Integrating Expert Knowledge with Deep Reinforcement Learning Methods for Autonomous Driving

Raphaël Chekroun

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Mines Paris

Integrating Expert Knowledge with Deep Reinforcement Learning Methods for Autonomous Driving

Intégrer des connaissances expertes dans des méthodes d'apprentissage par renforcement profond pour la conduite autonome

Soutenue par

Raphael Chekroun

Le 2 avril 2024

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité

Informatique temps réel, robotique et automatique.

Composition du jury :

David Filliat Professeur, ENSTA Paris	*Président*
Philippe Martinet Dir. recherche, INRIA Sophia	*Rapporteur*
Emmanuel Rachelson Professeur, ISAE-SUPAERO	*Rapporteur*
Fawzi Nashashibi Dir. recherche, INRIA Paris	*Examinateur*
Sascha Hornauer Maître de conférences, Mines Paris	*Examinateur*
Maria-Laura Delle Monache Assistant Professor, UC Berkeley	*Examinateur*
Marin Toromanoff Valeo Driving Assistant Research	*Examinateur*
Fabien Moutarde Professeur, Mines Paris	*Directeur de thèse*

MINES PARIS | PSL

Abstract

Two decades after the first autonomous driving challenge [1], which had no winners successfully navigating a 240 kilometers desert road in Mojave, the advancement of machine learning has brought remarkable progress to this field. Notably, the creation of open-source simulators made research for autonomous driving easier by sidestepping regulatory constraints and providing an affordable way to collect data. This, combined with the rise of neural networks, has expedited development of increasingly efficient methods. Recent research for motion planning mostly focuses on imitation learning (IL) and, to a lesser extent, on reinforcement learning (RL). By learning from data, machine-learning based methods are more adaptable than rule-based ones as they rely less on perfect and consistent representation of the environment. Nevertheless, IL approaches remain limited in grasping the long term consequences of their actions and suffer robustness issues stemming from distribution mismatch. Conversely, RL incorporates long-term return information and successfully overcomes distribution mismatch by learning through trial-and-error. However, it suffers from sample inefficiency, instability during training, and lacks of convergence guarantees. This thesis aims to synergize the strengths of both approaches while mitigating their weaknesses by integrating expert knowledge with deep reinforcement learning methods for different autonomous driving applications.

After recapitulating existing methods for autonomous driving, this thesis investigates how to introduce expert knowledge in reinforcement learning algorithms for several autonomous driving tasks. Firstly, we introduced a novel method for distilling expertise in model-free RL and applied it to end-to-end autonomous driving on the CARLA simulator. Secondly, we developed an approach leveraging an IL-based prior to guide a model-based RL algorithm in a partially learned model of the environment for mid-to-end autonomous driving on the nuPlan simulator. Finally, we designed a real-time mesoscale traffic forecasting module to be leveraged by a model-free RL centralized speed planner within a hierarchical control framework for real world traffic dissipation on highways leveraging a fleet of 100 Connected and Autonomous Vehicles (CAVs).

[1] DARPA Grand Challenge: https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles

Résumé en Français

Deux décennies après le premier défi de conduite autonome, qui n'a vu aucun gagnant réussir à naviguer 240 kilomètres de route désertique dans le désert de Mojave, les évolutions en apprentissage automatique ont permis d'importants progrès dans ce domaine. En particulier, la création de simulateurs open-source a facilité la recherche en matière de conduite autonome en permettant d'outrepasser les contraintes réglementaires et en offrant un moyen abordable de collecter des données. Cela, combiné à la montée des réseaux de neurones, a accéléré le développement de méthodes de plus en plus efficaces. Les recherches récentes en matière de planification de mouvement se concentrent principalement sur l'apprentissage par imitation et, dans une moindre mesure, sur l'apprentissage par renforcement. En apprenant à partir de données, les méthodes d'apprentissage automatique sont plus adaptables que celles basées sur des systèmes de règles, car elles dépendent moins d'une représentation parfaite et cohérente de l'environnement. Néanmoins, les approches par imitation restent limitées dans la compréhension des conséquences à long terme de leurs actions et rencontrent des problèmes de robustesse résultant d'une inadéquation de distribution. En revanche, l'approche par renforcement intègre des informations de retour à long terme et surmonte avec succès les problèmes de distribution en apprenant par essais et erreurs. Cependant, cette approche souffre d'inefficacité d'échantillonnage, d'instabilité pendant l'entraînement et d'un manque de garanties de convergence. Cette thèse vise à synergiser les points forts des deux approches tout en atténuant leurs faiblesses en intégrant des connaissances expertes avec des méthodes d'apprentissage par renforcement profond pour différentes applications liées à la conduite autonome.

Après avoir récapitulé les méthodes existantes en matière de conduite autonome, cette thèse examine différentes facons d'introduire des connaissances expertes dans des algorithmes d'apprentissage par renforcement pour plusieurs tâches de conduite autonome. Tout d'abord, nous avons introduis une nouvelle méthode pour distiller de l'expertise dans un apprentissage par renforcement sans modèle et l'avons appliquée à la conduite autonome de bout en bout sur le simulateur CARLA. Ensuite, nous avons développé une approche tirant parti d'une base d'apprentissage par imitation pour guider un algorithme d'apprentissage par renforcement basé sur modèle dans un modèle partiellement appris de l'environnement pour la conduite autonome de milieu-à-fin sur le simulateur nuPlan. Enfin, nous avons conçu un module de prévision du trafic à l'échelle mésoscopique en temps réel, fait pour être utilisé avec un planificateur de vitesse centralisé basé sur l'apprentissage par renforcement sans modèle dans le cadre d'un controleur hiérarchique pour la dissipation du trafic en temps réel sur les autoroutes en utilisant une flotte de 100 véhicules connectés et autonomes.

Remerciements

Tout d'abord, un grand merci à Philippe Martinet, David Filliat et Fawzi Nashabishi d'avoir rapporté et/ ou examiné ma thèse. Merci de votre bienveillance et de l'échange constructif qui a suivi ma présentation. Merci à Thibaut Buhet, membre invité du jury et chef d'équipe chez Valeo, pour ton aide tout le long de ma thèse et tes remarques bienveillantes lors de la soutenance.

Merci à Fabien Moutarde pour ta direction et ces nombreuses heures de dicussions sur des sujets divers et variés, et à Marin Toromanoff pour tes conseils techniques et nos intenses "bootcamp" rédactions avant chaque soumission. Merci à Maria Laura Delle Monache pour ton encadrement à Berkeley et pour cette expérience californienne que tu m'as permis de vivre. Merci à Sascha Hornauer pour ton soutien et ta précieuse aide de relecture de papiers et de thèse. Ces recherches n'auraient pas pu aboutir sans votre expertise, votre soutien et votre bienveillance.

Mais au-delà de l'expérience intellectuelle et technique et de la fierté du travail accompli, ces trois années de thèse ont aussi étés plaisantes et enrichissantes d'un point de vue personnel. Merci aux copains de la V026 Arthur, Camille, Jesus, Joseph, Thomas auprès de qui le CAOR et le quartier latin sont devenus tous les mardis soir un lieu de fête. Merci aussi à Sami, Sofiane, Amandine, Simon, Angelika, Nassim, et tous les autres pour avoir fait de ce doctorat une période si agréable. Merci aussi à l'équipe DRIL de Valeo. Merci Thibault et Marin pour votre aide et vos conseils, et merci Auguste, Valentin, Waël, Thomas, Amine et Antoine pour ces pauses cafés, les soirées jeux, et vos collaborations techniques. Et Waël : bon courage pour ta thèse ! Je te lègue avec plaisir tous mes équipements chez Valeo et aux Mines. Merci aussi à tous mes amis du lycée, de l'ENS, du MVA, et tous les autres avec qui je partage depuis de nombreuses années tous les grands moments de ma vie d'avoir été là du début de ma thèse jusqu'à ma soutenance.

Mais je n'aurais jamais pu en arriver aussi loin sans l'aide de mes parents. Merci pour votre amour, vos conseils et votre guidance qui m'ont permis de toujours me dépasser, en particulier lorsque l'école et le besoin d'attention en cours me semblait une épreuve insurmontable. Je suis fier de l'homme que je suis devenu, et je n'en remercierai jamais assez leurs auteurs. Merci à mes soeurs et beaux-frères Eva & Sam et Elisa & Gad pour votre soutien constant. Merci à ma nièce Ora et mon neveu Ariel d'égayer les weekends en famille et de m'apprendre à nouveau à m'amuser avec de la pâte à modeler virtuelle. Puisse la famille toujours s'agrandir avec autant de joie ! Merci à mes futurs beaux-parents Carole et Olivier de m'accueillir chez eux comme leur fils et avec autant d'amour. J'ai trouvé avec vous ma deuxième famille et j'en suis très reconnaissant. Merci à Laura & Elie, et à Lalou de m'avoir adopté comme frère. Enfin, et pas des moindres, un grand merci à ma fiancée et (très proche) future femme Axelle. Merci de m'avoir toujours soutenu dans cette aventure, dans les moments productifs ou les périodes creuses, et merci de tout le bonheur que l'on vit ensemble. Tu es ma meilleure amie, et j'ai hâte de tout ce qu'on va continuer à vivre ensemble. 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 the integration of expert knowledge in reinforcement learning setups for autonomous driving systems.

1.1 Context

1.1.1 The autonomous driving pipeline

Autonomous driving (AD) systems are traditionally built following an architecture made of three main components, illustrated in Figure 1.1. This pipeline runs at each time step so the vehicle can plan its trajectory and adapt to its dynamic environment.

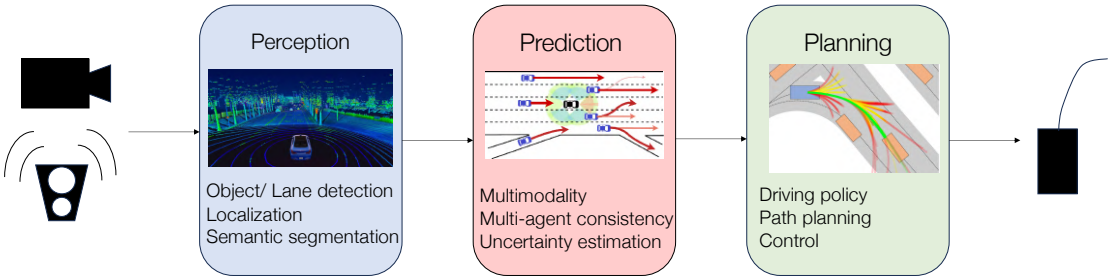


Figure 1.1: Autonomous driving pipeline. Illustrations from [Velodyne, 2018, Deo and Trivedi, 2018, Chekroun et al., 2023a]

Perception The perception module process sensors (cameras, LiDARs, radars, etc...) signals to extract high level semantic information of the vehicle's environment, such as the map layout, other agents position or object detection, or lower level information such as depth or segmentation maps. This module can leverage handcrafted solutions [Lowe, 2004, Moutarde et al., 2007, Labatut et al., 2007, Deschaud and Goulette, 2010], but more recent deep learning methods allows significantly improved image processing [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2016] in all perception-related tasks such as object detection [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, Liang et al., 2018b, 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 on radar or LiDAR data [Qi et al., 2017a, Yang et al., 2018, Lang et al., 2019, Thomas et al., 2019].

Prediction The prediction module is fed the processed static representation of the environment around the ego agent and aims at inferring its evolution in the next few seconds *i.e.* its dynamics. In other term, the prediction module should allow the ego agent to estimate its next state given its current state and its chosen action. While most state transitions can be derived from kinematics, some of elements of the transition might be independent from agent's action and past observations (traffic light states, other agents at an intersections, etc...). Also, trajectory estimation of other agents should consider both agent-to-environment and agent-to-agent interactions.

Planning From the knowledge of the ego's environments and its dynamics in the next few seconds, the planning module builds a plan to follow for the ego agent, so it can reach its destination while respecting traffic rules, avoiding collisions and staying in the designated drivable area. This module's objective is to generate a safe and reliable sequence of actions for the ego to follow to form a trajectory. Steering and throttle orders are derived from this trajectory by the ego vehicle controller.

However, for the past few years, progress in deep learning and computer vision for the perception module has led the research focus of Automated Driving (AD) to shift towards improving motion planning and the introduction of machine-learning in this module. This trend led to the emergence of two competitive approaches: mid-to-end AD, which leverages the traditional pipeline and consider perception to be solved to focus on motion planning, and end-to-end AD which aims at simplifying the approach and learn ego planning directly from sensory inputs.

1.1.2 Mid-to-end and end-to-end autonomous driving

In the realm of autonomous driving, motion planning can be tackled in different ways, each with its own advantages and weaknesses.

End-to-end autonomous driving refers to a system where a single neural network directly processes raw sensor inputs, such as camera images and LIDAR data, and outputs control commands such as steering angles and throttle values or trajectory planning via successive waypoints. This approach offers simplicity of implementation as it eliminates the need for complex intermediate modules, making it easier to deploy. However, training an end-to-end system can be difficult as it implicitly learns to solve various tasks. End-to-end systems also lack of interpretability thus making it challenging to understand how the network makes decisions, which is a critical aspect for safety and regulatory purposes. Additionally, end-to-end systems might require extensive and diverse data for training, which can be resource-intensive.

Mid-to-end autonomous driving, on the other hand, assesses perception to be already solved and focuses on the task of prediction and planning. This approach allows for more interpretability as the module in fault can be detected if unexpected behaviors are observed. Its modularity also makes the debugging easier, and specific improvement of a given task possible without overhauling the entire system. Nonetheless, predefined inputs and outputs of individual sub-systems might not be optimal for the final driving task in different scenarios, and interactions between each module can lead to unexpected side effects. Coordination between prediction and planning stages can be complex, and handcrafting these modules might limit the system's ability to handle diverse and unpredictable real-world scenarios.

In summary, end-to-end autonomous driving offers simplicity of implementation and overall adaptability but can be harder to train and lacks interpretability, while mid-to-end approaches provide more transparency and modularity but can be challenging to integrate. The choice between these approaches depends on the specific requirements of the autonomous driving application and the balance between complexity, interpretability, and performance.

1.1.3 Simulators in Autonomous Driving

Simulators were developed for research in AD primarily to overcome the need of data collection, which can be expensive and time consuming, and the risks associated with real-world model training and validation. Indeed, simulators offer a safe, controlled, and cost-effective alternative, providing a virtual environment where various driving scenarios, including rare or dangerous situations, can be simulated with high fidelity. This allows for extensive and diverse data collection, and facilitates rapid iteration and testing of algorithms without the logistical and regulatory constraints of on-road testing, therefore significantly accelerating the development of reliable autonomous driving technologies.

CARLA simulator [Dosovitskiy et al., 2017] was released in 2017 and became one of the most prominent open-source playground for the end-to-end autonomous driving community. CARLA is an urban driving simulator of synthesis images incorporating realistic vehicle physics, different towns to drive in, with various weather, and diverse pedestrians and vehicles designs. CARLA allows the ego agent to be equipped with various sensors at inference (cameras, LiDAR...) and even a more complete set of cameras to gather ground truth for datasets (depth camera, semantic segmentation camera...), see Figure 1.2.

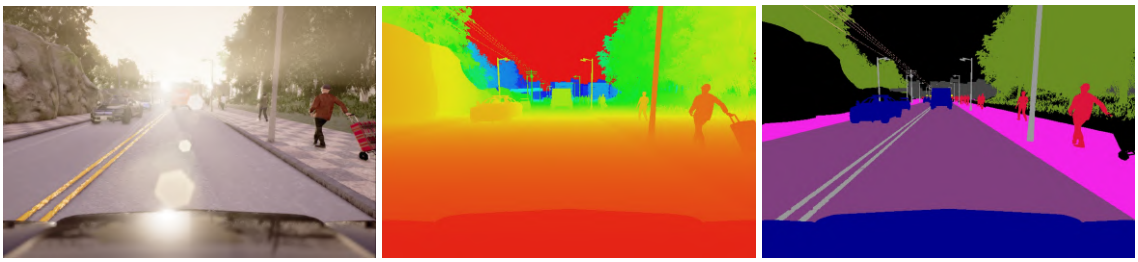


Figure 1.2: Three of the sensing modalities provided by the CARLA simulator. From left to right: normal vision camera, ground-truth depth, and ground-truth semantic segmentation. Figure from [Dosovitskiy et al., 2017].

Since 2019, CARLA has been hosting the annual CARLA Challenge, a public contest centered on autonomous driving using the CARLA simulator. This challenge focuses on driving in urban settings and aims to navigate through unfamiliar maps, using sensor data for vehicle control. Key objectives include maintaining proper lane discipline, following high-level navigation instructions at intersections (such as turning right, left, or continuing straight), executing lane changes, and avoiding pedestrians and other vehicles. Additionally, the challenge involves recognizing and responding to traffic lights as per both US and European standards.

In 2021 Motional released nuPlan [Caesar et al., 2021], another simulator aimed at both mid-to-end and end-to-end research in autonomous driving. Contrarily to CARLA, nuPlan is based on captured images of real scenes and consists of 1500 hours of driving in several cities and on different scenarios (going straight, stuck in traffic, turning left, on a drop-off location...). nuPlan dataset not only provides sensors data, but also higher level representations (bounding boxes, polylanes...) for mid-to-end autonomous driving, see Figure 1.3.

