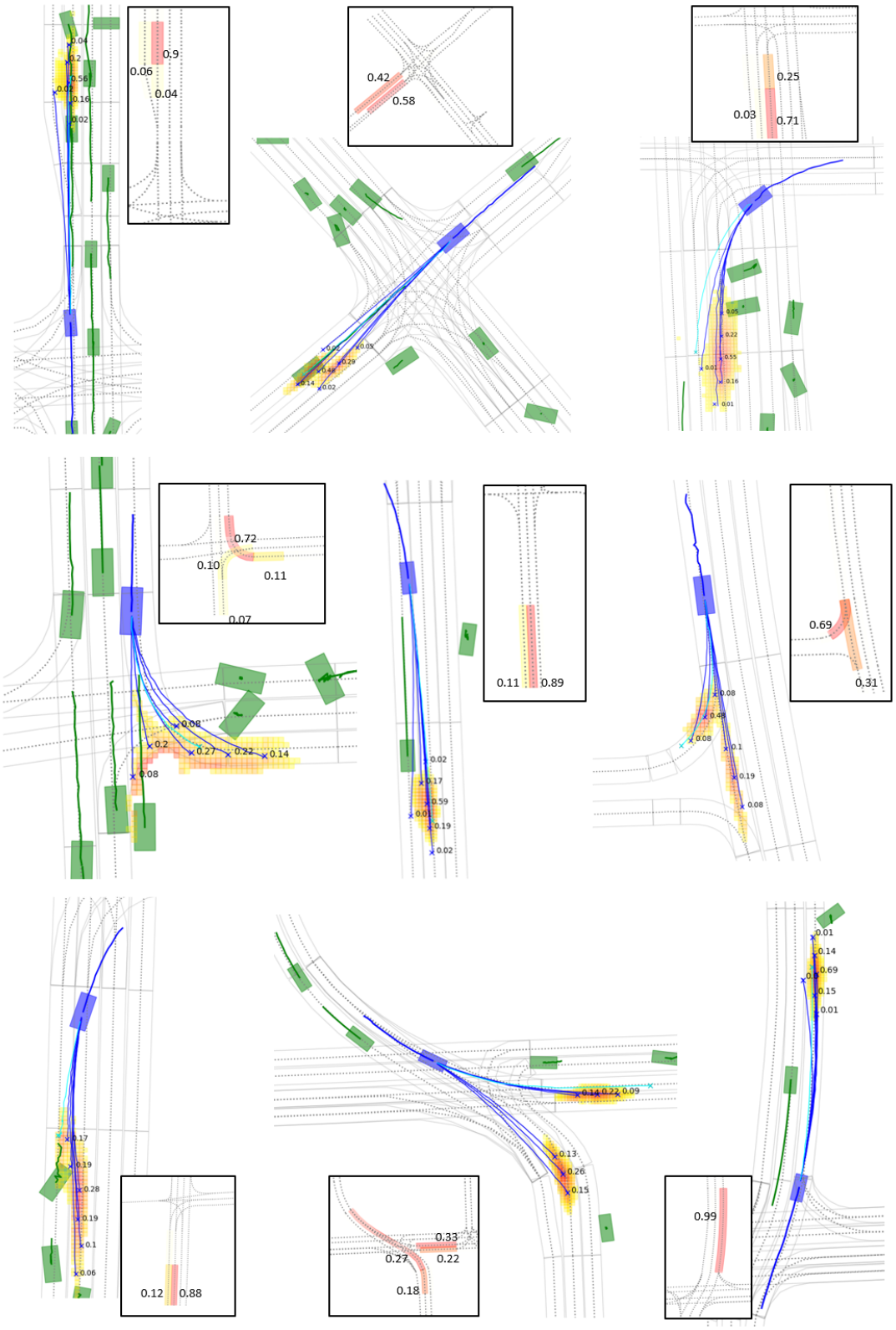


Figure A.2: Additional qualitative cases, with both heatmap and lane classification (in the small frame)

C Detailed architecture of THOMAS

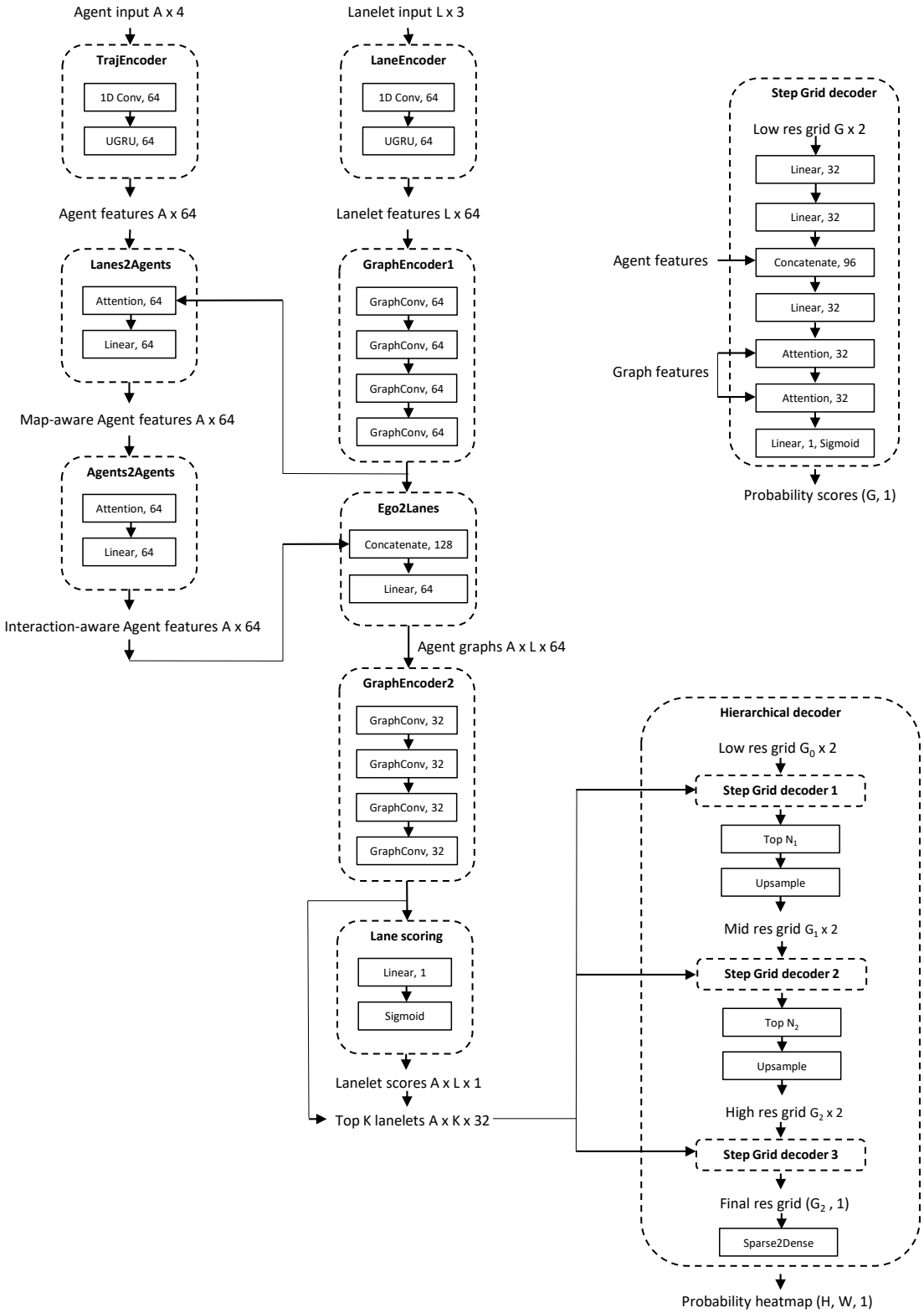


Figure A.3: Detailed illustration of the THOMAS heatmap generator model

Appendix B

Résumé en français

Introduction Cette partie présente le contexte de la prédiction de trajectoire pour voitures autonomes, ses problématiques et les principales solutions proposées dans cette thèse.

Méthodes de prédiction de trajectoire non basées sur l'apprentissage machine Dans ce chapitre, nous présentons les méthodes traditionnellement utilisées pour la prédiction de trajectoire en excluant l'apprentissage machine. Nous présentons les modèles cinématiques utilisant la physique du mouvement et le temps, les modèles stochastiques modélisant les possibles phénomènes aléatoires dans les trajectoires futures, les manières de représenter les interactions entre voitures et des méthodes hiérarchiques pour décomposer les trajectoires en plusieurs couches décisionnelles.

Méthodes d'apprentissage pour la prédiction de trajectoire Dans ce chapitre, nous analysons les méthodes existantes d'apprentissage pour les différents aspects de la prédiction de trajectoire. Nous détaillons les différentes architectures pour le traitement de signal temporal et la modélisation des interactions entre agents. Ensuite, nous étudions les manières d'intégrer le contexte routier autour du véhicule autonome, en ajoutant la carte sous la forme d'une image, d'un graph ou d'un ensemble de points ou de lignes. Enfin, nous observons les différentes manières de représenter une future trajectoire en sortie d'un réseau de neurones, et notons leurs limitations actuelles.

Prédiction de trajectoire par des cartes de probabilité Dans ce chapitre, nous présentons notre première principale contribution, qui consiste à représenter une future trajectoire non pas par une séquence de coordonnées, mais par une carte de probabilité pour un point futur à un horizon de temps donné. Nous expliquons ensuite comment une telle carte de probabilité peut être décodée en une trajectoire. Nous détaillons les différentes manières de générer ce genre de carte de probabilité, que ce soit par des réseaux de convolutions (HOME) ou par des graphes de segments de route (GOHOME). Enfin, nous expliquons les avantages de cette nouvelle représentation, notamment vis à vis de leur capacité à représenter plusieurs futures en même temps.

Prédiction cohérente de plusieurs agents Dans ce chapitre, nous commençons par expliquer les challenges de la prédiction de trajectoire de plusieurs agents à la fois, notamment pour leurs interactions.

Nous proposons une méthode adaptée aux cartes de probabilité présentées dans le chapitre précédent, pour transformer des prédictions marginales indépendantes pour chaque agent en une seule prédiction jointe et cohérente. Nous détaillons les expériences conduites pour prouver les performances de cette approche.

Repenser l'évaluation des modèles de prédiction de trajectoire Dans ce chapitre, nous posons le problème de la capacité des modèles de trajectoire appris à généraliser sur de nouvelles donnes. Nous menons une analyse systématique sur quatre datasets et comparons les capacités et limitations de différents modèles. Nous proposons ensuite une nouvelle manière d'estimer l'incertitude des cartes de probabilités et de l'utiliser pour prédire les données hors distribution et améliorer les performances du modèle.

Conclusion

Nous résumons cette thèse et identifions les orientations futures potentielles de notre recherche.

Appendix C

Bibliography

[lea,] Argoverse motion forecasting competition. `https://eval.ai/web/challenges/challenge-page/454/leaderboard/1279`. Accessed: 2021-06-15.

[Alahi et al., 2016] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971.

[Altché and de La Fortelle, 2017] Altché, F. and de La Fortelle, A. (2017). An lstm network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 353–359. IEEE.

[Amirloo et al., 2022] Amirloo, E., Rasouli, A., Lakner, P., Rohani, M., and Luo, J. (2022). Latentformer: Multi-agent transformer-based interaction modeling and trajectory prediction. *arXiv preprint arXiv:2203.01880*.

[Aoude et al., 2011] Aoude, G. S., Desaraju, V. R., Stephens, L. H., and How, J. P. (2011). Behavior classification algorithms at intersections and validation using naturalistic data. In *2011 ieee intelligent vehicles symposium (iv)*, pages 601–606. IEEE.

[Atev et al., 2010] Atev, S., Miller, G., and Papanikolopoulos, N. P. (2010). Clustering of vehicle trajectories. *IEEE transactions on intelligent transportation systems*, 11(3):647–657.