Lastly, Waymo released Waymax in 2023 [Gulino et al., 2023], unfortunately too late to conduct research on it during this PhD. Waymax is a multi-agent simulator for autonomous driving

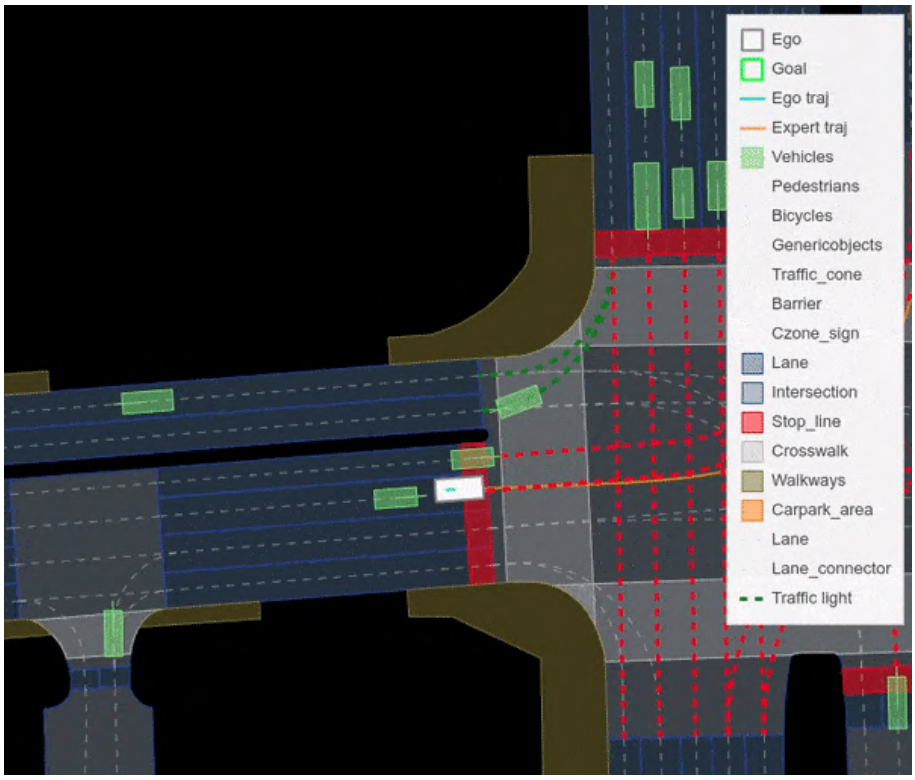


Figure 1.3: Vizualisation of nuPlan simulator supported features. Other vehicles oriented bouding boxes, lanes, crosswalks, traffic lights positions and states, etc are provided and can be obtained in a vectorized way via requests to the API. Figure from nuPlan website.

research based on the Waymo Open Motion Dataset for mid-to-end autonomous driving. It contains different types of scenarios, as represented in Figure 1.4. Waymax is designed to support research for all aspects of behavior research in autonomous driving - from closed-loop simulation for planning and sim agent research to open-loop behavior prediction.

1.2 Reinforcement Learning and Expert Knowledge

1.2.1 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to behave in an environment by performing actions and receiving rewards or penalties in return [Sutton and Barto, 2018]. The agent's objective is to maximize the cumulative reward over time. It's based on the principle of trial-and-error, whereby an agent makes decisions by acting in its environment, observing the outcomes, and then refining its strategy based on the feedback received. A fundamental element of RL is the balance between exploration (trying out new actions) and exploitation (using actions that yield the highest reward according to the current knowledge of the environment). The agent uses this feedback to update its knowledge and improve its future actions. The RL principle is represented in Figure 1.5. This process can be mathematically described using Markov Decision Processes (MDPs). RL is a vast field of research that can be divided into two

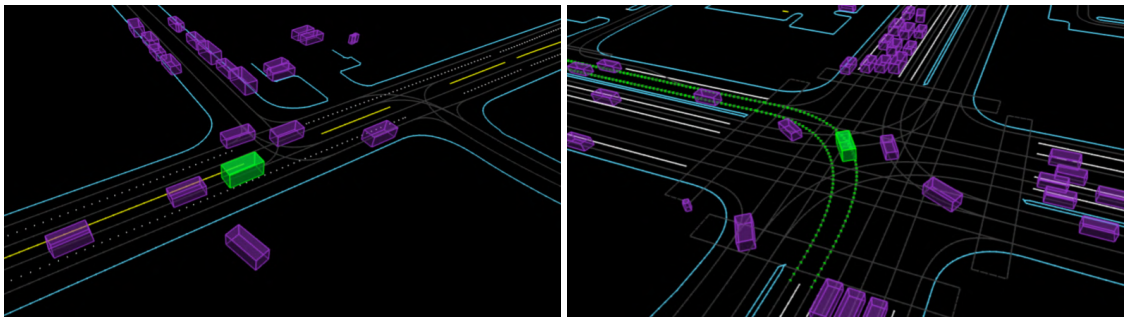


Figure 1.4: Two examples demonstrating the types of interactive, urban driving scenarios available in Waymax. **(a)** shows a vehicle waiting for oncoming traffic to pass before turning into a narrow street. **(b)** shows an agent performing an left turn at a 4-way intersection while following a route (boundaries highlighted in green). Figure and caption from [Gulino et al., 2023].

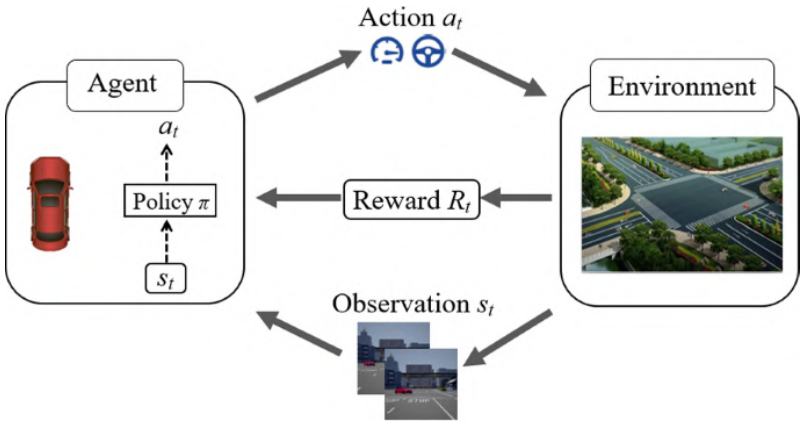


Figure 1.5: Representation of agent and environment interaction in RL. For a given state s_t, the agent chose an action a_t following a learned policy π which lead to a new state s_{t+1} and give a reward r_{t+1}. Figure from [Li et al., 2020a]

primary approaches: model-free and model-based.

Model-free reinforcement learning methods aim to learn an optimal policy directly from interactions with the environment without explicitly modeling its dynamics. The advantage of this approach is that the agent does not need to understand how the environment works, it simply needs to determine what actions yield the highest cumulative rewards. Two of the most prevalent techniques within this category are Value-based and Policy-based methods. In value-based methods, like DQN [Mnih et al., 2013], the agent seeks to learn its incentive of taking a particular action in a given state, represented as a Q-value which is the expected cumulative reward. In policy-based methods, such as REINFORCE [Williams, 1992] or Proximal Policy Optimization (PPO) [Schulman et al., 2017a], the agent directly optimizes the policy function without using an intermediate value function. They are by construction flexible and broadly applicable to different kind of applications as they do not require any prior modelisation and learn intuitions on the consequence of agents' actions directly by interacting with their environments. However, the training of such models is notably unstable and require a large amount of data to converge to an optimal

policy since they operate based on trial-and-error. Using deep neural network as functions approximations even removes all theoretical guarantees on the convergence and reliability of model-free reinforcement learning algorithms, thus leading to instabilities in practice.

Model-based reinforcement learning, as the name suggests, incorporates a model of the environment's dynamics. Instead of just relying on trial-and error, the agent seeks to understand how the environment responds to different actions. By doing so, the agent can predict the outcomes of its actions before taking them, leading to more informed decisions. The primary steps in this approach include modeling the environment, followed by a planning using this model to decide on the best course of action, and finally acting in the actual environment. Given that it utilizes a model, this approach is more data-efficient compared to model-free methods. However, the accuracy of a model-based approach largely depends on the fidelity of the model. If the model does not capture the environment's dynamics accurately, the agent's policy can be suboptimal or even detrimental therefore leading to safety-critical behavior.

Overall, model-free RL excels in its simplicity and general applicability, as it doesn't require a model of the environment and can adapt to various situations through trial and error. However, it often requires a large amount of data and can be inefficient in learning, especially in complex environments. On the other hand, model-based RL can learn more efficiently by utilizing a model of the environment to plan and predict outcomes, leading to quicker learning with less data. Yet, its effectiveness is heavily dependent on the accuracy of the environment model, and it may struggle in environments where accurate modeling is difficult or infeasible. While both RL approaches showed promising results, their inherent flaws and instability make them hardly applicable to real-life autonomous driving.

1.2.2 Expert Knowledge

Another approach for learning to drive is by leveraging expert data. In particular, one can train a neural network to mimic an expert behavior via Imitation Learning (IL). This approach offers a stabler training, but does not bear information on the long term consequences of the agents actions. Thus, IL stability is complementary with RL more complete understanding of its environment.

In this thesis, we qualify as Expert Knowledge both data obtained by gathering demonstration from an expert, or via a supervised network trained on these demonstrations. In the realm of autonomous driving, trained network generally aims at mimicking expert behavior via Imitation Learning (IL) and Expert Knowledge bears information of expert-level driving policies. However, training based on Expert Knowledge are limited by their lack of generality as they incorporate observed behavior only and do not allow to build an intuition on the consequences of an agent's action on his environment outside of the scope of the recorded data.

1.3 Problem statement: Integrating expert knowledge in RL for AD

This thesis aims at exploring how to overcome inherent limitations of both model-based and model-free reinforcement learning driving setups by leveraging expert knowledge on diverse autonomous driving related tasks, which are end-to-end and mid-to-end autonomous driving, and Connected and Autonomous Vehicles (CAVs) for traffic dissipation.

We first present a method to distill expert knowledge in a model-free deep reinforcement learning setup for end-to-end autonomous driving in urban environment. Then, we extend a model-based method to follow a learned prior in a partially-learned environment for mid-to-end autonomous driving in several urban scenarios. Finally, we introduce a learning-based traffic forecasting method aiming at improving prediction in a model-free RL hierarchical speed planner mid-to-end pipeline for traffic dissipation.

1.4 Publications, awards, and communications

This PhD has been conducted in the Center for Robotics of Mines Paris, PSL University in the context of a CIFRE collaboration with Valeo Driving Assistance Research, in the Deep Reinforcement and Imitation Learning (DRIL) team.

1.4.1 First author publications

First author publications are:

- Chekroun et al., GRI: General Reinforced Imitation and its Application to Autonomous Driving. Presented in ML4AD workshop in NeurIPS 2021. Revised version published in Robotics 2023, 12(5), 12.
- Chekroun et al., MBAPPE: MCTS-Built-Around Predictions for Planning Explicitely. Submitted in IEEE IV 2024.
- Chekroun et al., Mesoscale Traffic Forecasting for Real-Time Bottleneck and Shockwave Prediction. Submitted in Transportation Research - Part C.

1.4.2 Second author publications

Second author publications are:

- Bujalance Martin et al., Learning from demonstrations with SACR2: Soft Actor-Critic with Reward Relabeling. Presented in Deep RL workshop in NeurIPS 2021.
- Wang et al., Hierarchichal Speed Planner for Automated Vehicles: A Framework for Lagrangian Variable Speed Limit in Mixed Autonomy Traffic. Accepted in IEEE Control Systems Magazine.

- Lee et al., Traffic Control via Connected and Automated Vehicles: An Open-Road Field Experiment with 100 CAVs. Accepted in IEEE Control Systems Magazine.

1.4.3 Awards, communication and events

Communication and events are:

- Laureate of the CARLA Challenge 2021. Invited speaker in ML4AD workshop in NeurIPS 2021. December 2021.
- Visiting scholar position in University of California, Berkeley. Under the supervision of Maria Laura Delle Monache. August 2022 - January 2023.
- Laureate of the Honorable mention for Innovation award in nuPlan challenge 2023. May 2023.
- Invited speaker in the workshop Industrial Reinforcement Learning. November 2023.

1.5 Outline

This thesis consists of 7 chapters:

Chapter 1: Introduction. We contextualize our work by presenting the global autonomous pipeline and the different ways to approach it. We then introduce the notion of reinforcement learning and expert knowledge before stating this thesis objective.

Chapter 2: Non-learning based Motion Planning. We recapitulate existing non learning-based methods for autonomous driving and motion planning. In particular, we introduce families of model based of physical kinematics, interactions between agents of different types, probabilistic approaches and finally hierarchical models based on high level planning or graph search.

Chapter 3: Machine Learning for Motion Planning. We present an overview of machine learning based approaches for motion planning. After introducing the different components of a neural network, our focus lays first on imitation learning methods, and then on reinforcement learning approaches for autonomous driving.

Chapter 4: Expertise Distillation in Model-Free RL for End-to-end Autonomous Driving. This chapter introduces a method distilling expert knowledge in a reinforcement learning setup. This method goes through ablation studies and hyperparameter analysis on the Mujoco environment, and is finally applied to end-to-end autonomous driving on the CARLA simulator. We justify the advantage of this method over vanilla reinforcement learning via an ablation study and benchmark the performance of our approach against the existing state-of-the-art, both prior and after the release of out method. Lastly, we present some qualitative insights and limitations of the proposed methods.

Chapter 5: Leveraging Learned Prior in Model-Based RL for Mid-to-End Autonomous Driving. We present a method where a neural network trained via imitation learning is leveraged as a prior to guide a MCTS exploration of a partially-learned environment of the nuPlan simulator

for mid-to-end autonomous driving. After detailing the designed MCTS and tree steps, we justify some core elements of our approach through an ablation study and comparison with existing state-of-the-art. Finally, the explainability and explicit nature of this method is highlighted via a qualitative analysis.

Chapter 6: Traffic Forecasting for RL-based Traffic Dissipation. This chapter introduces the CIRCLES consortium and our work on traffic dissipation with connected and autonomous vehicles (CAVs). We present the open-road field experiment in mixed-autonomy leveraging 100 CAVs that took place in November 2022 near Nashville, TN and the hierarchical control framework designed to guide those vehicles. In particular, we detail the next iteration of a specific component of the overall system and focus on real-time mesoscale traffic forecasting which output is to be leveraged by the model-free reinforcement learning decentralized speed planner. We justify our approach through ablation studies and comparison with existing methods. Lastly, we provide qualitative observations and analysis of different approaches behavior in key traffic stages to provide more insights on the tackled problem.

Chapter 7: Conclusion. Finally, we summarize our contribution to the fields, and discuss potential limitations of our approaches and how to tackle them.

Chapter 2

Literature review: from kinematic models to non learning-based motion planning

Contents

2.1 Physical kinematics models

Future position of a vehicle can, with the knowledge of its initial speed v and heading θ and the assumption they are constants in a short time δt, be approximated via iterative call of the Constant Velocity (CV) model, mathematically represented in Equation 2.1.

$$\begin{pmatrix} x_{t+\delta t} \\ y_{t+\delta t} \end{pmatrix} = \begin{pmatrix} x_{\delta t} \\ y_{\delta t} \end{pmatrix} + v\delta t \begin{pmatrix} \cos\theta \\ sin\theta \end{pmatrix} \tag{2.1}$$

However this first order approximation no longer yields coherent physical behavior when the horizon time δt become too big as vehicles may be accelerating or turning. Assuming the Constant Turn Rate (ω) and Acceleration (a) (CTRA) model, future positions of a vehicles can be computed via iterative call of the set of equations represented in Equation 2.2.

$$\begin{aligned} \begin{pmatrix} x_{t+\delta t} \\ y_{t+\delta t} \end{pmatrix} &= \begin{pmatrix} x_{\delta t} \\ y_{\delta t} \end{pmatrix} + v\delta t \begin{pmatrix} \cos\theta \\ sin\theta \end{pmatrix} \\ v_{t+\delta t} &= v_t + a\delta t \\ \theta_{t+\delta t} &= \theta t + \omega\delta t \end{aligned} \tag{2.2}$$

This quantities can be extracted from sequences of past positions with a Kalman filter [Welch et al., 1995]. Classically, these filters require hand-tuned parameters for initial uncertainty values, but recent research leveraged learning-based methods to obtain these values through backpropagation of the filter prediction errors [Jouaber et al., 2021]. In the realm of autonomous driving, Kalman filters generally use a derivative of the CTRA model hypothesizing a constant steering value δ and computing the turn rate ω from the steering, wheelbase L and speed via the Equation 2.3.

$$\omega = \frac{v}{L} \times \tan\delta \tag{2.3}$$

This model is called the bicycle model and its elements are represented in Figure 2.1.

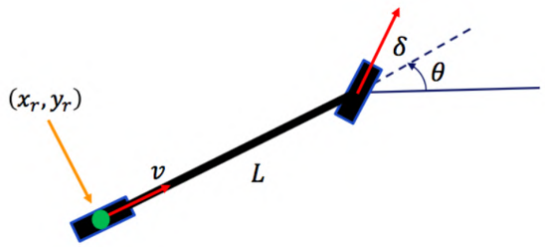


Figure 2.1: Bicycle model with constant steering angle δ and wheelbase L. Figure from [Ding, 2020]

2.2 Interaction models

2.2.1 Intelligent driver model

The Intelligent Driver Model (IDM) [Treiber et al., 2000a] is a behavior model designed to directly control the acceleration of an agent by following a lead vehicle while respecting a safety distance and a comfortable behavior with limited acceleration and braking deceleration. IDM is a rule-based method mathematically defined in the Equation 2.4.

$$\frac{dv}{dt} = a * \left(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right) \tag{2.4}$$

with $\Delta v = v - v_t$ the speed of approach of the lead vehicle, δ an experimentally obtained constant (generally $\delta = 4$) used to compare v and the desired speed v_0. s is the observed distance between the agent and the vehicles it follows, and s^* is computed with Equation 2.5.

$$s^*(v, \delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right) \tag{2.5}$$

with T the time-gap to the leading vehicle, s_0 the minimum authorized spacing, a the maximum authorized acceleration, and b the authorized braking deceleration. In Equation 2.5, $\frac{v\Delta v}{2\sqrt{ab}}$ represents the dynamic part implementing the "intelligent" braking strategy where the kinematic acceleration necessary for safety is safe-regulating towards the comfort deceleration.

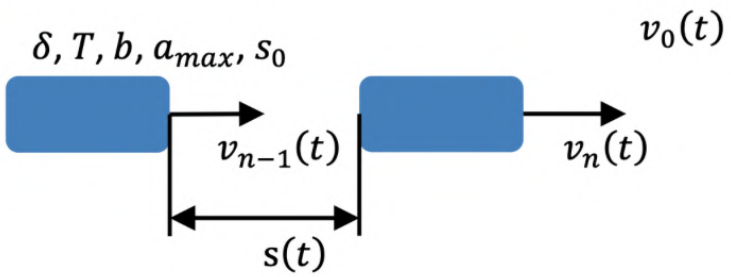


Figure 2.2: Generalized notations of the IDM model with n following cars. Figure from [Salles et al., 2022].

The IDM is designed to make an agent follow leading cars, mainly for traffic simulation of collision avoidance, and the scenario covered are very specific (e.g. highways). It is more adaptive to its surrounding than physical models, but yields less realistic behavior. Several works aims at improving the realism of IDM e.g. by computing adaptive gaps between vehicles [Salles et al., 2022] or bounding acceleration and deceleration values [Albeaik et al., 2022].