[Azevedo et al., 2022] Azevedo, C., Gilles, T., Sabatini, S., and Tsishkou, D. (2022). Exploiting map information for self-supervised learning in motion forecasting. *arXiv preprint arXiv:2210.04672*.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[Badrinarayanan et al., 2017] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

[Bahari and Alahi, 2019] Bahari, M. and Alahi, A. (2019). Feed-forwards meet recurrent networks in vehicle trajectory prediction. In *Swiss Transport Research Conference (STRC)*, number CONF.

[Bahari et al., 2022] Bahari, M., Saadatnejad, S., Rahimi, A., Shaverdikondori, M., Shahidzadeh, A. H., Moosavi-Dezfooli, S.-M., and Alahi, A. (2022). Vehicle trajectory prediction works, but not everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17123–17133.

[Bahram et al., 2015] Bahram, M., Lawitzky, A., Friedrichs, J., Aeberhard, M., and Wollherr, D. (2015). A game-theoretic approach to replanning-aware interactive scene prediction and planning. *IEEE Transactions on Vehicular Technology*, 65(6):3981–3992.

[Bansal et al., 2018] Bansal, M., Krizhevsky, A., and Ogale, A. (2018). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*.

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

[Berndt and Dietmayer, 2009] Berndt, H. and Dietmayer, K. (2009). Driver intention inference with vehicle onboard sensors. In *2009 IEEE international conference on vehicular electronics and safety (ICVES)*, pages 102–107. IEEE.

[Biktairov et al., 2020] Biktairov, Y., Stebelev, M., Rudenko, I., Shliazhko, O., and Yangel, B. (2020). Prank: motion prediction based on ranking. *Advances in Neural Information Processing Systems*, 33:2553–2563.

[Broadhurst et al., 2005] Broadhurst, A., Baker, S., and Kanade, T. (2005). Monte carlo road safety reasoning. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 319–324. IEEE.

[Brubaker et al., 2013] Brubaker, M. A., Geiger, A., and Urtasun, R. (2013). Lost! leveraging the crowd for probabilistic visual self-localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3057–3064.

[Buhet et al., 2021] Buhet, T., Wirbel, E., Bursuc, A., and Perrotton, X. (2021). Plop: probabilistic polynomial objects trajectory prediction for autonomous driving. In *Conference on Robot Learning*, pages 329–338. PMLR.

[Caesar et al., 2020] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *CVPR*.

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

[Casas et al., 2020] Casas, S., Gulino, C., Suo, S., Luo, K., Liao, R., and Urtasun, R. (2020). Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision*, pages 624–641. Springer.

[Casas et al., 2018] Casas, S., Luo, W., and Urtasun, R. (2018). Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956. PMLR.

[Casas et al., 2021] Casas, S., Sadat, A., and Urtasun, R. (2021). Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412.

[Chai et al., 2019] Chai, Y., Sapp, B., Bansal, M., and Anguelov, D. (2019). Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*.

[Chang et al., 2019] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757.

[Chekroun et al., 2021] Chekroun, R., Toromanoff, M., Hornauer, S., and Moutarde, F. (2021). Gri: General reinforced imitation and its application to vision-based autonomous driving. *arXiv preprint arXiv:2111.08575*.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[Christopher, 2009] Christopher, T. C. T. (2009). *Analysis of dynamic scenes: Application to driving assistance*. PhD thesis, Institut National Polytechnique de Grenoble-INPG.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Codevilla et al., 2019] Codevilla, F., Santana, E., López, A. M., and Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338.

[Colyar and Halkias, 2006] Colyar, J. and Halkias, J. (2006). Us highway 80 dataset," federal highway administration (fhwa), vol. *Tech, no. Rep*.

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

[Cui et al., 2021] Cui, A., Casas, S., Sadat, A., Liao, R., and Urtasun, R. (2021). Lookout: Diverse multi-future prediction and planning for self-driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16107–16116.

[Cui et al., 2019] Cui, H., Radosavljevic, V., Chou, F.-C., Lin, T.-H., Nguyen, T., Huang, T.-K., Schneider, J., and Djuric, N. (2019). Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

[Danielsson et al., 2007] Danielsson, S., Petersson, L., and Eidehall, A. (2007). Monte carlo based threat assessment: Analysis and improvements. In *2007 IEEE Intelligent Vehicles Symposium*, pages 233–238. IEEE.

[Deo and Trivedi, 2018a] Deo, N. and Trivedi, M. M. (2018a). Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476.

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

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

[Deo et al., 2022] Deo, N., Wolff, E., and Beijbom, O. (2022). Multimodal trajectory prediction conditioned on lane-graph traversals. In *Conference on Robot Learning*, pages 203–212. PMLR.

[Deschaud and Goulette, 2010] Deschaud, J.-E. and Goulette, F. (2010). A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing. In *3DPVT*. Hal Archives-Ouvertes Paris, France.

[Ding, 2020] Ding, Y. (2020). Simple understanding of kinematic bicycle model. `https://dingyan89.medium.com/simple-understanding-of-kinematic-bicycle-model-81cac6420357`medium.com [Online; posted 15-February-2020].

[Djuric et al., 2018] Djuric, N., Radosavljevic, V., Cui, H., Nguyen, T., Chou, F.-C., Lin, T.-H., and Schneider, J. (2018). Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks. *arXiv preprint arXiv:1808.05819*, 1(2):6.

[Dolgov et al., 2008] Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. *Ann Arbor*, 1001(48105):18–80.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

[Eidehall and Petersson, 2006] Eidehall, A. and Petersson, L. (2006). Threat assessment for general road scenes using monte carlo sampling. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1173–1178. IEEE.

[Ettinger et al., 2021] Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C. R., Zhou, Y., et al. (2021). Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719.

[Firl et al., 2012] Firl, J., Stübing, H., Huss, S. A., and Stiller, C. (2012). Predictive maneuver evaluation for enhancement of car-to-x mobility data. In *2012 IEEE Intelligent Vehicles Symposium*, pages 558–564. IEEE.

[Gao et al., 2020] Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., and Schmid, C. (2020). Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533.

[Gesnouin, 2022] Gesnouin, J. (2022). *Analysis of pedestrian movements and gestures using an on-board camera to predict their intentions*. PhD thesis, Université Paris sciences et lettres.

[Gesnouin et al., 2022] Gesnouin, J., Pechberti, S., Stanciulescu, B., and Moutarde, F. (2022). Assessing cross-dataset generalization of pedestrian crossing predictors. *arXiv preprint arXiv:2201.12626*.

[Gilles et al., 2021a] Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., and Moutarde, F. (2021a). Gohome: Graph-oriented heatmap output forfuture motion estimation. *arXiv preprint arXiv:2108.09640*.

[Gilles et al., 2021b] Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., and Moutarde, F. (2021b). Home: Heatmap output for future motion estimation. In *ITSC*.

[Gilles et al., 2022] Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., and Moutarde, F. (2022). Thomas: Trajectory heatmap output with learned multi-agent sampling. In *ICLR*.

[Girgis et al., 2021] Girgis, R., Golemo, F., Codevilla, F., Weiss, M., D'Souza, J. A., Kahou, S. E., Heide, F., and Pal, C. (2021). Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations*.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Giuliari et al., 2021] Giuliari, F., Hasan, I., Cristani, M., and Galasso, F. (2021). Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*, pages 10335–10342. IEEE.

[Godard et al., 2017] Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279.

[Gomes and Wolf, 2022] Gomes, I. and Wolf, D. (2022). A review on intention-aware and interaction-aware trajectory prediction for autonomous vehicles.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

[Greene et al., 2011] Greene, D., Liu, J., Reich, J., Hirokawa, Y., Shinagawa, A., Ito, H., and Mikami, T. (2011). An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists. *IEEE Transactions on intelligent transportation systems*, 12(4):942–953.

[Gu et al., 2021] Gu, J., Sun, C., and Zhao, H. (2021). Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312.

[Gundersen, 2020] Gundersen, G. (2020). Inference for hidden markov models. `https://dingyan89.medium.com/simple-understanding-of-kinematic-bicycle-model-81cac6420357`.

[Guo et al., 2017] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

[Gupta et al., 2018] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264.

[Hasan et al., 2022] Hasan, I., Liao, S., Li, J., Akram, S. U., and Shao, L. (2022). Pedestrian detection: Domain generalization, cnns, transformers and beyond. *arXiv preprint arXiv:2201.03176*.

[He et al., 2022] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Helbing and Molnar, 1995] Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282.

[Heo et al., 2018] Heo, J., Lee, H. B., Kim, S., Lee, J., Kim, K. J., Yang, E., and Hwang, S. J. (2018). Uncertainty-aware attention for reliable interpretation and prediction. *Advances in neural information processing systems*, 31.

[Hermes et al., 2009] Hermes, C., Wohler, C., Schenk, K., and Kummert, F. (2009). Long-term vehicle motion prediction. In *2009 IEEE intelligent vehicles symposium*, pages 652–657. IEEE.

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

[Hong et al., 2019] Hong, J., Sapp, B., and Philbin, J. (2019). Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462.

[Horváth et al., 2022] Horváth, D., Erdős, G., Istenes, Z., Horváth, T., and Földi, S. (2022). Object detection using sim2real domain randomization for robotic applications. *IEEE Transactions on Robotics*.

[Houenou et al., 2013] Houenou, A., Bonnifait, P., Cherfaoui, V., and Yao, W. (2013). Vehicle trajectory prediction based on motion model and maneuver recognition. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 4363–4369. IEEE.

[Houston et al., 2021] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. (2021). One thousand and one hours: Self-driving motion prediction dataset. In *Conference on Robot Learning*, pages 409–418. PMLR.

[Hu et al., 2006] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., and Maybank, S. (2006). A system for learning statistical motion patterns. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1450–1464.

[Hu et al., 2018] Hu, Y., Zhan, W., and Tomizuka, M. (2018). Probabilistic prediction of vehicle semantic intention and motion. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 307–313. IEEE.

[Huang et al., 2022] Huang, Z., Mo, X., and Lv, C. (2022). Multi-modal motion prediction with transformer-based neural network for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2605–2611. IEEE.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

[Ivanovic et al., 2020] Ivanovic, B., Elhafsi, A., Rosman, G., Gaidon, A., and Pavone, M. (2020). Mats: An interpretable trajectory forecasting representation for planning and control. In *CoRL*.

[Ivanovic et al., 2022a] Ivanovic, B., Lee, K.-H., Tokmakov, P., Wulfe, B., Mcllister, R., Gaidon, A., and Pavone, M. (2022a). Heterogeneous-agent trajectory forecasting incorporating class uncertainty. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12196–12203. IEEE.

[Ivanovic et al., 2022b] Ivanovic, B., Lin, Y., Shrivastava, S., Chakravarty, P., and Pavone, M. (2022b). Propagating state uncertainty through trajectory forecasting. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2351–2358. IEEE.

[Ivanovic et al., 2022c] Ivanovic, B., Lin, Y., Shrivastava, S., Chakravarty, P., and Pavone, M. (2022c). Propagating state uncertainty through trajectory forecasting. In *ICRA*.

[Ivanovic and Pavone, 2019] Ivanovic, B. and Pavone, M. (2019). The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384.

[Ivanovic and Pavone, 2021a] Ivanovic, B. and Pavone, M. (2021a). Injecting planning-awareness into prediction and detection evaluation. *arXiv preprint arXiv:2110.03270*.

[Ivanovic and Pavone, 2021b] Ivanovic, B. and Pavone, M. (2021b). Rethinking trajectory forecasting evaluation. *arXiv preprint arXiv:2107.10297*.

[Joseph et al., 2011] Joseph, J., Doshi-Velez, F., Huang, A. S., and Roy, N. (2011). A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400.