2.2.2 Pedestrian interactions

To reach a safe and reliable driving, the planner must consider evolution of all the agents in the scene, whether they are other vehicles or pedestrians. However, pedestrians dynamics is fairly

different from that of vehicles. Indeed, while cars must avoid collisions, most of the decision making consist of following lanes and strict driving rules in a very constrained environment. On the other hands, pedestrians evolve in unconstrained environments and their motion is influenced almost only by other pedestrians in crowds. Therefore, [Helbing and Molnar, 1995] modelized pedestrian motion planning through the application of social forces, where other pedestrians and obstacles, including sidewalks limits, are repulsive forces, and aimed destination are attractive forces.

Another approach to modelize crowds movements or vehicle traffic is via fluid dynamics systems [Henderson, 1974]. However, those methods mainly aims at macroscopic-level modelization and do not allow accurate microscopic predictions, as represented in Figure 2.3.

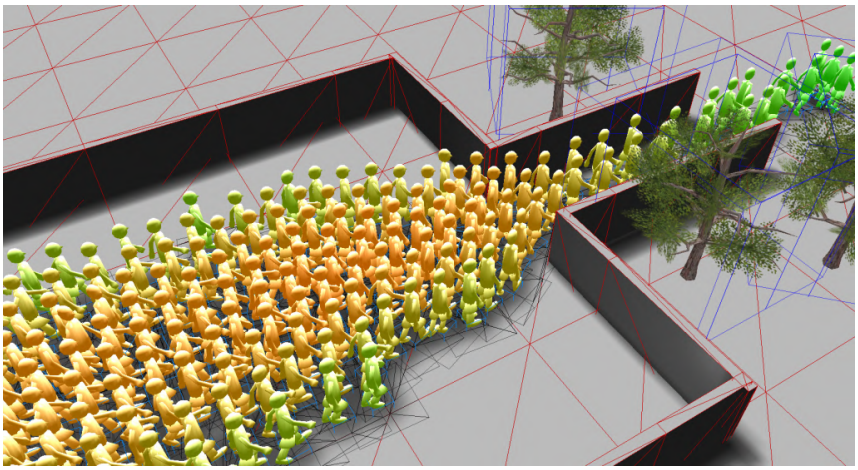


Figure 2.3: Pedestrian crowds movements can be modelized as a fluid dynamic problem at the macroscopic scale. Figure from vadere.org.

A data driven approach to modelize pedestrian interactions is via social pooling. [Alahi et al., 2016] introduced a new LSTM model which jointly reasons across multiple individuals in a scene by sharing information between their respective LSTMs through a pooling layer. The same kind of pooling mechanism has been leveraged with Generative Adversarial Networks (GANs) to generate socially acceptable pedestrians trajectories [Gupta et al., 2018].

2.2.3 Game-theory approaches

Multi-agent motion forecasting can be modelized as a game solving problem where each agent aims at maximizing its own internal reward while considering other agents rewards [Schwarting et al., 2019]. This model can be solved via a Stackelberg game [Li et al., 2017] or a game tree [Bahram et al., 2015]. In this game, each vehicle successively chooses an action in order to reach a Nash equilibrium *i.e.* a solution where no player can increase its reward by solely changing its action.

2.3 Probabilistic planning models

2.3.1 Hidden Markov models

The car trajectory can be represented as a sequence of partial observation x derived from a hidden variable z and modelized as a Hidden Markov Model (HMM) [Berndt and Dietmayer, 2009, Firl et al., 2012, Christopher, 2009]. Following the Markov chain hypothesis, s_t distribution of probability only depends of s_{t-1}, and x_t depends on s_t only, as represented in Figure 2.4. Therefore, HMM are particularly fitted for sequential data. In HMM framework, the joint probability of s and x can be computed via Equation 2.6.

$$p(x,s) = \prod_{t}^{T} p\left(x_t \mid s_t\right) p\left(s_t \mid s_{t-1}\right) \tag{2.6}$$

with $p\left(x_t \mid s_t\right)$ the emission probability and $p\left(s_t \mid s_{t-1}\right)$ the transition probability. Hence, HMM can be seen as a generative model representing underlying distributions of probability.

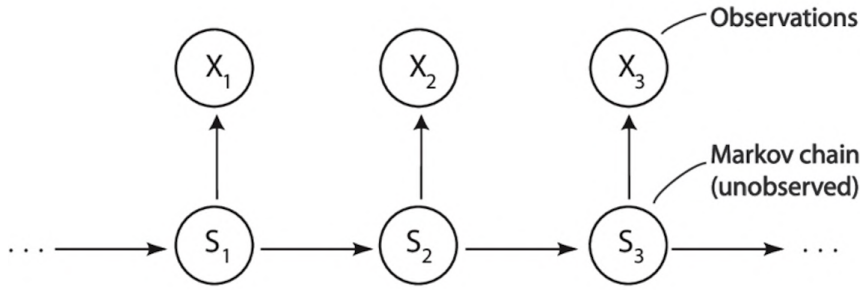


Figure 2.4: Representation of hidden states s and observable states x in an HMM. Figure from dlab.berkeley.edu.

2.3.2 Conditional Random Fields

Like HMM, Conditional Random Fields (CRF) can also be used to modelize sequences of partially observable observations but are discriminative models for the conditional probability $p(s|x)$ only. In the realm of autonomous driving, [Ohn-Bar et al., 2015] leveraged CRFs to predict latents events of braking or overtaking.

2.3.3 Bayesian networks

Another approach are Bayesian networks, which are less focused on temporal dependencies than HMMs but emphasize more on the relationships between the different variables. Within the sphere of autonomous driving, [Lefèvre et al., 2011] modelized dependencies between several variables (chosen maneuver, turn signal...) as a graph an leveraged Bayesian networks to infer the probability of each possible lane.

2.3.4 Monte Carlo simulation

Monte Carlo sampling can be used to simulate future path from a sequence of uniformly sampled controls while checking collisions to infer paths avoinding colisions [Broadhurst et al., 2005, Eidehall and Petersson, 2006]. Such approach does not require linear physical models.

2.4 Hierarchicl planning models

To consider the different scales related to a realistic motion planning, the process can be subdivided in successive intermediary steps.

2.4.1 High level planning

Some work focus on high level planning. This tasks aims at modeling more general concepts than a trajectory, such as classifying driving intentions such as lane change, turn left or following a leading vehicle. [Wen and Gong, 2023] leveraged an MCTS for behavior planning, [Greene et al., 2011] used a Kalman filter to select the most likely intended maneuver, and [Streubel and Hoffmann, 2014] utilized a HMM to estimate the likelihood of a vehicle going straight or turning.

2.4.2 Graph search

Some approaches aim at planning on road map modelized as a graph via graph search. In particular, A* [Hart et al., 1968] is based on Dijkstra algorithm and is designed to find the most efficient path from a start node to a goal node in a graph while considering the cost of each edge or step. A* uses a heuristic function to estimate the cost of reaching the goal from the current node. The heuristic helps guide the search towards the goal, making A* more efficient than Dijkstra's algorithm in many cases. This approach was extended to continuous state search by storing continuous states in reached discrete cells with Hybrid A* [Dolgov et al., 2008].

Chapter 3

Literature review: from deep learning models to learning-based motion planning

Contents

Machine learning has significantly advanced the field of autonomous driving, offering innovative solutions and methodologies for creating intelligent, self-driving vehicles. Two prominent machine-learning approaches applied in this domain are imitation learning and reinforcement learning, each contributing uniquely to the development of autonomous driving systems.

3.1 Deep learning model components

This section introduces deep learning model components which are relevant for our study. Those components are the different blocks composing most of the deep learning models introduced in this thesis.

3.1.1 Multi layer perceptron

A Feed-Forward Network with Fully Connected Layers, also called Multi-Layer Perceptron (MLP) and represented in Figure 3.1 is a type of artificial neural network designed for supervised learning [Bishop, 1995]. It consists of three main layers: an input layer, one or more hidden layers, and an output layer. Each layer is composed of nodes, also known as neurons. In a MLP, information flows through the network in a forward direction, with input data passing through the hidden layers to produce an output. Neurons in each layer are connected to neurons in the subsequent layer, and each connection is associated with a weight that is adjusted during the training process to optimize the network's performance. MLPs (Multi-Layer Perceptrons) employ activation functions to introduce non-linearity to the model, enabling it to capture complex patterns and relationships in the data. To enable affine transformations of the data, *i.e.* adjusting the output independently of the inputs, we must include a set of parameters known as biases. These biases are added to the weighted sum of inputs to shift the activation function. Training an MLP involves adjusting the weights through a process called backpropagation, where the model learns by comparing its predictions to the actual target values and updating the weights accordingly. MLPs are widely used in various applications, including image recognition, natural language processing, and financial forecasting.

3.1.2 Convolutional neural network

A Convolutional Neural Network (CNN) is a type of artificial neural network designed for processing structured grid data, such as images. CNNs are particularly well-suited for tasks like image recognition or other computer vision related tasks [LeCun et al., 2015]. The key innovation of CNNs is the use of convolutional layers, which apply filters or kernels to small, overlapping regions of the input data. These filters help capture local patterns and features in the data, allowing the network to recognize hierarchical representations of visual information. CNNs typically consist of multiple convolutional layers followed by pooling layers to reduce spatial dimensions and alleviate computational intensity. The final layers are usually fully connected layers that make

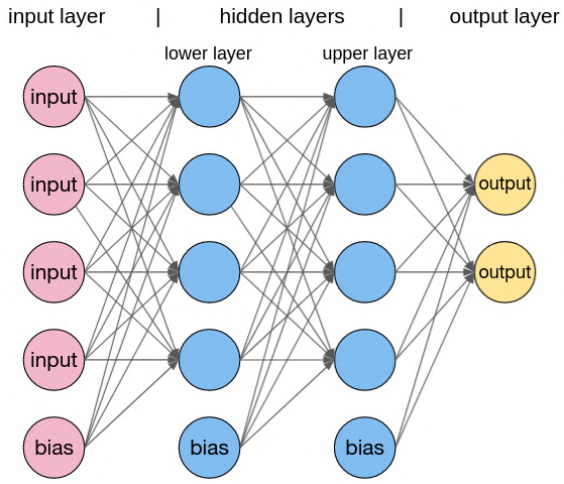


Figure 3.1: Representation of a MLP. It is composed of multiple layers of nodes (neurons), including an input layer, one or more hidden layers, and an output layer, with each node connected to all nodes in the next layer and having associated weights and biases. Figure from medium.com.

predictions based on the learned features. The architecture of CNNs is inspired by the organization of the visual cortex in animals. LeCun et al. [LeCun et al., 1995] describe the effectiveness of CNNs in capturing spatial hierarchies of features, making them well-suited for image-related tasks. A CNN classical architecture is represented in Figure 3.2.

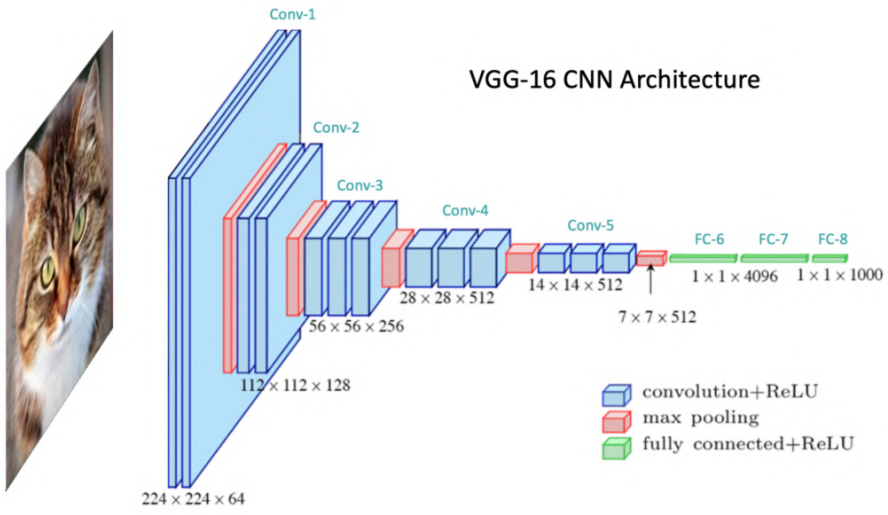


Figure 3.2: Representation of a CNN. It comprises layers of convolutional filters, pooling, and fully connected layers to process and classify image data effectively. Figure from learnopencv.com.

3.1.3 Attention and transformers

Attention is a method that allows a model to weight different input positions differently when making predictions, enabling it to consider contextual information effectively. Attention operates akin to a dictionary search among tokens, a fundamental unit of data such as a word or number, with each token identifying and utilizing only the relevant information from other tokens. This process involves creating three tensors from the tokens through linear projection: the queries Q, keys K, and values V. In time encoding scenarios, all these tensors are derived from time steps T and thus have a shape of (T, d), where d represents the feature size. When queries, keys, and values all originate from the same tensor, it's known as self-attention. Attention update the tokens via a weighted by X sum. X is computed following Equation 3.1.

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

The Transformer is a type of neural network architecture based on Attention that has gained prominence for its success in natural language processing tasks. Introduced by Vaswani et al. [Vaswani et al., 2017a], the Transformer architecture eschews traditional recurrent or convolutional layers and relies on a mechanism called attention. The Transformer consists of an encoder and a decoder, each containing multiple layers of self-attention and feedforward neural networks. The self-attention mechanism enables capturing long-range dependencies in the input data, making Transformers highly effective for tasks such as machine translation and language understanding. The use of self-attention also facilitates parallelization of training, enhancing efficiency compared to sequential models. The Transformer architecture has since been adapted and extended for various applications beyond natural language processing [Devlin et al., 2018], including image recognition [Dosovitskiy et al., 2020], data series [Wen et al., 2022a] and generative tasks [Hudson and Zitnick, 2021].

3.1.4 Recurrent neural network

A Recurrent Neural Network (RNN) is a class of artificial neural networks designed for processing sequential data and capturing temporal dependencies. In contrast to traditional feedforward neural networks, RNNs have connections that form directed cycles, allowing them to maintain a hidden state that evolves over time. This hidden state serves as a memory that retains information about previous inputs, enabling RNNs to consider context and sequential information in their predictions. However, traditional RNNs often face challenges in learning long-term dependencies due to issues like vanishing or exploding gradients. Hochreiter et al. [Hochreiter and Schmidhuber, 1997a] introduced the Long Short-Term Memory (LSTM) architecture as a solution to these problems, incorporating memory cells with gated mechanisms to selectively store and retrieve information. LSTMs have become a popular choice in RNN architectures for tasks such as natural language processing, speech recognition, and time-series analysis, where modeling temporal dependencies is crucial. An LSTM cell is represented in Figure 3.4. Another type of RNN is the Gated Recurrent Unit (GRU) [Cho et al., 2014], which is similar to LSTM (Long Short-Term Memory), but with a

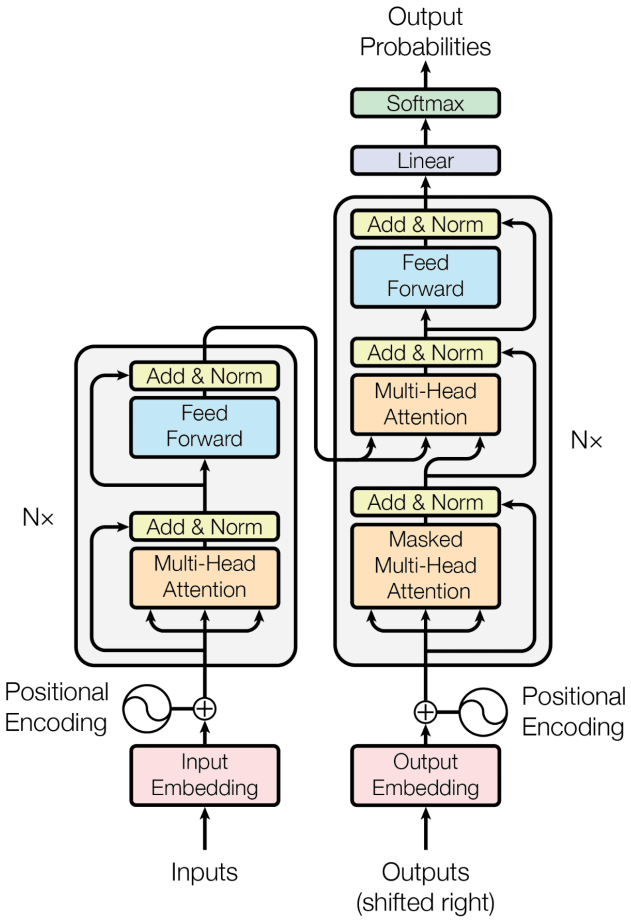


Figure 3.3: Representation of a Transformer. It is made of an encoder and a decoder, each consisting of layers of self-attention mechanisms and feed-forward neural networks. Figure from machinelearningmastery.com.

simpler structure and fewer parameters, making it computationally more efficient.

3.2 Imitation learning for motion planning

Imitation Learning, a subset of supervised learning, has been pivotal in teaching autonomous vehicles to mimic human driving behavior. This approach involves training machine learning models on datasets of human driving behavior to learn the complex patterns of navigating roads, responding to traffic signals, and avoiding obstacles. A key advantage of imitation learning is its ability to leverage real-world driving data, providing a more practical and realistic learning environment for the autonomous systems while overcoming the inherent rigidity of rule-base methods.

3.2.1 Imitation learning for end-to-end autonomous driving

Imitation Learning (IL) is by design simpler to setup than rule-based methods, as it removes the requirements of mathematical modelization. Simply training a classical neural network end-to-

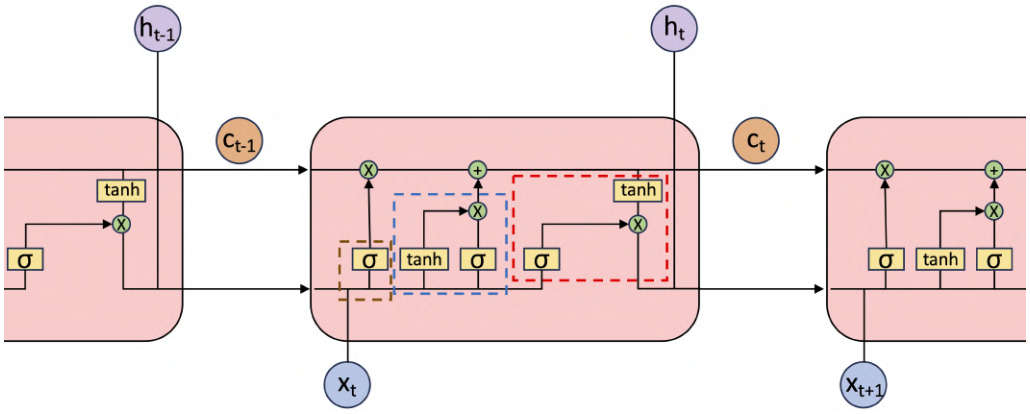


Figure 3.4: Representation of an LSTM cell. The Cell State (C_t, in orange) runs through the entire sequence. It stores and transmits information across time steps while selectively modifying or forgetting parts of it. The Hidden State (h_t, in purple) is the output of the LSTM cell at a specific time step. It carries information that is relevant to the current time step's prediction or output. It is also influenced by the cell state and the input at that time step. LSTMs employ three gate types (forget in brown, input in blue, and output in red) to regulate how information is managed within the cell state and the hidden state. Figure from [Chekroun et al., 2024].

end on expert data can lead to interesting results. In 1989, Pomerleau et al. [Pomerleau, 1988] pioneered the field with ALVINN, a method leveraging a shallow fully connected neural network for end-to-end prediction of steering wheel angle from camera images and radars signals, able to follow lanes on public roads. Obstacle avoidance was first achieved in 2005 by [Lecun et al., 2004] with DAVE, a robot car navigating in a cluttered backyard with two cameras to extract depth information. First notable real-life applications on commercial vehicles were reached in 2016 by Bojarki et al. [Bojarski et al., 2016] who introduced a convolutional neural network (CNN) trained to steer a car in a range of driving conditions (highways and residential roads) by mimicking a human driver.

Since then, the creation of open-source simulators allowed an acceleration of this domain of research by providing safe training and testing environment, and easy access to diverse scenes and sensor data. In this context, Chitta et al. developed TransFuser [Chitta et al., 2022], an end-to-end autonomous driving pipeline leveraging attention for LiDAR and RGB camera fusion. It leverages a MLP and a GRU on multi-modal vision features to generate an ego-centric trajectory in the bird-eye-view domain for the ego. To improve the training process, TranFuser leverages auxiliary losses during training: the vision part of the system is trained in the same time on generating features to be used by the controller part, and features yieldings good performances when associated with decoders for given vision tasks (semantic segmentation, depth estimation, other vehicle detection, and road reconstruction). TransFuser full training architecture is represented in Figure 3.6.

However, an imitation learning model is trained to mimic an expert's handling of traffic scenarios that the expert themselves has caused. Therefore, when the model is in control of the car, its decisions directly affect what the car will encounter next. As a result, the model must adapt to the outcomes of its own driving. This can be problematic if the model's driving choices result in

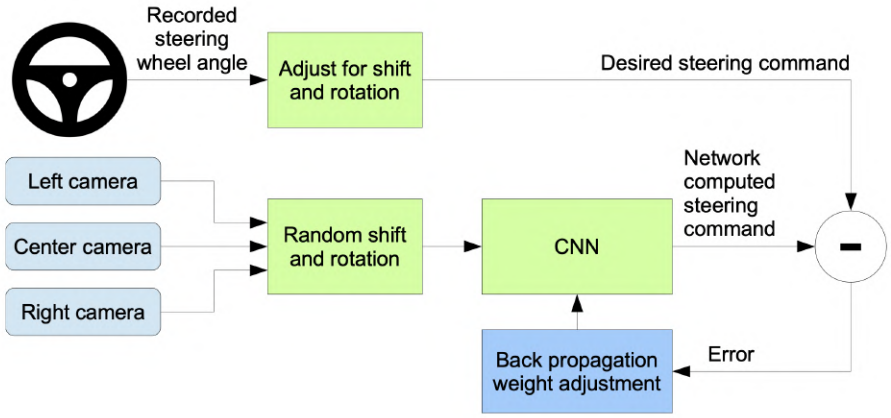


Figure 3.5: Training loop of Bojarski et al. end-to-end imitation learning autonomous driving system. Images are fed into a CNN which computes a proposed steering command. The proposed command is compared to the desired command for that image and the weights of the CNN are updated using back propagation. Figure from [Bojarski et al., 2016].

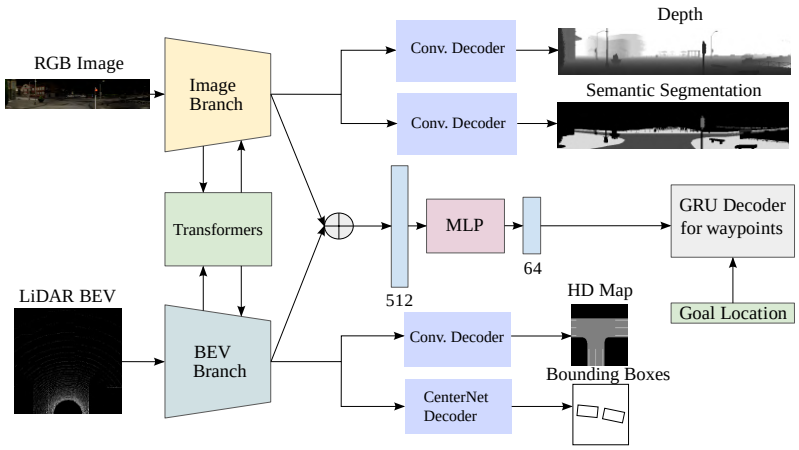


Figure 3.6: TransFuser is made of a multi-modal vision subsystem composed of inter-communicating LiDAR module, RGB images modules and Transformers modules to efficiently fusion RGB image and LiDAR inputs. It outputs the future trajectory of the ego, in ego-centric bird-eye-view space. TransFuser is trained on several auxiliary tasks demonstrated to improve final metrics on the driving. Figure from [Chitta et al., 2022].

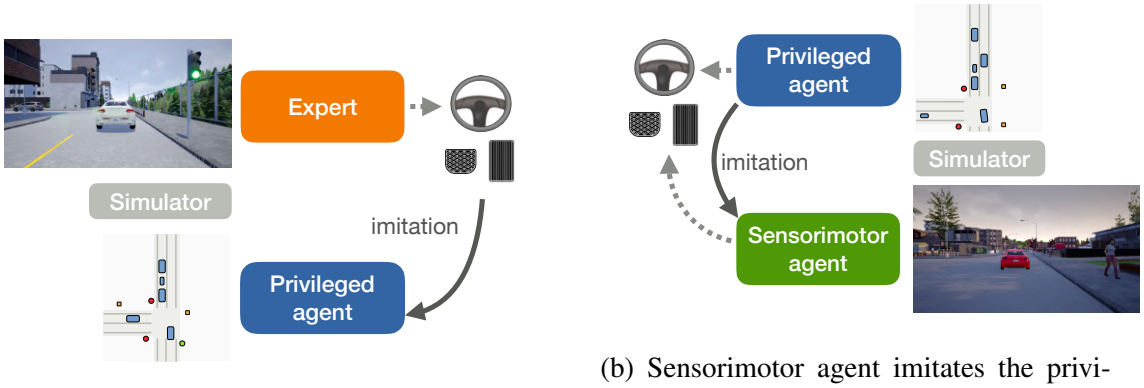
unfamiliar situations, leaving the model unsure of the appropriate action to take. The issue where a self-driving car, guided by the model, encounters situations during actual driving that were not covered during training is known as the *distribution shift* problem. This occurs because the real-world experiences differ from the scenarios presented during training. For instance, if the training only involved the expert driving mainly in the center of the road, the model would not have learned how to correct the car's course when it strays towards the roadside.

3.2.2 Solutions to the distribution shift

Overcoming the distribution shift is a first step toward safer IL models for autonomous driving. To make models more general and more robust to this issue, data need to be more diverse, either by collecting more data or by generating additional samples.

A simple hardware approach to tackle the distribution shift is to add left and right cameras on the vehicle and associate them with opposite steering. This allows to provide example of desired reaction when the vehicle drifts away from the center of lane [Müller et al., 2018, Bojarski et al., 2016].

A method less reliant on the need of more hardware components is on-policy learning. It is a training method where data is collected by an agent alternating between the expert policy and the model's one. DAgger [Ross et al., 2010] first introduced this approach which proved to teach the model to be able to recover from errors and handle situations the expert would never have been exposed to. In DAgger, the expert provides expert data solving situations reached by the model. To reduce human involvement, SafeDAgger [Zhang and Cho, 2016] developed a module deciding whether expert's help is required at an given moment. Other methods partially or completely remove the need of a human expert by applying conventional controller stack when required to annotate encountered states [Pan et al., 2017, Li et al., 2018]. Chen et al. [Chen et al., 2020a] went further in *Learning by Cheating* (LBC) and replaced the controller stack by a model called the *privileged agent* who have access to ground truth information, and trained to replace the expert for the training of a *sensorimotor agent* with access to regular RGBs camera data only as input, as represented in Figure 3.7.



(a) Privileged agent imitates the expert

(b) Sensorimotor agent imitates the privileged agent

Figure 3.7: Overview of LBC. **(a)** The privileged agent with access to ground truth information learns to imitate expert demonstrations. **(b)** A sensorimotor agent with access to sensory data onlylearns to imitate the privileged agent. The privileged agent provides high-capacity on-policy supervision. Figure from [Chen et al., 2020a].

Without resorting to on-policy learning, distribution shift can be reduced by generating artificial data via data augmentation [Shorten and Khoshgoftaar, 2019]. In the realm of autonomous driving, flipping, rotating, scaling or cropping RGB images help create variations of existing data and have shown to improve the impact of accumulation error on distribution shift in lane keeping. Other data augmentations like blurring, erasing, noising, brightness or hue changing to RGB im-

ages are standard data augmentation [Sobh et al., 2018, Carton, 2021] helping to adapt on unseen scenes with different weather or lighting conditions.

Despite their efficiency and generalizability on validation sets and some real-life applications, IL methods frequently operate as well trained black boxes, thus lacking explainability. This results in behavior that cannot be proven and offers no theoretical guarantees of preventing safety-critical issues.

3.2.3 Toward more explainability and robustness

To address these issues, other approaches focus on making the planning decisions more interpretable and robust.

Some methods aims at improving the robustness by generating multiple planning options with deep learning models, evaluate all of them with a cost function assessing some driving features (no collision, stay within the drivable area...), and finally select the one minimizing it [Cui et al., 2021, Sadat et al., 2020, Sadat et al., 2019, Fan et al., 2018, Zeng et al., 2020]. This cost function can be partially learned [Cui et al., 2021, Sadat et al., 2020] or handcrafted [Sadat et al., 2019, Fan et al., 2018]. This kind of method is represented in Figure 3.8. Other methods, such as GameFormer Planner [Huang et al., 2023], generate a single trajectory via deep learning models and then refine it via mathematical optimization [Huang et al., 2023, Aydemir et al., 2023], therefore improving overall robustness.

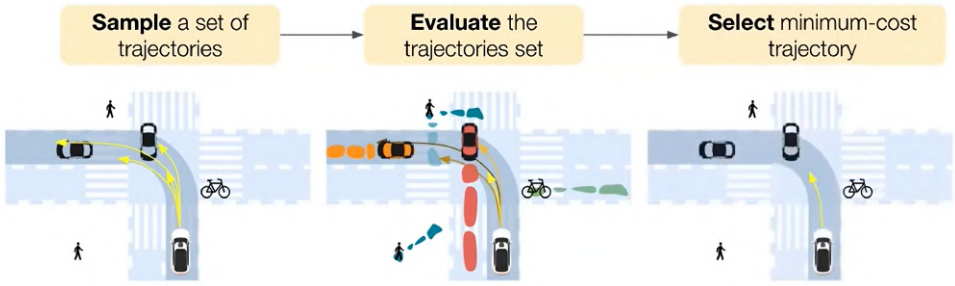


Figure 3.8: Pipeline of trajectory proposals methods for autonomous driving. Several trajectories are generated with a deep learning model. This trajectories are then attributed a score assessing the driving quality related to it. The trajectory yielding the best score is selected. Figure from [Sadat et al., 2020] ECCV presentation.

Tackling the explainability limitation, Dauner et al. developed Predictive Driver Model (PDM) [Dauner et al., 2023] to combine an interpretable IDM with a simple fully connected neural network.

3.3 Reinforcement learning in motion planning

Instead of copying human behavior like IL does, Reinforcement Learning (RL) models leverage a reward system to assess how good a strategy is, and learns by trial and error. This can lead

to improved decision-making, sometimes even outperforming humans [Silver et al., 2016, Silver et al., 2017, Schrittwieser et al., 2020].

3.3.1 Reinforcement learning for autonomous driving

In 2019, [Kendall et al., 2017] presented a RL method allowing a real car to follow lanes in an empty street. In 2020, [Amini et al., 2020] trained a RL agent for lateral control on the VISTA data driven simulator. Since then, no RL methods have been shown to be competitive with IL on end-to-end autonomous driving. A cause might be the weakness of the RL reward signal which is not enough to train the convolutional part of the network. However RL has been successfully applied in autonomous driving to fine-tune models pre-trained with IL [Liang et al., 2018a, Ohn-Bar et al., 2020], or when combined with supervised learning [Toromanoff et al., 2020a, Chekroun et al., 2021]. Implicit Affordances (IAs) method [Toromanoff et al., 2020a] presented an approach where a shallow model-free DRL model was trained on the latent space of a frozen ResNet pre-trained on several auxiliary tasks, such as semantic segmentation and classification. This method reached state-of-the-art performance on the CARLA simulation at its release and its pipeline is represented in Figure 3.9.

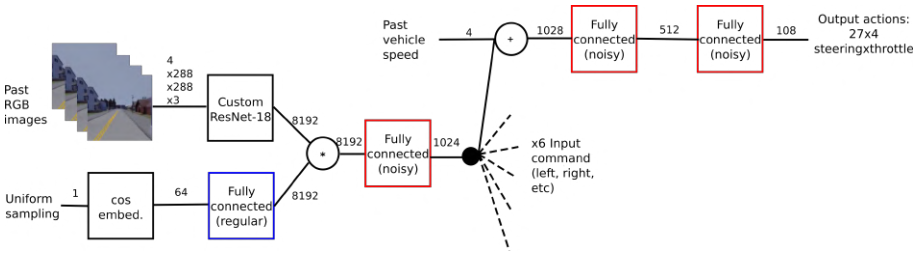
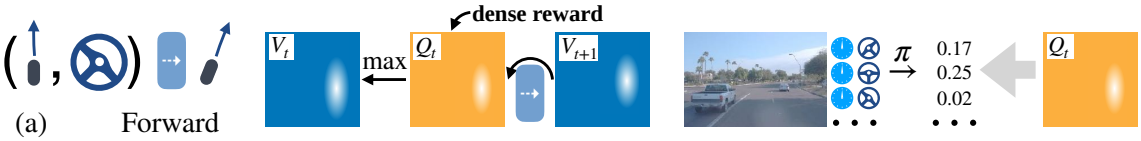


Figure 3.9: IAs network architecture. A ResNet is pre-trained, with decoders and associated auxiliary tasks, to encode RGB images from the onboarded camera and frozen. Fully Connected Networks (FNCs) are then trained with an RL training on the ResNet output. Figure from [Toromanoff et al., 2020a].

3.3.2 Reinforcement learning for dataset curation

Other methods effectively leveraged RL to planning or control tasks where the network has access to privileged simulator information [Knox et al., 2023, Zhang et al., 2022, Zhang et al., 2021, Chen et al., 2020a] whether for driving directly or as a dataset curator. In particular, Zhang et al.'s ROACH method trained a RL model on privileged bird-eye-view semantic segmentation and exploits the obtained policy to automatically collect a dataset used to train an online imitation learning agent, thus tackling the distribution shift issue following intuitions presented in Section 3.2.2. Finally, Chen et al. World On Rails (WoR) [Chen et al., 2021a] employs traditional RL to generate additional or enrich labels for a static dataset used to train a visuomotor model, as represented in Figure 3.10.



(a) Forward model (b) Bellman update (c) Distillation

Figure 3.10: Overview of WoR method. (a) A forward model is learned from a dataset of offline driving trajectories of sensor readings, driving states, and actions. (b) With the offline driving trajectories, action-values under a predefined reward and learned forward model using dynamic programming and backward induction on the Bellman equation are then computed. (c) Action-values are leveraged to train a reactive visuomotor driving policy through policy distillation. Figure from [Chen et al., 2021a].

3.4 Conclusion

RL and IL both have their respective strengths and weaknesses, and those are complementary. Indeed, IL suffers from distribution mismatch contrarily to RL. Alternatively, as RL learns from scratch, it is less data efficient than IL, which incorporates prior Expert Knwoledge during training. In the next Chapter 4, we will introduce General Reinforced Imitation (GRI), a new paradigm leveraging both Expert Knowledge and exploration to enhance the RL vanilla approach by distilling expertise in model-free RL.

Chapter 4

Expertise distillation in model-free RL for end-to-end autonomous driving

Contents

4.1 Basics of model-free RL

The RL problem can mathematically be defined as an Markov Decision Process (MDP). An MDP is a 5-tuple (S, A, P, r, γ) with:

- S is the state space (generally finite).
- A is the action space (discrete or continuous).
- P is the probability of the transition from one state to another. We note $p(s_2|s_1, a_1)$ the probability to reach state s_2 when taking the action a_1 in state s_1.
- r is the reward function. We note $r(s, a) \in \mathbb{R}$ the reward obtained by doing the action a in the state s.
- $\gamma \in [0, 1]$ is the discount factor. It represents the extent to which future rewards are considered in decision-making, with higher values giving more weight to long-term rewards.

A trajectory τ is a sequence $(s_0, a_0), (s_1, a_1), ..., (s_N, a_N)$. The goal of an agent is to maximize the sum of accumulated reward $R_t = \sum_{k=0}^{T} \gamma^k r_{t+k}$ obtained throughout the associated trajectory.

A policy $\pi : S \rightarrow A$ is a function that associates to each state a distribution of probability over the action space. The policy is said to be deterministic if there is just a single action associated to each state. We call $\pi(a_t|s_t)$ the probability to take action a_t in the state s_t.

We define the state-value function V_π of a policy π as an expectation of the accumulated reward at this state following π. This can be interpreted as a quantifier of how *good* a given state is. This value function depends of the policy and is mathematically defined as follow:

$$V_\pi(s) = \mathbb{E}\left[R_t|s_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k}|s_t = s\right] \tag{4.1}$$

with $\mathbb{E}_\pi$ the expected value over the choice of trajectories with the policy π.