[Jouaber et al., 2021] Jouaber, S., Bonnabel, S., Velasco-Forero, S., and Pilte, M. (2021). Nnakf: A neural network adapted kalman filter for target tracking. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4075–4079. IEEE.

[Käfer et al., 2010] Käfer, E., Hermes, C., Wöhler, C., Ritter, H., and Kummert, F. (2010). Recognition of situation classes at road intersections. In *2010 ieee international conference on robotics and automation*, pages 3960–3965. IEEE.

[Karle et al., 2022] Karle, P., Geisslinger, M., Betz, J., and Lienkamp, M. (2022). Scenario understanding and motion prediction for autonomous vehicles-review and comparison. *IEEE Transactions on Intelligent Transportation Systems*.

[Kendall and Cipolla, 2017] Kendall, A. and Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5974–5983.

[Kendall et al., 2015] Kendall, A., Grimes, M., and Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2938–2946.

[Kendall et al., 2017] Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., and Bry, A. (2017). End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE international conference on computer vision*, pages 66–75.

[Khandelwal et al., 2020] Khandelwal, S., Qi, W., Singh, J., Hartnett, A., and Ramanan, D. (2020). What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*.

[Khosroshahi et al., 2016] Khosroshahi, A., Ohn-Bar, E., and Trivedi, M. M. (2016). Surround vehicles trajectory analysis with recurrent neural networks. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2267–2272. IEEE.

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

[Kim et al., 2021] Kim, B., Park, S. H., Lee, S., Khoshimjonov, E., Kum, D., Kim, J., Kim, J. S., and Choi, J. W. (2021). Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *CVPR*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[Kurbiel et al., 2020] Kurbiel, T., Sachdeva, A., Zhao, K., and Buehren, M. (2020). Prognosenet: A generative probabilistic framework for multimodal position prediction given context information. *arXiv preprint arXiv:2010.00802*.

[Labatut et al., 2007] Labatut, P., Pons, J.-P., and Keriven, R. (2007). Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE.

[Lakshminarayanan et al., 2017] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

[Lang et al., 2019] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705.

[LaValle et al., 1998] LaValle, S. M. et al. (1998). Rapidly-exploring random trees: A new tool for path planning.

[Lawitzky et al., 2013] Lawitzky, A., Althoff, D., Passenberg, C. F., Tanzmeister, G., Wollherr, D., and Buss, M. (2013). Interactive scene prediction for automotive applications. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1028–1033. IEEE.

[LeCun et al., 1995] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[Lee et al., 2017] Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H., and Chandraker, M. (2017). Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 336–345.

[Lefèvre et al., 2011] Lefèvre, S., Laugier, C., and Ibañez-Guzmán, J. (2011). Exploiting map information for driver intention estimation at road intersections. In *2011 ieee intelligent vehicles symposium (iv)*, pages 583–588. IEEE.

[Lefèvre et al., 2014] Lefèvre, S., Vasquez, D., and Laugier, C. (2014). A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14.

[Lenz et al., 2017] Lenz, D., Diehl, F., Le, M. T., and Knoll, A. (2017). Deep neural networks for markovian interactive scene prediction in highway scenarios. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 685–692. IEEE.

[Li et al., 2017] Li, N., Oyler, D. W., Zhang, M., Yildiz, Y., Kolmanovsky, I., and Girard, A. R. (2017). Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Transactions on control systems technology*, 26(5):1782–1797.

[Liang et al., 2020] Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., and Urtasun, R. (2020). Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer.

[Liang et al., 2018] Liang, Z., Feng, Y., Guo, Y., Liu, H., Chen, W., Qiao, L., Zhou, L., and Zhang, J. (2018). Learning for disparity estimation through feature constancy. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2811–2820.

[Liebner et al., 2013] Liebner, M., Klanner, F., Baumann, M., Ruhhammer, C., and Stiller, C. (2013). Velocity-based driver intent inference at urban intersections in the presence of preceding vehicles. *IEEE Intelligent Transportation Systems Magazine*, 5(2):10–21.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[Liu et al., 2021a] Liu, J., Mao, X., Fang, Y., Zhu, D., and Meng, M. Q.-H. (2021a). A survey on deep-learning approaches for vehicle trajectory prediction in autonomous driving. In *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 978–985. IEEE.

[Liu et al., 2018] Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., and Yosinski, J. (2018). An intriguing failing of convolutional neural networks and the coordconv solution. *Advances in neural information processing systems*, 31.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

[Liu et al., 2021b] Liu, Y., Zhang, J., Fang, L., Jiang, Q., and Zhou, B. (2021b). Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Luo et al., 2020] Luo, C., Sun, L., Dabiri, D., and Yuille, A. (2020). Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2370–2376. IEEE.

[Luo et al., 2016] Luo, W., Schwing, A. G., and Urtasun, R. (2016). Efficient deep learning for stereo matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5695–5703.

[Luo et al., 2018] Luo, W., Yang, B., and Urtasun, R. (2018). Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577.

[MacQueen, 1967] MacQueen, J. (1967). Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297.

[Mahjourian et al., 2022] Mahjourian, R., Kim, J., Chai, Y., Tan, M., Sapp, B., and Anguelov, D. (2022). Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*.

[Malinin, 2019] Malinin, A. (2019). *Uncertainty estimation in deep learning with application to spoken language assessment*. PhD thesis, University of Cambridge.

[Malinin et al., 2021] Malinin, A., Band, N., Chesnokov, G., Gal, Y., Gales, M. J., Noskov, A., Ploskonosov, A., Prokhorenkova, L., Provilkov, I., Raina, V., et al. (2021). Shifts: A dataset of real distributional shift across multiple large-scale tasks. *arXiv preprint arXiv:2107.07455*.

[Mangalam et al., 2021] Mangalam, K., An, Y., Girase, H., and Malik, J. (2021). From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15233–15242.

[McAllister et al., 2022] McAllister, R., Wulfe, B., Mercat, J., Ellis, L., Levine, S., and Gaidon, A. (2022). Control-aware prediction objectives for autonomous driving. *arXiv preprint arXiv:2204.13319*.