We also define the action-value function Q_π as the value obtained when starting a trajectory from a state s with an action a and then following the policy π until it reaches a terminal state. This Q-function is mathematically define as follows:

$$Q_\pi(s) = \mathbb{E}\left[R_t|s_t = s, a_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k}|s_t = s, a_t = a\right] \tag{4.2}$$

An important propriety of the state-value and action-value functions is that they can be defined iteratively with the Bellman equation:

$$Q(s, a) = \sum_{s'} P(s'|s, a)\left(r(s, a, s') + \gamma \cdot \max_{a'} Q(s', a')\right) \tag{4.3}$$

These three notions of policy, state-value function, and action-value function are at the core of most RL algorithms. In essence, solving a RL problem consists in finding a policy that maximizes long-term rewards. A policy π_1 is defined as being greater or equal than another policy π_2 if

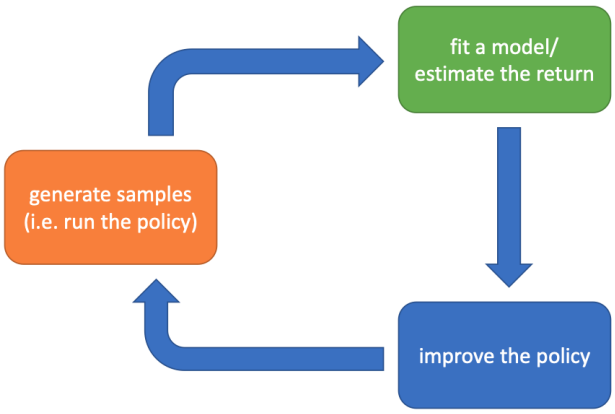


Figure 4.1: General architecture of a reinforcement learning algorithm. Figure from S. Levine CS294 Deep Reinforcement Learning course, Fall 2017, UC Berkeley.

the expected return is greater for each state s when following π_1 compared to π_2, *i.e.* $\pi_1 \geq \pi_2$ if $\forall s \in S, V_{\pi_1}(s) \geq V_{\pi_2}(s)$. This defines a partial order on the set of policies. Furthermore, there always exist at least one deterministic policy greater than all the other. It is refered to the optimal policy, noted π^*. Consequently, every RL algorithms aims at finding this policy, and they follow the same global architecture represented in Figure 4.1.

4.2 Deep Q Network

4.2.1 Q-learning

Q-learning is a model-free reinforcement learning algorithm which operates in discrete state and action spaces and iteratively updates its estimates of action values, denoted as Q-values, based on observed experiences. The core idea is to approximate the optimal action-value function for an agent in a Markov decision process (MDP). The main equation in Q-learning is the Q-learning update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left(r + \gamma \cdot \max_a Q(s',a) - Q(s,a) \right) \tag{4.4}$$

During training, the agent explores the environment, receives rewards, and updates its Q-values according to the Q-learning update rule 4.4. Over time, these Q-values converge to their optimal values, enabling the agent to make informed decisions by selecting actions with the highest estimated Q-values. Q-learning is widely applied in various domains, including robotics, gaming, and autonomous systems, due to its simplicity and effectiveness in solving reinforcement learning problems. State space and action space being discrete, the Q-learning algorithm stores information in 2D tables where (s,a) coordinates contains the information $Q(s,a)$.

4.2.2 Deep Q Network

Real-life applications often present a very large state space. Therefore, Q-learning tabular representation cannot be considered, as stored tables would be too large. In this context, Mnih et al. [Mnih et al., 2015] developed the Deep Q Network (DQN), an extension of Q-learning leveraging a neural network (NN) trained to approximate the optimal Q-function via learned parameters θ such that $Q(s,a,\theta) \approx Q^*(s,a)$. This NN is optimized via backpropagation with the following loss function, which incorporate the update equation of Q-learning:

$$L\left(s_t, a_t, r_{t+1}, s_{t+1}, \theta\right) = \underbrace{\left(r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a, \theta\right)}_{\text{target}} - Q\left(s_t, a_t, \theta\right)\right)^2 \tag{4.5}$$

We observe the first part of the equation (the target) to be dependant from θ. Hence, this method is not a gradient descent, but a *semi-gradient* one. This causes instability and strong oscillations. These issues are tackled by Minh et al. by leveraging a *target network* parametrized with parameter θ' used to compute the target. In practice, the target network is a copy of the first neural network which is updated in very large intervals of time. Minh et al. also clipped the loss to avoid high loss samples harming previous training. Finally, as data are gathered sequentially during the simulation, experiments used for the backpropagation are highly correlated. To avoid negative consequences of this correlation, all the explored transitions $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}$ are stored in a *replay buffer*. Q-learning being an off-policy algorithm *i.e.* we can leverage samples from another policy, or an older version of the neural network, to update parameters, the issue can be mitigated by backpropagating on approximately decorelated transitions randomly sampled from the memory buffer.

Therefore, DQN theoretically is a powerful tool for model-free reinforcement learning, which can be leveraged to learn a policy by trial and error via interactions with a simulator. However, even with all the mentioned tricks, DQN still suffers from its limitations which are sample inefficiency, instability, and overestimation bias. It struggles with the exploration-exploitation trade-off, especially in high-dimensional state spaces, and is sensitive to hyperparameter choices. Memory requirements for experience replay can be high, generalization across tasks has been shown to be limited, and safety guarantees are lacking. To overcome these limitations, some research focused on combining incremental improvements on DQN [Hessel et al., 2018] or building distributed DQN setup to mitigate sample efficiency [Horgan et al., 2018]. Another approach consists of leveraging expert data into DQN training to reduce instability and improve sample efficiency [Reddy et al., 2019b, Hester et al., 2018].

4.3 General reinforced imitation

This sections introduces General Reinforced Imitation (GRI) [Chekroun et al., 2021], a method designed for distilling expert knowledge from demonstration data in a model-free off-policy reinforcement algorithm. GRI aims at overcoming IL distribution mismatch and RL data inefficiency

by combining exploration and demonstration through distillation of Expert Knowledge in a classical online RL training. GRI was applied to camera-based Autonomous Driving on the CARLA simulator with the GRIAD algorithm. It ranked first on the CARLA leaderboard for ~ 6 months and won the CARLA Challenge 2021.

4.3.1 Related work

RL and IL strengths and weaknesses are complementary. Indeed, IL suffers from distribution mismatch contrarily to online RL. Alternatively, as RL learns from scratch, it is less data efficient than IL, which incorporates prior demonstration knowledge during training. To take the best of both worlds, some algorithms combine IL's supervision and RL's trial-and-error to maximize efficacy through the leverage of both expert data and exploration, aiming at a generalizable and data efficient system [Hester et al., 2018, Reddy et al., 2019b, Rajeswaran et al., 2017, Martin et al., 2021].

In particular, demonstrations can be used to initialize policies by pretraining the network [Xu et al., 2018, Rajeswaran et al., 2017, Hester et al., 2018]. DQfD [Hester et al., 2018] is based on DQN, introduced in Section 4.2, an off-policy RL algorithm with a replay buffer. DQfD first pretrains the agent on expert data with both IL and RL losses using the real reward given by the environment. After some steps of pretraining, the agent starts gathering data from the environment in the memory buffer. The network is then trained on batches composed of exploration data with an RL loss and expert data with both IL and RL losses. Nonetheless, DQfD uses simultaneously reinforcement and imitation, which can have divergent losses and are difficult to jointly optimize [Gao et al., 2018]. Other approaches differentiate expert and exploration data via specific reward but using RL losses only [Hester et al., 2018, Reddy et al., 2019b]. Soft-Q Imitation Learning (SQIL) [Reddy et al., 2019b] completes the imitation task with an RL agent. To do so, the replay buffer is initially filled with demonstrations, associated with a constant reward $r_{demo} = 1$. An RL agent collects data from exploration into the replay buffer, associated with a constant reward $r_{explo} = 0$. Thus, SQIL designed an RL agent that learns to imitate expert behavior, and has been mathematically demonstrated to be equivalent to regularized behavior cloning. However, SQIL does not efficiently leverage exploration as environment rewards are never used.

The method we present in this section leverages both demonstrations and exploration exclusively with an RL loss, and thus cannot suffer from the divergent losses issue. Moreover, contrarily to DQfD, GRI does not require the true environment rewards for the expert data, which cannot always be obtained, thus making it an easier solution for real-life applications.

4.3.2 Methodology

GRI is a method which is straightforward to implement over any model-free off-policy RL algorithm using a replay buffer, such as SAC [Haarnoja et al., 2018], DDPG [Lillicrap et al., 2016], DQN [Mnih et al., 2015], and its successive improvements [Hessel et al., 2017, Dabney et al., 2018]. GRI is built upon the hypothesis that expert demonstrations can be seen as perfect data

whose underlying policy gets a constant high reward. We denote this as demonstration reward, r_{demo}. In our experiments, we chose r_{demo} to be the maximum of the reachable reward by an agent. GRI introduces the notion of offline *demonstration agents* which sends expert data associated with the reward r_{demo} to the memory buffer. These agents collect transitions from an expert dataset and work concurrently and indistinguishably with exploration agents connected with the simulator to collect states, actions, and rewards. GRI algorithm is presented in Algo. 1.

Algorithm 1: GRI: General Reinforced Imitation.

```
Input: r_demo demonstration reward value, p_demo probability to use demonstration
  agent;
Initialize empty buffer B;
while not converged do
    if len(B) ≥ min_buffer then
        do a DRL network update;
    end
    if random.random() ≥ p_demo then
        collect episode (s_t^online, a_t, r_t, s_{t+1}^online)_t in buffer B with exploration
          agent
    else
        add episode (s_t^offline, a_t, r_demo, s_{t+1}^offline)_t in buffer B with demonstration
          agent;
    end
end
```

The idea of GRI is to distill expert knowledge from demonstrations into an RL agent during the training phase. To do so, we defined two types of agents: (i) the online *exploration agent*, which is the regular RL agent exploring its environment to gather experiences $(s_t^{online}, a_t, r_t, s_{t+1}^{online})$ into the memory buffer, and (ii) the offline *demonstration agent*, which sends expert data associated with a constant demonstration reward $(s_t^{offline}, a_t, r_{demo}, s_{t+1}^{offline})$ to the memory buffer. s_t is the state, a_t the chosen action and r_t the reward at time t. At any given training step, the next episode to add to the replay buffer comes from the demonstration agent with a probability of p_{demo}, else from the exploration agent.

4.3.3 Ablation study on Mujoco

We first validate GRI on simple selected environments from the Mujoco benchmark, represented in Figure 4.2.

Expert data were generated using chainerrl [Fujita et al., 2021] pretrained RL agent weights and contain 200,000 samples. For each environment, the value of r_{demo} was chosen as the highest value chainerrl expert agent reached during the generation of the dataset. As we did not find real expert data on Mujoco environments, expert data are not always significantly better than our trained vanilla RL network. Hence, this study assesses the efficiency of GRI even with suboptimal expert data.

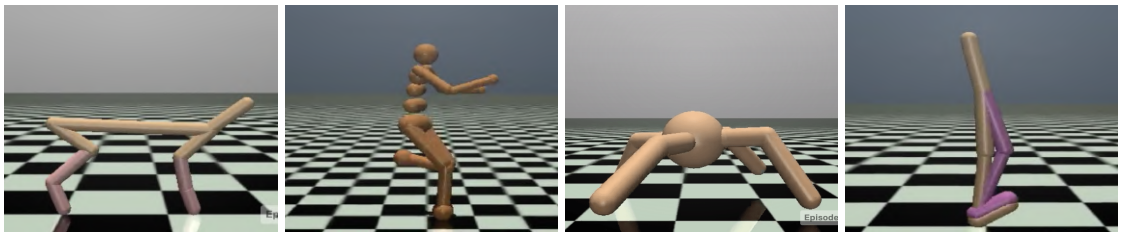


Figure 4.2: Mujoco environments used for our experiments. Respectively HalfCheetah-v2, Humanoid-v2, Ant-v2, and Walker2d-v2. Articulations are controlled to make them walk. Rewards depend on the covered distance.

To validate GRI and further comprehend the impact of demonstration agents we compare RL training with different proportions of demonstration agents and with exploration agents only. For these experiments, we used a GRI-SAC, i.e., a GRI algorithm using SAC [Haarnoja et al., 2018] as DRL backbone, and we vary the proportion of demonstration agents between 0% *i.e.* vanilla RL and 40%. Each experiment has been repeated three times, with different seeds. Figure 4.3 presents the results with the variances and the evaluation reward of the expert. Experiments were conducted with public code from GitHub (original code from `https://github.com/dongminlee94/deep_rl`, accessed on 3 November 2021), which has been adapted with GRI.

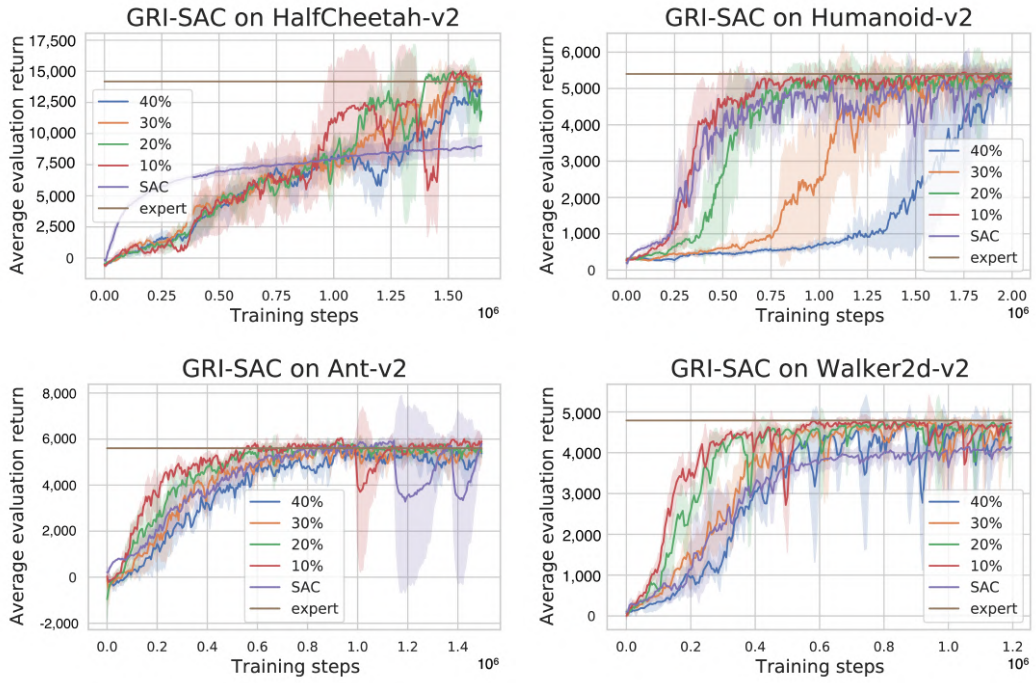


Figure 4.3: Ablation over demonstration agents with the GRI-SAC setup on Mujoco environments, and analysis of the evolution of the evaluation reward in function of the proportion of demonstration agents. GRI-SAC with 0% demonstration agent is vanilla SAC. We observe that GRI-SAC always reaches the level of the expert even when the expert is significantly better than the trained vanilla SAC. The proportion of demonstration agent has a significant impact on the dynamics of the convergence.

We observe three different dynamics.

- For HalfCheetah-v2, a difficult task on which the expert is significantly stronger than the trained SAC, we observe that the beginning of the training is slower using GRI-SAC; we call this a warm up phase, which we will explain further in Section 4.6. However, the rewards turns out to become significantly higher after some time. Here, GRI-SAC is better than SAC with every proportion of demonstration agents. The best scores were reached with 10% and 20% of demonstration agents.
- For Humanoid-v2, a difficult task on which the expert is just a little stronger than the trained SAC, we observe that the higher the number of demonstration agents is, the longer the warm up phase is. Nonetheless, GRI-SAC models end up having higher rewards after their warm up phase. The best scores are reached with 10% and 20% of demonstration agents.
- Ant-v2 and Walker2d-v2 are the easiest tasks of the four evaluated. On Ant-v2, the SAC agent reaches the expert level, converging similarly as GRI-SAC regardless of the number of demonstration agents used. Nevertheless, GRI-SAC converges faster with 10% and 20% demonstration agents. On Walker2d-v2, the final reward of GRI-SAC is significantly higher and reaches the expert level, while SAC remains below.

More experiments were conducted, with the proportion of demonstration agents varying between 50% and 90%. Results were significantly worse than using 20% demonstration agents. Therefore, we conclude that the proportion of demonstration agent should not exceed 50%. We discuss some qualitative insights in Section 4.6.

These experiments reveal, at least on the evaluated Mujoco environments, that 20% demonstration agents seems to be the best choice for GRI-SAC to reach the expert level.

We also investigated the contribution of the DRL backbone to assess the generalizability of the GRI method. To do so, we evaluated the same tasks with the Deep Deterministic Policy Gradient (DDPG) algorithm [Lillicrap et al., 2016] instead of SAC. For these experiments, we fixed the proportion of demonstration agents to 20%. Results are shown in Figure 4.4.

We observe that, similar to GRI-SAC with a proportion of 20% demonstration agents, GRI-DDPG reaches better results than DDPG on all the tested environments. However, GRI-DDPG does not systematically reach the level of the expert. While final rewards are better with SAC and GRI-SAC, the dynamics of the rewards evolution is about the same with both backbones (compare Figure 4.3). We can conclude that GRI is easily adaptable and generalizes to locomotion tasks, where it robustly outperforms the two alternative methods.

4.4 GRI for autonomous driving

GRI being demonstrated as more sample efficient, providing stabler training and reaching better results than vanilla RL, we decided to apply it in an end-to-end autonomous driving pipeline. We chose to work on the CARLA simulator, as it offers strong baseline, a clear API and the possibility to compare our results to other state-of-the-art method. We named our method General Reinforced Imitation for Autonomous Driving (GRIAD).