[Mercat, 2021] Mercat, J. (2021). *Motion forecasting of the objects in road scenes*. PhD thesis, Université Paris-Saclay.

[Mercat et al., 2020] Mercat, J., Gilles, T., El Zoghby, N., Sandou, G., Beauvois, D., and Gil, G. P. (2020). Multi-head attention for multi-modal joint vehicle motion forecasting. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9638–9644. IEEE.

[Mersch et al., 2021] Mersch, B., Höllen, T., Zhao, K., Stachniss, C., and Roscher, R. (2021). Maneuver-based trajectory prediction for self-driving cars using spatio-temporal convolutional networks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4888–4895. IEEE.

[Messaoud et al., 2021] Messaoud, K., Deo, N., Trivedi, M. M., and Nashashibi, F. (2021). Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 165–170. IEEE.

[Messaoud et al., 2019] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. (2019). Non-local social pooling for vehicle trajectory prediction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 975–980. IEEE.

[Messaoud et al., 2020] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. (2020). Attention based vehicle trajectory prediction. *IEEE Transactions on Intelligent Vehicles*, 6(1):175–185.

[Meyer and Thakurdesai, 2020] Meyer, G. P. and Thakurdesai, N. (2020). Learning an uncertainty-aware object detector for autonomous driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10521–10527. IEEE.

[Mo et al., 2020] Mo, X., Xing, Y., and Lv, C. (2020). Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction. *arXiv:2012.05032*.

[Mo et al., 2021] Mo, X., Xing, Y., and Lv, C. (2021). Heterogeneous edge-enhanced graph attention network for multi-agent trajectory prediction. *arXiv:2106.07161*.

[Moreau et al., 2023] Moreau, A., Gilles, T., Piasco, N., Tsishkou, D., Stanciulescu, B., and de La Fortelle, A. (2023). Imposing: Implicit pose encoding for efficient visual localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2892–2902.

[Moreau et al., 2022] Moreau, A., Piasco, N., Tsishkou, D., Stanciulescu, B., and de La Fortelle, A. (2022). Coordinet: uncertainty-aware pose regressor for reliable vehicle localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2229–2238.

[Moutarde et al., 2007] Moutarde, F., Bargeton, A., Herbin, A., and Chanussot, L. (2007). Robust on-vehicle real-time visual detection of american and european speed limit signs, with a modular traffic signs recognition system. In *2007 IEEE intelligent vehicles symposium*, pages 1122–1126. IEEE.

[Naeini et al., 2015] Naeini, M. P., Cooper, G., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

[Narayanan et al., 2021] Narayanan, S., Moslemi, R., Pittaluga, F., Liu, B., and Chandraker, M. (2021). Divide-and-conquer for lane-aware diverse trajectory prediction. In *CVPR*.

[Nayakanti et al., 2022] Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K. S., and Sapp, B. (2022). Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*.

[Ngiam et al., 2021] Ngiam, J., Vasudevan, V., Caine, B., Zhang, Z., Chiang, H.-T. L., Ling, J., Roelofs, R., Bewley, A., Liu, C., Venugopal, A., et al. (2021). Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*.

[Ohn-Bar et al., 2015] Ohn-Bar, E., Tawari, A., Martin, S., and Trivedi, M. M. (2015). On surveillance for safety critical events: In-vehicle video networks for predictive driver assistance systems. *Computer Vision and Image Understanding*, 134:130–140.

[Ovadia et al., 2019] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32.

[Park et al., 2018] Park, S. H., Kim, B., Kang, C. M., Chung, C. C., and Choi, J. W. (2018). Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE.

[Phan-Minh et al., 2020] Phan-Minh, T., Grigore, E. C., Boulton, F. A., Beijbom, O., and Wolff, E. M. (2020). Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083.

[Phillips et al., 2017] Phillips, D. J., Wheeler, T. A., and Kochenderfer, M. J. (2017). Generalizable intention prediction of human drivers at intersections. In *2017 IEEE intelligent vehicles symposium (IV)*, pages 1665–1670. IEEE.

[Poggenhans et al., 2018] Poggenhans, F., Pauls, J.-H., Janosovits, J., Orf, S., Naumann, M., Kuhnt, F., and Mayr, M. (2018). Lanelet2: A high-definition map framework for the future of automated driving. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pages 1672–1679. IEEE.

[Polack et al., 2017] Polack, P., Altché, F., d'Andréa Novel, B., and de La Fortelle, A. (2017). The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE intelligent vehicles symposium (IV)*, pages 812–818. IEEE.

[Postnikov et al., 2021] Postnikov, A., Gamayunov, A., and Ferrer, G. (2021). Transformer based trajectory prediction. *arXiv preprint arXiv:2112.04350*.

[Pustynnikov and Eremeev, 2021] Pustynnikov, A. and Eremeev, D. (2021). Estimating uncertainty for vehicle motion prediction on yandex shifts dataset. *arXiv preprint arXiv:2112.08355*.

[Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

[Ranftl et al., 2021] Ranftl, R., Bochkovskiy, A., and Koltun, V. (2021). Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Rhinehart et al., 2019] Rhinehart, N., McAllister, R., Kitani, K., and Levine, S. (2019). Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2821–2830.

[Ridel et al., 2020] Ridel, D., Deo, N., Wolf, D., and Trivedi, M. (2020). Scene compliant trajectory forecast with agent-centric spatio-temporal grids. *IEEE Robotics and Automation Letters*, 5(2):2816–2823.

[Rigoulet, 2022] Rigoulet, X. (2022). Lidar: 3d perception and object detection. `https://medium.datadriveninvestor.com/lidar-3d-perception-and-object-detection-311419886bd2medium.com` [Online; posted 06-January-2022].

[Rowold et al., 2022] Rowold, M., Ögretmen, L., Kerbl, T., and Lohmann, B. (2022). Efficient spatiotemporal graph search for local trajectory planning on oval race tracks. In *Actuators*, volume 11, page 319. MDPI.

[Rozenberg et al., 2021] Rozenberg, R., Gesnouin, J., and Moutarde, F. (2021). Asymmetrical bi-rnn for pedestrian trajectory encoding. *arXiv preprint arXiv:2106.04419*.