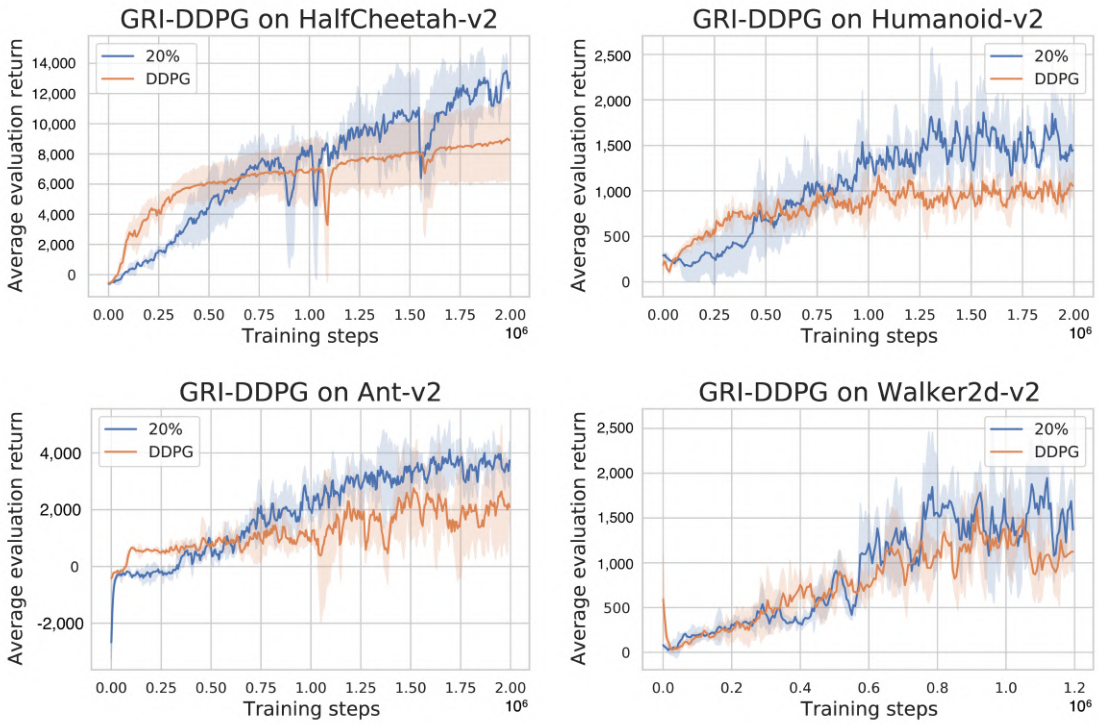


Figure 4.4: Ablation over demonstration agents with the GRI-DDPG, with 20% of demonstration agents on Mujoco environments. GRI-DDPG systematically leads to a better reward than vanilla DDPG. However, contrary to GRI-SAC, GRI-DDPG with 20% demonstration agents does not systematically reach the expert level.

4.4.1 Related Work

This section introduces methods reaching state-of-the-art performance on the CARLA leaderboard. Learning by Cheating (LBC) trains an agent having access to privileged ground truth information via imitation learning and distil its policy into a regular agent having only access to sensor informations. Implicit Affordances (IAs) method [Toromanoff et al., 2020a] develops a end-to-end pipeline where camera sensors are first encoded by a CNN before going through a model-free RL network. Transfuser [Prakash et al., 2021] is an IL method leveraging both LiDAR and cameras signals in a transformer-based sensor fusion system followed by a GRU-based waypoints prediction network and a PID controller. It went through several improvements in the data collection process and training method to give Transfuser+, reaching the highest score of this family of methods on the CARLA leaderboard. Latent Transfuser replaces LiDAR by simple positional encoding but significantly underperfoms his counterparts. World on Rails (WOR) [Chen et al., 2021b] distillate a vision based policy learned via model-based tabular RL approach in a CNN. Trajectory-guided Control Prediction (TCP) [Wu et al., 2022] aims at designing a long-term understanding IL-based control method. It predicts both control and trajectory planning via two interconnected neural network to provide the control branch with a guidance from the trajectory planning one, and finally fuse the two branches outputs. Learning from All Vehicles (LAV) [Chen and Krähenbühl, 2022a] leverage sensor fusion to build a bird-eye-view of the scene, and learns

driving policies from experiences collected not just from the ego-vehicle, but all vehicles around it. ReasonNet [Shao et al., 2023b] generates a bird-eye-view from sensor fusion and leverage a temporal reasoning module to process current and historic features. Then, a global reasoning module models the interaction and relationship among objects and the environment to detect adverse events (e.g. occlusion) and improve overall perception performance. Finally, InterFuser [Shao et al., 2023a] processes and fuses information from multi-modal multi-view sensors for achieving comprehensive scene understanding which is fed to a Transformer-based controller to generate driving actions.

4.4.2 The GRIAD pipeline

GRIAD is a camera-based end-to-end pipeline for autonomous driving. Therefore, it feeds camera data and outputs direct control of the ego vehicle. Its pipeline is made of two main components: a visual encoder, and a decision-making subsystem.

4.4.2.1 Visual Encoder

The designed visual subsystem is composed of two EfficientNet [Tan and Le, 2019] encoder models. They are trained with respectively one segmentation decoder for the first encoder, and several classifications and regressions of relevant driving information for the second one. The classified information are: the type of road ahead (straight road, intersection), the presence of traffic light and its color. The regressed information are the distance to the traffic light, and the vehicle rotation compared to the road centerline. The visual subsystem is represented in Figure 4.5 and an ablation study over the choice of encoder is presented in Table 4.1.

Task	Town, Weather	Visual Encoder	
		Toromanoff et al.	Chekroun et al.
Empty	train, train	85	**98.0 ± 1.0**
Regular		85	**98.6 ± 1.2**
Dense		63	**95.0 ± 1.6**
Empty	test, train	77	**96.3 ± 1.7**
Regular		66	**96.3 ± 2.5**
Dense		33	**78.0 ± 2.8**

Table 4.1: Ablation study of the visual encoder on the NoCrash benchmark. The designed visual encoder from Chekroun et al. [Chekroun et al., 2023b] appears to be more suited to be used to generate RL input features than the one from Toromanoff et al. [Toromanoff et al., 2020a]. Both encoders are evaluated with the same vanilla RL network.

EfficientNet encoders are trained jointly with the above-mentionned decoders. After training, encoders are frozen and decoders removed. The visual subsystem therefore feeds from RGB images and outputs encoded features, called Implicit Affordances (IAs) as in [Toromanoff et al., 2020a], which are then fed to the decision-making subsystem. The visual encoder has been trained

on a dataset of 400,000 samples, which corresponds to 44 hours worth of driving. This dataset has been generated with the CARLA autopilot on every town with random trajectories. Each sample of the dataset is composed of three images from the three cameras and the corresponding ground truth information, which are segmentation maps from CARLA, Booleans indicating the presence of an intersection, and the presence of a traffic light in front of the car. Furthermore, if there is a traffic light, a class corresponding to its color and the distance to it in meters. Trajectories have been augmented with random cameras translations and rotations.

4.4.2.2 Decision-making subsystem

The decision-making subsystem feeds four consecutive IAs and outputs ego vehicle next action. Therefore, a state contains visual features from the last 300 milliseconds, as the simulator runs at 10 FPS. An action is defined by the combination of the desired steering of the wheel, and the throttle or brake to apply. Generating data on the CARLA simulator is computationally expensive. We used a Rainbow-IQN Ape-X [Toromanoff et al., 2019b], which is a distributed DQN [Mnih et al., 2015] variante, to mitigate this issue. Therefore, we discretized the actions space in 27 steering values, and 4 acceleration (throttle or brake) values. The discretized action space contains $27 \times 4 = 108$ actions. The decision-making subsystem is represented in Figure 4.6.

Demonstration agents send to the memory buffer samples of expert trajectory from the demonstration dataset. This dataset consists of 22 h of driving, which correspond to 200,000 samples, generated using the autopilot from CARLA on predefined tracks published by CARLA. Each sample from the demonstration dataset consists of three images from the three cameras and a discrete

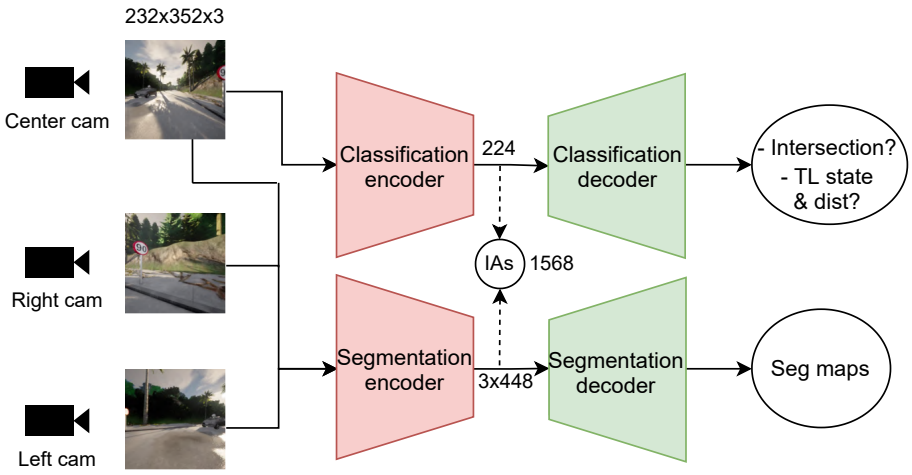


Figure 4.5: Feature extraction from RGB camera images for the visual subsystem. Two encoder-decoder networks are pretrained on segmentation, classifications, and regression tasks. Classifications and regression are only performed on the center image while all three images are segmented. After training, the visual encoders serve as fixed feature extractors with frozen weights. For the DRL backbone training, both encoder outputs are concatenated and sent to the memory buffer as input to DRL. Both encoders are Efficientnet-b1. The segmentation decoder is fully convolutional, and the classification decoder is an MLP with several outputs.

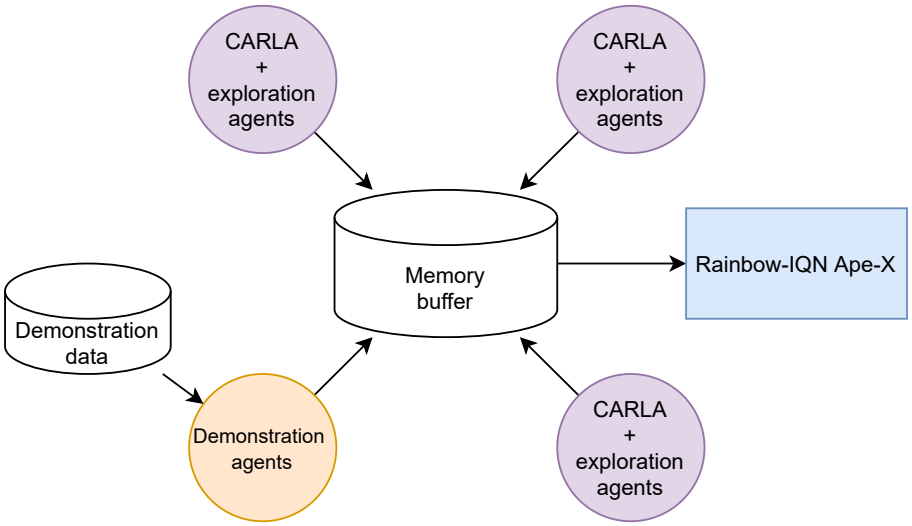


Figure 4.6: Simplified representation of the distributed GRIAD setup with a Rainbow-IQN Ape-X backbone. A central computer receives data in a shared replay buffer from both exploration and demonstration agents running on other computers. Data are sampled from this replay buffer to make the backpropagation and update the weights of all the agents. Images from the agents are encoded using the network presented in Figure 4.5 before being stored in the memory buffer.

action obtained by mapping continuous actions of the expert to our discrete set of RL actions. We did not use any data augmentation. We note that the autopilot makes driving errors, such as collisions, red light infractions, or the car getting stuck for hundreds of frames. As a result, ~10% of our demonstrations correspond to poor action choices. However, we decided to use this demonstration dataset as is in order to assess the robustness of our method to noisy demonstrations.

In our experiments on CARLA, GRIAD had a total of 12 agents, including 3 demonstration agents, running in a distributed setup and sending data to the memory buffer. As demonstration agents have been constrained to send data at the same frequency as exploration agents, this is equivalent to having $p_{demo} = 25\%$.

The training reward is primarily based on the waypoint API provided by the CARLA simulator. This API grants access to the dynamic positions and orientations of all lanes within the current environment, which is crucial for plotting the agent's path. It also details various options at intersections. An agent commences each episode at a random waypoint within the virtual city, from which the ideal route is calculated using the waypoint API. Upon approaching an intersection, the agent randomly selects a maneuver—left, straight, or right—and proceeds accordingly. The reward structure is based on three principal factors:

- Desired speed: this reward component peaks at 1 when agent matches target speed, and decreases linearly to zero the further it is. The desired speed in context-sensitive; it reduces gradually to zero as the agent nears a red traffic light and resumes the standard maximum speed once the light turns green. This same approach applies when the agent encounters obstacles, including pedestrians, bicycles, or other vehicles. In other scenarios, the agent maintains a constant maximum speed of 40 km/h.

- Position: this reward component is negatively correlated with the agent's deviation from the lane's center, as determined by the waypoints and represented in Figure 4.7. The reward reaches its peak of 0 when the agent is precisely centered in the lane and declines to -1 as it strays to a maximum set distance (Dmax) from the center. If the agent exceeds Dmax, which in our experiments is 2 meters (the distance from the lane's center to its edge), the episode ends. Episodes also ends if the agent collides with an object, runs a red light, or is immobile without any apparent obstacles or red lights, with a consequent reward of -1 for these terminations.

- Rotation: this reward component is inversely related to the angular disparity between the agent's orientation and that of the closest waypoint on the optimal path, as illustrated in Figure 4.7. This addition encourages the agent to align with the correct lane orientation, promoting smoother navigation.

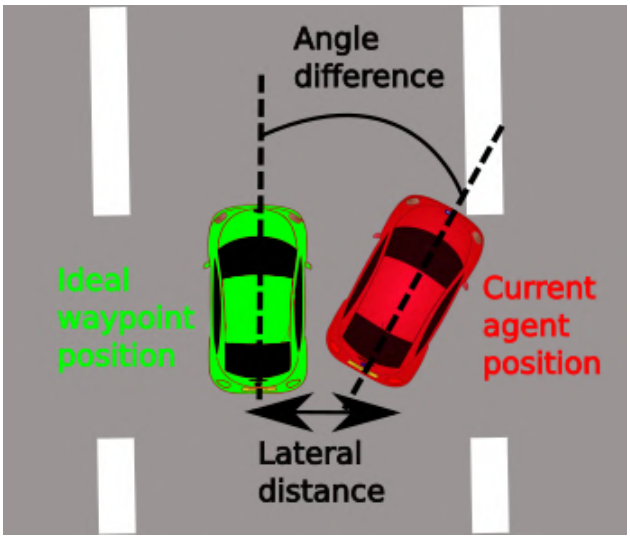


Figure 4.7: Lateral distance and angle difference for lateral and angle reward computation. The difference is measured between the ideal waypoint (in green) and the current agent position (in red). Figure and caption from [Toromanoff et al., 2020a].

Ablation study over components of this reward can be found in [Toromanoff et al., 2019a]. Since this reward is normalized to have a range between 0 and 1, we set the demonstration reward to $r_{demo} = 1$.

4.4.2.3 The global pipeline

The GRIAD pipeline is different at training and at inference. Indeed, training is two folded with 1) training of the visual subsystem which is then frozen, and 2) training of the decision-making subsystem, and is represented in Figure 4.8.

Inference is end-to-end *i.e.* feeds sensory data and outputs an action without any explicit intermediate view of the data in the pipeline, and is represented in Figure 4.9.

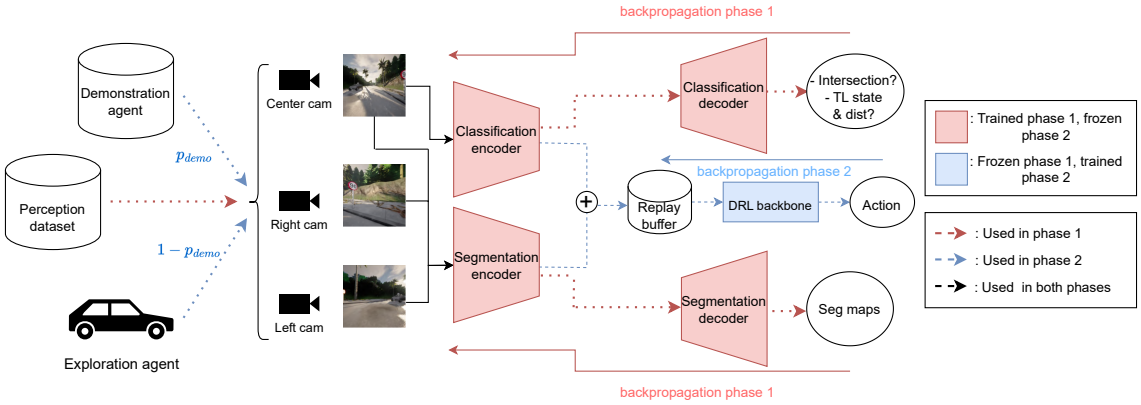


Figure 4.8: GRI pipeline is trained in two phases: (1) Visual encoders are pretrained on several auxiliary tasks, which are semantic segmentation, road type classification, relevant traffic light presence, and if there is such a traffic light, its state and the distance to it. (2) Visual encoders are frozen and a GRI-based DRL network is trained with both pre-generated expert data with an offline demonstration agent and an online exploration agent gathering data from a simulator. At any given training step, the next episode to add to the replay buffer comes from the demonstration agent with a probability of p_{demo}, else from the exploration agent. Actions correspond to a pair (steering, throttle) to apply to the car.

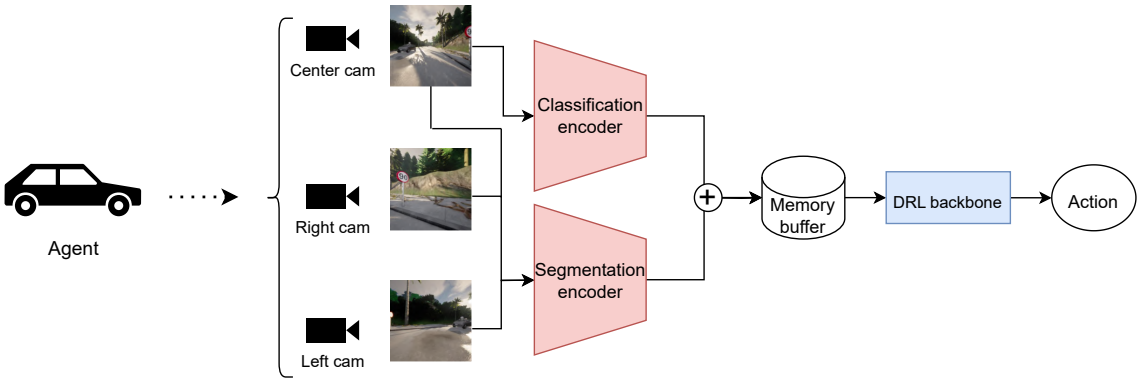


Figure 4.9: GRI is end-to-end at inference. Input are RGB images from onboarded cameras on the agent and output is an action. The DRL backbone feeds from the last 4 IAs encoded by the visual subsystem which are stored in the memory buffer. Actions correspond to a pair (steering, throttle) to apply to the car.