[Rudenko et al., 2020] Rudenko, A., Palmieri, L., Herman, M., Kitani, K. M., Gavrila, D. M., and Arras, K. O. (2020). Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935.

[Sadat et al., 2020] Sadat, A., Casas, S., Ren, M., Wu, X., Dhawan, P., and Urtasun, R. (2020). Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer.

[Sadeghian et al., 2019] Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., and Savarese, S. (2019). Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1349–1358.

[Salzmann et al., 2020] Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. (2020). Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer.

[Schäfer et al., 2022] Schäfer, M., Zhao, K., Bühren, M., and Kummert, A. (2022). Context-aware scene prediction network (caspnet). *arXiv preprint arXiv:2201.06933*.

[Schlechtriemen et al., 2015] Schlechtriemen, J., Wirthmueller, F., Wedel, A., Breuel, G., and Kuhnert, K.-D. (2015). When will it change the lane? a probabilistic regression approach for rarely occurring events. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1373–1379. IEEE.

[Schubert et al., 2008] Schubert, R., Richter, E., and Wanielik, G. (2008). Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th international conference on information fusion*, pages 1–6. IEEE.

[Schwarting et al., 2019] Schwarting, W., Pierson, A., Alonso-Mora, J., Karaman, S., and Rus, D. (2019). Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978.

[Scibior et al., 2021] Scibior, A., Lioutas, V., Reda, D., Bateni, P., and Wood, F. (2021). Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation. *arXiv:2104.11212*.

[Shi et al., 2015] Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Sohn et al., 2015] Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28.

[Song et al., 2022] Song, H., Luan, D., Ding, W., Wang, M. Y., and Chen, Q. (2022). Learning to predict vehicle trajectories with model-based planning. In *Conference on Robot Learning*, pages 1035–1045. PMLR.

[Stanciulescu et al., 2009] Stanciulescu, B., Breheret, A., and Moutarde, F. (2009). Introducing new adaboost features for real-time vehicle detection. *arXiv preprint arXiv:0910.1293*.

[Streubel and Hoffmann, 2014] Streubel, T. and Hoffmann, K. H. (2014). Prediction of driver intended path at intersections. In *2014 ieee intelligent vehicles symposium proceedings*, pages 134–139. IEEE.

[Suo et al., 2021] Suo, S., Regalado, S., Casas, S., and Urtasun, R. (2021). Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409.

[Talpaert et al., 2019] Talpaert, V., Sobh, I., Kiran, B. R., Mannion, P., Yogamani, S., El-Sallab, A., and Perez, P. (2019). Exploring applications of deep reinforcement learning for real-world autonomous driving systems. *arXiv preprint arXiv:1901.01536*.

[Tang and Salakhutdinov, 2019] Tang, C. and Salakhutdinov, R. R. (2019). Multiple futures prediction. *Advances in Neural Information Processing Systems*, 32.

[Thomas et al., 2019] Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420.

[Treiber et al., 2000] Treiber, M., Hennecke, A., and Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805.

[Varadarajan et al., 2022] Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K. S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C. P., Anguelov, D., et al. (2022). Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821. IEEE.

[Vasquez and Fraichard, 2004] Vasquez, D. and Fraichard, T. (2004). Motion prediction for moving objects: a statistical approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3931–3936. IEEE.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Völz et al., 2016] Völz, B., Mielenz, H., Siegwart, R., and Nieto, J. (2016). Predicting pedestrian crossing using quantile regression forests. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 426–432. IEEE.

[Wang et al., 2021a] Wang, C., Wang, Y., Xu, M., and Crandall, D. J. (2021a). Stepwise goal-driven networks for trajectory prediction. *arXiv:2103.14107*.

[Wang et al., 2021b] Wang, W., Qie, T., Yang, C., Liu, W., Xiang, C., and Huang, K. (2021b). An intelligent lane-changing behavior prediction and decision-making strategy for an autonomous vehicle. *IEEE Transactions on Industrial Electronics*, 69(3):2927–2937.

[Welch et al., 1995] Welch, G., Bishop, G., et al. (1995). An introduction to the kalman filter.

[Weng et al., 2021] Weng, X., Ivanovic, B., and Pavone, M. (2021). Mtp: Multi-hypothesis tracking and prediction for reduced error propagation. *arXiv preprint arXiv:2110.09481*.

[Weng et al., 2022] Weng, X., Ivanovic, B., and Pavone, M. (2022). Mtp: Multi-hypothesis tracking and prediction for reduced error propagation. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1218–1225. IEEE.

[Wiest et al., 2012] Wiest, J., Höffken, M., Kreßel, U., and Dietmayer, K. (2012). Probabilistic trajectory prediction with gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium*, pages 141–146. IEEE.

[Wojke et al., 2017] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.

[Wu and Wu, 2021] Wu, D. and Wu, Y. (2021). Air^2 for interaction prediction. Waymo Open Dataset Challenges Reports, CVPR Workshop on Autonomous Driving, `http://cvpr2021.wad.vision/`.

[Xin et al., 2018] Xin, L., Wang, P., Chan, C.-Y., Chen, J., Li, S. E., and Cheng, B. (2018). Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1441–1446. IEEE.

[Xing et al., 2019] Xing, Y., Lv, C., Wang, H., Wang, H., Ai, Y., Cao, D., Velenis, E., and Wang, F.-Y. (2019). Driver lane change intention inference for intelligent vehicles: framework, survey, and challenges. *IEEE Transactions on Vehicular Technology*, 68(5):4377–4390.

[Xue et al., 2017] Xue, H., Huynh, D. Q., and Reynolds, M. (2017). Bi-prediction: Pedestrian trajectory prediction based on bidirectional lstm classification. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE.

[Yang et al., 2018] Yang, B., Luo, W., and Urtasun, R. (2018). Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660.

[Yao et al., 2021] Yao, Y., Atkins, E., Johnson-Roberson, M., Vasudevan, R., and Du, X. (2021). Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation. *IEEE Robotics and Automation Letters*, 6(2):1463–1470.

[Ye et al., 2021] Ye, M., Cao, T., and Chen, Q. (2021). Tpcn: Temporal point cloud networks for motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11318–11327.

[Yi et al., 2015] Yi, S. G., Kang, C. M., Lee, S.-H., and Chung, C. C. (2015). Vehicle trajectory prediction for adaptive cruise control. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 59–64. IEEE.

[Yoon and Kum, 2016] Yoon, S. and Kum, D. (2016). The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1307–1312. IEEE.

[Yu et al., 2020] Yu, C., Ma, X., Ren, J., Zhao, H., and Yi, S. (2020). Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision*, pages 507–523. Springer.

[Yuan et al., 2021] Yuan, Y., Weng, X., Ou, Y., and Kitani, K. M. (2021). Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823.

[Zaklouta and Stanciulescu, 2014] Zaklouta, F. and Stanciulescu, B. (2014). Real-time traffic sign recognition in three stages. *Robotics and autonomous systems*, 62(1):16–24.

[Zeng et al., 2021] Zeng, W., Liang, M., Liao, R., and Urtasun, R. (2021). Lanercnn: Distributed representations for graph-centric motion forecasting. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 532–539. IEEE.

[Zhan et al., 2019] Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., Kummerle, J., Konigshof, H., Stiller, C., de La Fortelle, A., et al. (2019). Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*.

[Zhang et al., 2020] Zhang, L., Su, P.-H., Hoang, J., Haynes, G. C., and Marchetti-Bowick, M. (2020). Map-adaptive goal-based trajectory prediction. In *CoRL*.

[Zhao et al., 2021] Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C., et al. (2021). Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR.

[Zhao et al., 2019] Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., and Wu, Y. N. (2019). Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12126–12134.

[Zhou et al., 2019] Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *arXiv:1904.07850*.

[Zhou et al., 2022] Zhou, Z., Ye, L., Wang, J., Wu, K., and Lu, K. (2022). Hivt: Hierarchical vector transformer for multi-agent motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8823–8833.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

RÉSUMÉ

Les récentes avancées dans les méthodes d'apprentissage automatique ont permis des progrès considérables dans le domaine de la conduite autonome. L'attention se porte donc désormais sur les étapes suivantes du pipeline de la conduite autonome, où la prédiction joue un rôle important. Une fois que les agents routiers environnants ont été détectés, suivis et filtrés, le système de conduite doit prédire leur trajectoire future et planifier en conséquence pour éviter les collisions. Cette prédiction de trajectoire doit répondre à de multiples exigences. Tout d'abord, elle doit être évidemment précise et sûre, afin que son résultat puisse être utilisé de manière fiable dans les processus suivants. Le futur peut présenter de multiples possibilités, qu'il n'est pas toujours possible de differencier sur la seule base des données historiques passées. La prévision doit donc être multimodale, en prédisant plusieurs futurs probables simultanés. Puisque la prévision doit être faite sur tous les agents environnants, et que les comportements de ces agents sont très influencés par leurs interactions, le modèle doit prendre en compte ces interactions, et ses prévisions multimodales doivent être cohérentes entre elles. Enfin, pour la sécurité et la fiabilité, la prédiction de trajectoire doit être facile à interpréter, largement évaluée, capable de fournir des évaluations de confiance et conçue avec son utilisation finale dans le processus global à l'esprit.
Nous proposons d'aborder le problème de la prédiction de trajectoire en utilisant des grilles probabilistes pour faciliter la multimodalité. Nous concevons trois manières différentes de générer ces cartes thermiques et nous les évaluons les unes par rapport aux autres et par rapport à l'état de l'art existant. Nous fournissons également une méthode d'extraction complète pour obtenir les trajectoires réelles à partir de ces cartes de probabilités, et nous étudions les avantages et les inconvénients de ces méthodes de grilles par rapport à d'autres approches couramment utilisés. Dans le chapitre suivant, nous nous concentrons sur la prédiction multi-agents, et plus particulièrement sur les prédictions cohérentes au niveau de la scène, pour ce type de modèles de grilles par le biais de l'extraction et d'une seconde étape apprise. Enfin, nous explorons différentes manières d'étendre l'évaluation des modèles de prédiction par l'évaluation de l'incertitude, la calibration et l'analyse de la généralisabilité entre jeux de données.

MOTS CLÉS

Prediction de Trajectoire, Apprentissage automatique, Voiture autonome

ABSTRACT

Recent advances in machine learning methods have enabled tremendous progress in autonomous driving. The focus is now therefore shifting towards the next steps in the autonomous pipeline, where prediction plays an important role. Once the surrounding road agents have been detected and tracked, the driving system needs to predict their future trajectory and plan accordingly to have a collision-less course.
This trajectory prediction must follow multiple requirements. First, it should obviously be accurate and trustworthy, so that its output can be reliably used in the following processes. The future can present multiple possibilities, from which it may not always be possible to disambiguate solely based on past historical data. The forecast must therefore be multimodal, by predicting multiple simultaneous probable futures. Since the prediction is to be made on all surrounding agents, and these agents behaviors are very much influenced by their interactions with each other, the model should take these interactions into account, and its multimodal predictions should be coherent with each other. Finally, for safety and reliability, the trajectory prediction should be easy to interpret, extensively evaluated, able to provide confidence evaluates and designed with its final use in the pipeline in mind.
We propose to tackle the trajectory prediction problem using probability heatmaps to facilitate multimodality. We design three different ways of generating these heatmaps and evaluate them against each other and the existing state-of-the-art. We also provide a complete sampling method to extract actual trajectories from these heatmaps, and study the pro and cons of these heatmap methods compared to other commonly used frameworks. In the next chapter, we focus on multi-agent prediction, and more specifically consistent scene-level outputs, for these type of heatmap models through sampling and learned post-processing. Finally, we explore different ways of expanding prediction model evaluation by uncertainty assessment, calibration and cross-dataset generalizability analysis.

KEYWORDS

Trajectory Prediction, Machine learning, Autonomous Driving