4.5 Experimental results

4.5.1 Ablation study on the NoCrash benchmark

We provide an ablation study on the demonstration agents in the GRIAD setup. We compare GRIAD trained with nine exploration agents and three demonstration agents, to GRIAD trained with nine explorations agents, i.e., regular RL on the NoCrash benchmark [Codevilla et al., 2019]. Agents are trained on a single environment (Town01) under a specific set of training weather. They are then evaluated on several scenarios with different traffic density on the training (Town01) and test (Town02) town with training and test sets of weather.

For these experiments, GRIAD was trained on 16M samples corresponding to 12M explo-

ration steps + 25,000 expert data, which have been sampled 4M times in total. We present an ablation study to show how GRIAD compares to RL without GRI *i.e.* without demonstration agents, using two vanilla RL models: one trained on 12M exploration steps and the other on 16M exploration steps. Each agent was trained using the exact same visual encoder trained on another demonstration dataset of 100,000 samples coming exclusively from Town01 under training weather. Results are presented in Table 4.2.

Task	**Town, Weather**	**GRIAD**		
		Explo. 12 M	**Explo. 12 M + Demo. 4 M**	**Explo. 16 M**
Empty	train, train	96.3 ± 1.5	**98.0 ± 1.7**	**98.0 ± 1.0**
Regular		95.0 ± 2.4	**98.3 ± 1.7**	**98.6 ± 1.2**
Dense		91.7 ± 2.0	93.7 ± 1.7	**95.0 ± 1.6**
Empty	test, train	83.3 ± 3.7	94.0 ± 1.6	**96.3 ± 1.7**
Regular		82.6 ± 3.7	93.0 ± 0.8	**96.3 ± 2.5**
Dense		61.6 ± 2.0	**77.7 ± 4.5**	**78.0 ± 2.8**
Empty	train, test	67.3 ± 1.9	**83.3 ± 2.5**	73.3 ± 2.5
Regular		76.7 ± 2.5	**86.7 ± 2.5**	81.3 ± 2.5
Dense		67.3 ± 2.5	**82.6 ± 0.9**	80.0 ± 1.6
Empty	test, test	60.6 ± 2.5	**68.7 ± 0.9**	62.0 ± 1.6
Regular		59.3 ± 2.5	**63.3 ± 2.5**	56.7 ± 3.4
Dense		40.0 ± 1.6	**52.0 ± 4.3**	46.0 ± 3.3

Table 4.2: Ablation study of GRIAD on the NoCrash benchmark. Mean and standard deviation are computed over three evaluation seeds. Score is the percentage of road completed without any crash. Explo. xM + Demo. yM means the network has been trained on x million samples from exploration agents and y million samples from demonstration agents. GRIAD leveraging only exploration agents is regular RL. GRIAD experimentally generalizes more on test weather than RL trained on 12 M and 16 M exploration samples and globally gives the best agent. GRIAD trained with demonstration agents only leads to scores of 0 on every task, as every sample has the same reward during the training.

We first observe that GRIAD systematically gives better results than RL with 12 M steps, while taking approximately the same time to train (+∼4%). Indeed, as demonstration agents do not require any interaction with the simulator, we can add them at a negligible cost and still improve results. We also observe that while RL with 16M steps does better than GRIAD on train weather, GRIAD gives better results on the test weather while being ∼25% faster to train. We believe this is because RL tends to overfit on a given environment if it explores it too much. Hence, replacing 4M exploration data with 25,000 demonstration data sampled ∼160 times each appears to reduce the overfitting and allows a better generalization.

Further tests were conducted by training the same pipeline with SQIL [Reddy et al., 2019a] but the evaluation reward stayed particularly low during the 20M steps of training. The first test showed SQIL to be inefficient for end-to-end autonomous driving on CARLA, as it did not learn to drive at all, staying static or drifting off the road most of the time. It reached a score of 0 on every evaluated task. We believe that the reward signal as defined by SQIL is not rich enough to allow the network to converge on such a highly complex task.

4.5.2 On the CARLA leaderboard

We trained GRIAD for 60 M steps (~45 M exploration steps +200,000 expert data sampled ~15 M times). Both visual and decision-making parts were trained on all available maps with all available weather. We compare the top three of camera-based and LiDAR-based methods, also distinguishing methods exploiting or not the Inertial Movement Unit (IMU) sensor, on the CARLA Leaderboard. Our method outperforms World on Rails, the previous comparable leading method on the CARLA leaderboard, by ~17% on the main metric, the driving score, while using fewer sensors.

However, more recent LiDAR-based methods, or methods exploiting IMU sensor, give significantly better results but cannot be compared directly as inputs are of a different nature. Indeed, LiDAR allows accurate depth measurement, which allows an increase of accuracy on vision related task after fusion with camera data [Hu et al., 2022]. IMU sensor replaces approximate orientation estimation by precise measurement, therefore providing accurate positional information. Therefore, using LiDAR and/or IMU sensors on top of camera leads to richer input features for deep learning models. In this work, we chose to build an autonomous driving system using camera only, as it leads to less expensive and complex systems. GRIAD pipeline can be augmented with other sensors by changing the vision subsystem. CARLA Leaderboard results are presented in Table 4.3.

Method	Cam.	LiDAR	IMU	DS	RC	IS
GRIAD (ours)	3	✗	✗	**36.79**	**61.85**	**0.60**
Rails [Chen et al., 2021b]	4	✗	✗	31.37	57.65	0.56
IAs [Toromanoff et al., 2020b]	1	✗	✗	24.98	46.97	0.52
TCP [Wu et al., 2022]	1	✗	✓	**75.13**	**85.53**	**0.87**
Latent Transfuser [Prakash et al., 2021]	3	✗	✓	45.2	66.31	0.72
LBC [Chen et al., 2019]	3	✗	✓	10.9	21.3	0.55
ReasonNet [Shao et al., 2023b]	4	✓	✓	**79.95**	89.89	**0.89**
LAV [Chen and Krähenbühl, 2022a]	4	✓	✓	61.8	**94.5**	0.64
InterFuser [Shao et al., 2023a]	3	✓	✓	76.18	88.23	0.84
Transfuser+ [Prakash et al., 2021]	4	✓	✗	**50.5**	**73.8**	**0.68**
Transfuser [Prakash et al., 2021]	4	✓	✗	34.6	69.8	0.60

Table 4.3: Top three of camera-based and LiDAR-based agents with and without IMU sensors on the CARLA Leaderboard on August 2023. Results of reproduced methods are not considered. Driving metrics are: driving score (DS, main metric), route completion (RC), and infraction score (IS). Higher is better for all metrics. Our method improves the driving score by 17% relative to the prior camera-based IMU-less state-of-the-art method [Chen et al., 2021b], while using fewer cameras than the two other best methods in this category. Underlined methods were published before GRIAD. GRIAD was state-of-the art at time of publication.

4.6 Limitations and quantitative insights

The main limitations of this method are the consequences of our initial hypothesis that demonstration data can always be associated with a constant maximal reward r_{demo}.

A first limitation occurs if the demonstration data are not constantly optimal, e.g., due to low expert performance on some aspect of a given task, as this introduces noise in the reward function. This is the case in our demonstration dataset on the CARLA simulator, as expert data have been generated using an imperfect autopilot containing $\sim$10% noisy demonstrations. Still, GRIAD showed to improve our model by a significant margin over vanilla RL. Therefore, we can consider the GRI setup to present some robustness to noisy demonstrations.

A second limitation of our approach is the warm-up phase on some difficult environments, as observed in Figure 4.3 on HalfCheetah-v2 and Humanoid-v2. This warm-up phase can be seen as the consequence of a distribution shift. Indeed, GRI suffers from a sort of distribution shift when the training expert data mostly represent actions made in states not reached yet by the exploration agents. In particular, we observed this effect on HalfCheetah-v2: the expert agent does not walk but jumps as soon as it touches the ground, which is a complex yet highly efficient strategy. However, to reach a state where it can successfully jump, it needs to warm up to gain the required speed and momentum by doing some low reward actions. Hence, our GRI-SAC agent learns to jump before it is able to walk, making it fall. Once the agent learned how to reach the jumping state, rewards steadily increase until convergence. However, we observe that the lower the proportion of demonstration agents is, the faster the model is able to recover from this distribution shift. Indeed, collecting more exploration data following the current agent policy compensates for the distribution shift between demonstration and exploration data.

Finally, a third limitation of our approach is the inconsistency of the rewards associated with some common actions collected by both the demonstration and exploration agents. Still for the HalfCheetah-v2 example, the demonstration agent will reward expert actions at the beginning of the agent run with the high demonstration reward, while the exploration agent will receive poor reward for the same exact actions. This induces a sort of discrepancy between data coming from the offline demonstration agent and experiences coming from the online RL exploration agent. It also implies an overestimation of demonstration actions. However, allocating high reward to demonstration data which are not correlated with the actual reward of the environment might encourage the agent to get to states closer to the expert ones. Nonetheless, it is difficult to assess the impact on the training in practice.

4.7 Conclusion

This chapter introduces the basics of model-free Deep RL and how to distillate expertise in such models training to improve stability and efficiency. In particular we present General Reinforced Imitation (GRI), a novel method for distilling expert data in RL training, and how we applied to End-to-end Autonomous Driving with the GRIAD method. GRIAD achieved state-of-the-art results on the CARLA leaderboard and won the CARLA Challenge 2021.

To go further. This method showed that simple RL enriched with expert knowledge can outperform well-designed complex IL methods. However, the GRIAD approach came with a significant margin of improvement. In particular, it focused on camera only based autonomous driving. However, observations in Table 4.3 highlighted a correlation between number of sensors and higher metrics. More specifically, building a bird-eye-view via LiDAR and camera fusion [Chen and Krähenbühl, 2022b, Shao et al., 2023c] led to improved result as environment representations are both richer and simpler. We intuited that if leveraging such a simplified representation of the environment could improve GRIAD results, it will be exploited even more efficiently with model-based RL.

Therefore, we implemented a MCTS-based approach inspired by MuZero [Schrittwieser et al., 2020] on centerline-centered bird-eye-view inputs generated by the carla-birdeye-view library available on GitHub [1]. In this approach, bird-eye-view input images are encoded via a CNN and each tree operation is tackled via another neural network. An example of generated bird-eye-view is represented in Figure 4.10.

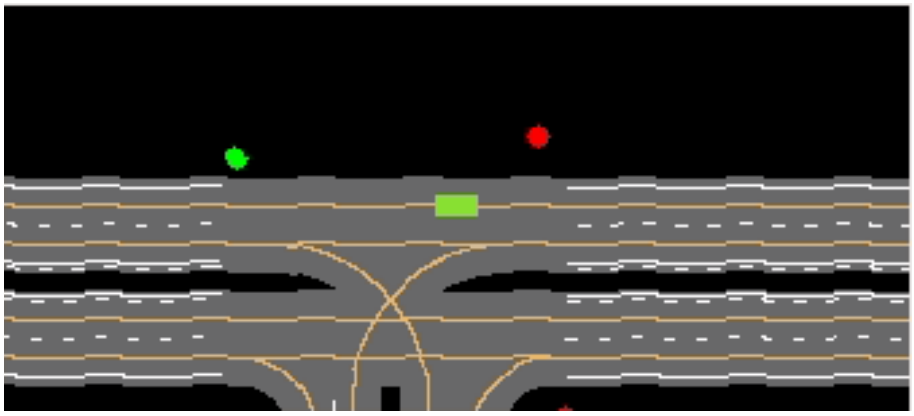


Figure 4.10: Bird-eye-view built by fetching ground truth data from the CARLA API. Generated with the library carla-birdeye-view available on GitHub. We observe some artifacts in the road layout.

Despite a lot of hyperparameters tuning, training were highly unstable and sample inefficient, as a month of training only led the ego agent to decently follow centerline and stay on the drivable area. We believed this to be due to the fact that our model was training several neural networks in the same time (one for image encoding, one for state forecasting, and another one for state evaluation *i.e.* infering a reward from an encoded state), while being dependant of the slow CARLA simulator for environment exploration and data gathering. Therefore, we decided to directly focus on higher level semantics than images to allow more explicit transitions and reward functions, and designed an MCTS-based mid-to-end method for autonomous driving, presented in Chapter 5.

[1] https://github.com/deepsense-ai/carla-birdeye-view

Chapter 5

Leveraging learned priors in model-based RL for mid-to-end autonomous driving

Contents

In this chapter, we introduce the notion of Monte-Carlo Tree Search and how we applied it for mid-to-end autonomous driving in a partially-learned environment by leveraging supervised learning to train a learned prior in [Chekroun et al., 2023a].

5.1 Monte-Carlo tree search and deep learning

This section introduces the intuition behind the notion of Monte-Carlo Tree Search (MCTS) and how it can be, and has been, integrated with supervised learning for planning task.

5.1.1 Introducing the Monte-Carlo tree search

MCTS is a popular algorithm used in decision-making processes, particularly in game-playing and other domains with large search spaces. It was introduced in 2006 for computer Go [Coulom, 2006a, Chaslot et al., 2006, Yoshimoto et al., 2006], a complex board game, but has since been applied to various other games and problems. In the context of MCTS, "Monte Carlo" refers to the use of random sampling or simulation to make informed decisions. The basic idea is to explore the decision space by sampling possible outcomes and using the results of these samples to guide the decision-making process. The "Tree Search" refers to the MCTS functioning, which builds and explores a search tree that represents the possible decisions and their outcomes. The tree is expanded and updated iteratively and each node of the tree is scored via a pre-defined reward function. The algorithm finally selects the branch maximizing the cumulated reward. Therefore, MCTS is used to find the planning *i.e.* sequence of actions in a decision-making process, typically in the context of game playing with a perfectly known and deterministic environment. The different steps of MCTS functioning are represented in Figure 5.3.

5.1.2 Integrating deep learning with MCTS

Integrating MCTS with deep learning techniques has emerged as a compelling approach to enhance decision-making processes in various domains and yields superhuman capability on some games with well known deterministic environments. AlphaGo [Silver et al., 2016], developed by DeepMind, marked a significant advancement in the field by mastering the game of Go. Its strength lies in its use of a combination of supervised learning from human games and MCTS from self-play. AlphaGo demonstrated remarkable strategic understanding and surpassed human champions, such as Lee Sedol. However, AlphaGo heavily relies on a vast dataset of human games, making it less adaptable to novel situations. Additionally, its computational requirements are immense, limiting its practicality in real-world applications. AlphaZero [Silver et al., 2017], a successor to AlphaGo, addressed these limitations. Unlike its predecessor, AlphaZero is trained from scratch through self-play without human data, as represented in Figure 5.1. This approach allows it to generalize strategies beyond the confines of human knowledge, showcasing its adaptability and creativity. AlphaZero's main strength lies in its ability to learn optimal strategies with minimal prior knowledge.

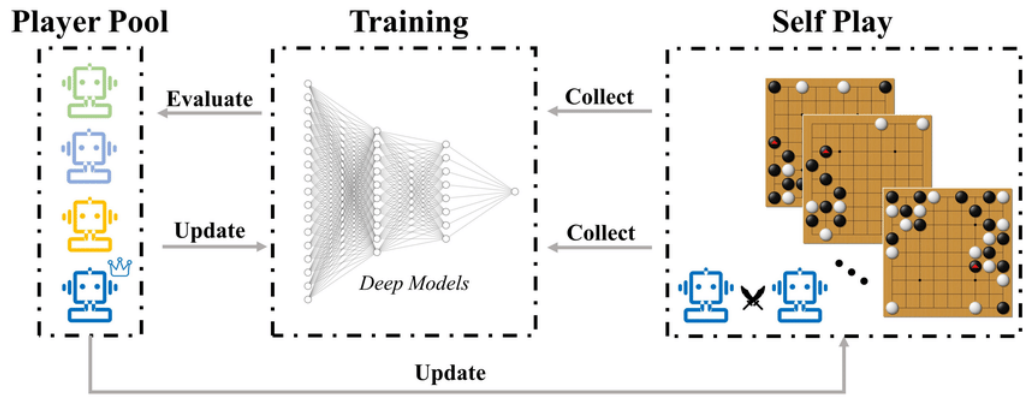


Figure 5.1: AlphaZero pipeline. Games are continuously generated via self-play and used to train a neural network. After each round of training, the new model is compared with the previous one, and the training process restart on the model yielding the highest metrics. Figure from [Gao and Wu, 2021].

MuZero [Schrittwieser et al., 2020], also developed by DeepMind, is a further evolution. Unlike AlphaZero, MuZero is not game-specific and can handle a variety of environments. Its strength lies in its capacity for model-based planning without an explicit model of the environment, making it more versatile in dynamic scenarios. MuZero's weaknesses include the computational demands required for training, its sensitivity to hyperparameters, and the lack of theoritical guarantees caused by its implicitness.

In the realm of autonomous driving, Chen et al. [Chen et al., 2020b] integrated MCTS with deep learning but relied on implicitness for the tree transitions and the value function, possibly leading to inexplicable behaviors which are not desirable for this domain of application. Other published methods effectively leverage MCTS or MCTS-like approaches but constraint their applicative fields for decision-making to custom environment such as highway driving [Ha et al., 2020, Galceran et al., 2015], lane changes [Sunberg and Kochenderfer, 2022], or high level tactical decisions [Hoel et al., 2019]. Cai et al. [Cai and Hsu, 2022] extended the Partially Observable Markov Decision Process (POMDP) from [Sunberg and Kochenderfer, 2022] to unregulated dense traffic urban driving, *i.e.* without any driving rules, but uses high level actions at the lane level (an action is a tuple consisting of a lane decision in {Left, Keep, Right} and an acceleration in {Acc, Maintain, Dec}), which is then executed by another heuristics algorithms, and does not provide comparisons with other state-of-the-art autonomous driving algorithms.

5.2 MCTS built-around predictions for planning explicitly

This section introduces MCTS Built-Around Predictions for Planning Explicitly (MBAPPE) [Chekroun et al., 2023a], a novel approach to motion planning for autonomous driving combining tree search with a partially-learned model of the environment by leveraging both an MCTS and supervised learning.

5.2.1 Introducing MBAPPE

MBAPPE focuses on the mid-to-end stage of autonomous driving, presuming that perception tasks have already been accomplished and working toward an efficient and explainable motion planning, as presented in Section 1.1.2. In this realm, recent research mostly focus on Imitation Learning (IL) [Huang et al., 2023, Renz et al., 2023] or hybrid IL and rule-based methods [Dauner et al., 2023, Hallgarten et al., 2023]. However, rule-based methods for autonomous driving are limited by their lack of scalability, adaptability, robustness in complex and ambiguous situations, and their inability to handle unconventional scenarios. This contrasts with machine-learning based approaches that address these limitations through data-driven learning and adaptability. Nonetheless, while Neural Networks (NN) provide a powerful and flexible tool for learning to drive using supervised labels with IL methods [Bojarski et al., 2016, Prakash et al., 2021], they remain limited in the long-term understanding of the consequences of their actions. Therefore, they may not comprehend the full scope of interactions with the map and other agents. Deep Reinforcement Learning (Deep RL) based methods [Kendall et al., 2019, Chen et al., 2021b, Chekroun et al., 2023b] aim to incorporate long-term returns of such consequences in the training of these networks. However, this causal understanding remains implicit and not guaranteed, and Deep RL training is most often sample inefficient.

MBAPPE aims to get the best of both worlds by using an IL prior to guide a MCTS [Coulom, 2006b, Kocsis and Szepesvári, 2006] into explicitly exploring the consequences of actions, validating the NN trajectory if it respects driving constraints, or exploring new actions if required. The main challenge in running a MCTS is that it generally assumes environment transitions to be deterministic and perfectly known. While this is true for the displacement of the ego vehicle given its actions, and for the update of the map that remains the same, other agents will also move on their own accord. In order to have a realistic world model, MBAPPE leverages an IL model to predict all the other agents future trajectories. This way we get an approximate of the future transitions that enables us to roll out the consequences of our chosen actions on multiple time-steps. In other words, MBAPPE extends the MCTS paradigm to partially-learned environment and apply it to autonomous driving.

5.2.2 MBAPPE framework

The MBAPPE pipeline. At each time-step, a neural network (based on an open-loop version of Urban Driver [Scheel et al., 2022]) predicts an estimation of the ego trajectory and of the future trajectories of every other agents around the ego. This information is fed to the MCTS, which will deploy an internal lightweight simulation where the ego trajectory is used as a prior to guide the first steps of exploration, and other agents trajectories are leveraged to build the world model. At each simulation-step, which follows a planning time axis inside the tree, the MCTS explores the possible actions and internally simulates the evolution of the environment to check how those explored actions will impact its driving performances (driving out of area, check for collisions with static objects, check collisions with other agents thanks to their estimated trajectory, etc). The global pipeline is represented in Figure 5.2.

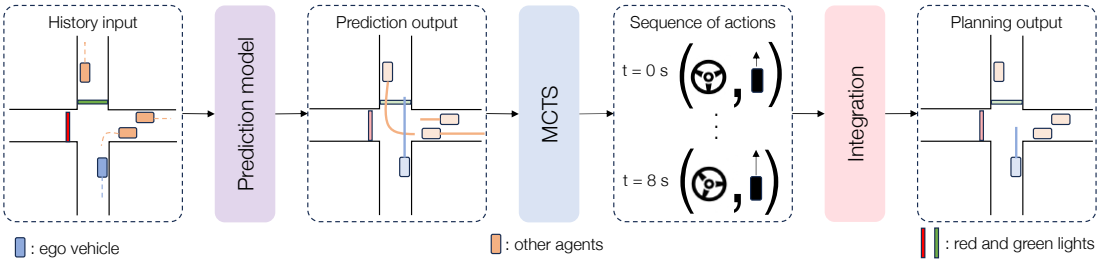


Figure 5.2: **The MBAPPE pipeline** At each simulation time step, a prediction model infers future trajectories of other agents in the scene. This information is fed to the MCTS which outputs a sequence of consecutive low-level actions. Those are integrated to form an improved trajectory planning for the ego.

A Partially-Learned World Model. By construction a MCTS requires a perfectly known and deterministic environment to explore and plan in. However, the autonomous driving setup is neither perfectly known nor deterministic. In particular, as we consider the mid-to-end driving problem as presented in Section 1.1.2, the perception is supposed solved so that we have knowledge of instant positions of other agents and map layout, but the environment's dynamic is unknown, which is unfit for the internal simulation required for MCTS exploration. To overcome this limitation, we designed a partially-learned environment combining known features with the ones learned by a neural network through supervised learning. This environment is made of two categories of features:

- Known features:
  - The map information, including traffic light,
  - Static objects such as traffic cones and barriers
  - Dynamic objects such as neighboring vehicles, bicycles or pedestrians, which we will consider as other agents evolving in the simulated environment
- Learned features:
  - Estimated future trajectories of other agents given by the neural network prediction.

Hence, MBAPPE generalize MCTS to a partially-known environment for autonomous driving.

5.2.3 MCTS design and tree steps

Our MCTS is based on a kinematic bicycle model of the vehicle. Actions are low-level are defined as a tuple (a, δ), where a is the acceleration and δ the steering angle. Accelerations and steering are discretized in 13 values each, in the respective range of $[-3, 3]$ m.s^{-2} and $[-\pi/4, \pi/4]$ rad. Actions are integrated every 0.1 s.

The simulation process of our tree search is detailed in Fig. 5.3. The tree is initialized with a single root node representing the current context. Each tree node stores 3 values: Q the expected

return, P the action prior and N the number of visits. The nodes are built and evaluated iteratively through the following steps:

- **Selection**: We follow the PUCTS [Silver et al., 2016] formula to select the next action following a trade-off between the exploitation of Q and the exploration of unvisited nodes with low N.

  At a node state $\mathcal{S}$ the action $\mathcal{A}$ is chosen using the following formula:

$$\mathcal{A}_t = \arg\max_{\mathcal{A}} \left[Q(\mathcal{S},\mathcal{A}) + c_{\text{puct}} \cdot P(\mathcal{S},\mathcal{A}) \cdot \frac{\sqrt{\sum_{\mathcal{B}} N(\mathcal{S},\mathcal{B})}}{1+N(\mathcal{S},\mathcal{A})} \right] \tag{5.1}$$

  with c_{puct} a hyper-parameter balancing the trade-off between exploration and exploitation. We found $c_{puct} = 2$ to perform the best in our experiments.

- **Expansion**: We expand leaf nodes by all physically possible actions from the state of the leaf node, following a prior P and some continuity constraints. These constraints ensure both comfort and physical feasibility of successive actions. Prior design and continuity constraints are described in Section 5.2.4.

- **Evaluation**: We consider that driving rewards are rather short term (crash or not, exit road or not within the next 6 or 8 seconds). Therefore they do not need to be bootstrapped by a learned value network, but rather can be evaluated at the current simulation step by checking for them directly. Our computed reward r_t at state s_t is made of these main components:

  - Progress: distance advanced since the last node, normalized by maximum allowed speed limit ($[0, 1]$),
  - Collision: penalty for collision with car and pedestrian (-5) or object (-2),
  - Route: -0.5 if the vehicle is not on the expected road,
  - Drivable area: -1 if the vehicle is not on the drivable area,
  - Center of the road:
    - $-sin(\theta)/2$ where θ is the angle difference between the ego heading and the closest centerline heading,
    - $-d/2$ where d is the distance between the ego position and the closest centerline.

- **Back up**: We update the Q values using the cumulative reward as in MuZero [Schrittwieser et al., 2019]:

$$\begin{aligned} G^k &= \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} \\ Q\left(s^{k-1}, a^k\right) &:= \frac{N\left(s^{k-1}, a^k\right) \times Q\left(s^{k-1}, a^k\right) + G^k}{N\left(s^{k-1}, a^k\right) + 1} \\ N\left(s^{k-1}, a^k\right) &:= N\left(s^{k-1}, a^k\right) + 1 \end{aligned} \tag{5.2}$$

  We use a discount factor γ of 1.

The different tree steps are represented in Figure 5.3

5.2.4 Prior and continuity constraints

An efficient MCTS exploration process can be achieved by leveraging two approaches. Firstly, providing the MCTS an intuition over actions to be explored in order to prioritize the more probable ones. This issue is tackled using a prior over the distribution of actions for each node. This prior is usually learned and inferred for every node [Schrittwieser et al., 2019], which is computationally expensive, or handcrafted. Secondly, to further streamline the exploration process, we narrowed down the action space, thereby reducing the overall actions that need to be explored to the most critical ones. To achieve this, we integrated continuity constraints into the MCTS to ensure not only the physical feasibility of the actions explored but also to enhance comfort and to reduce the exploration time.

5.2.4.1 The prior

We designed a prior which relies on both handcrafted rules and learned rules, all without incurring any additional computational overhead.

The prior function is made of two parts:

- The handcrafted prior P_h prioritizes exploration around the constant speed with null steering angle,
- The learned prior P_l is obtained by deriving the prediction of the ego trajectory by the NN into consecutive actions. This prior advantages the possibility of following NN actions for the first T time steps of the internal simulation of the MCTS. We found $T = 1$ s to perform the best in our experiments.

Both P_l and P_h are Gaussians centered on the chosen action. The Gaussian are parameterized with a very high variance ($\sigma^2 = 100$) to encourage an almost uniform exploration.

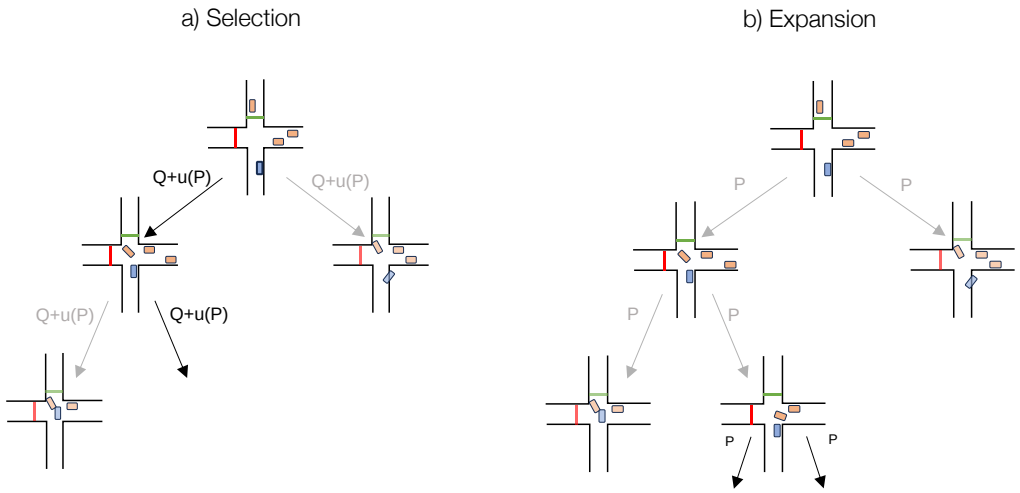
The designed prior can be written:

$$P^t = \begin{cases} P_h^t + P_l^t & \text{if } t \leq T \\ P_h^t & \text{if } t > T \end{cases} \tag{5.3}$$

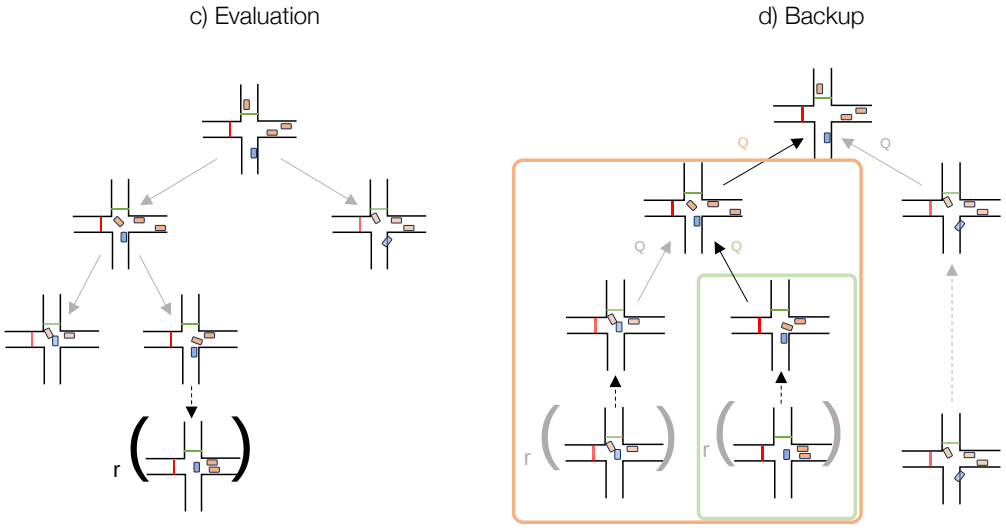
5.2.4.2 Continuity constraints

To ensure the output trajectory is physically feasible and to minimize the total number of actions to explore, we implemented continuity constraints in the MCTS, represented in Figure 5.4. These constraints are two folded:

- The Tree Constraint: At a given step t of the real-world vehicle movement, the root node of the novel tree will be constrained to explore neighboring accelerations and steering angles relatively to the actions taken at time $t-1$ by the ego in the simulator. This constraint favors a behavior continuity between successive time-steps and corresponding MCTS.



(a) **MCTS steps 1/2.** a) Selection: Each simulation pass in the tree follows a trade-off between exploitation of the best Q value of an action, and the exploration term $u(P)$ that encourages to explore nodes with less visits N along the prior P. b) Expansion: The leaf node is possibly expanded following some probabilities depending on the prior P and the continuity constraints.



(b) **MCTS steps 2/2.** c) Evaluation: The leaf node is possibly expanded following some probabilities depending on the prior P and the continuity constraints. (Backup) After the simulation, the leaf node is evaluated by explicitly computing the reward r described in Section 5.2.3. d) Backup: Q-values are updated so means of the rewards r in the sub-tree below each actions are tracked.

Figure 5.3: Representation of the four MCTS steps.

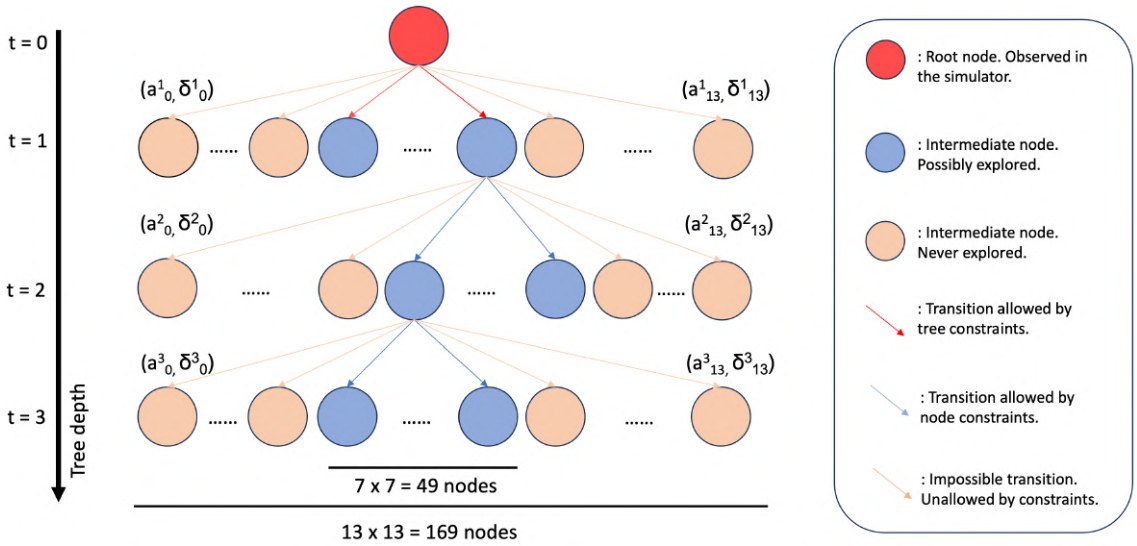


Figure 5.4: **Representation of a constrained tree exploration.** Both at the root and inside a given tree, exploration is constrained around the 7 neighboring accelerations and the 7 steering angle values relatively to the action that led to the parent node. Therefore, transitioning to a higher depth from a given node cannot yield more than $7 \times 7 = 49$ exploration steps instead of the 169 theoretically possible ones.

- The Node Constraint: During the MCTS internal expansion phase, exploration only focuses on neighboring accelerations and steering angle values relatively to the actions of his parent node. This constraint favors a behavior continuity during the expansion phase of a given MCTS.

We formulate both continuity constraints as restricting the following action (a^{t+1}, δ^{t+1}) to be within a range of $a^t \pm 0.15$ m.s^{-2} for the acceleration and $\delta^t \pm \pi/240$ rad for the steering angle with (a^t, δ^t) the action at the previous time-step for node constraints, or the last ran in simulator action in the previous tree for tree constraints.

5.3 MBAPPE on the nuPlan simulator

5.3.1 Practical details

Dataset. We show results on the nuPlan dataset. It encompasses 1500 hours worth of real vehicle motion data along with its corresponding simulator. Within the nuPlan framework, we chose to provide planners performance comparisons on the closed-loop non-reactive agents benchmark. We focus on this benchmark, as evaluations conducted in closed-loop more effectively assess an agent's driving capabilities without the need to compare them to a flawed 'ideal' behavior as typically seen in open-loop assessments. Additionally, we chose to provide non-reactive agents metrics only for our study, as our complete set of local experiments have demonstrated that outcomes are largely consistent between reactive and non-reactive agents on the nuPlan simulator, which is in line with observations from other performance benchmarks [Dauner et al., 2023, H. Cae-

sar, 2021]. This might be due to the fact that new predictions are computed at every simulation time step, as represented in Figure 5.2. All simulations are ran on 100 scenarios of each of the 14 scenarios types (totaling 1,118 scenarios in practice, as all 14 types do not have 100 available scenarios) of the nuPlan challenge, following the Val14 benchmark validation set [Dauner et al., 2023].

Score and metrics. We use the nuPlan official score, which measures driving quality between 0 and 100 through a combination of 16 normalized driving metrics related to infraction rate, ego comfort, or progress toward the goal. We decided to put a special emphasis on the metrics of collision rate (CR), driving area non-compliance (DA) and ego progress (EP) in our experiments, as they are key elements for a safe and efficient autonomous driving system.

Implementation details. For ablations studies, the number of simulation steps is limited to 256 in each MCTS. In our setup (Intel Core i7-9700K CPU @ 3.60GHz) the whole pipeline inference time is ~ 0.15 seconds for this setup, including input pre-processing, prediction model, MCTS and post-processing. The pipeline runs on CPU only. For inference speed purposes, we only expand new possible actions every 1 s. We observed no drop of performance.

Prediction models. For ablation studies we evaluated MBAPPE with Urban Driver [Scheel et al., 2022], the official nuPlan baseline for mid-to-end planning, extended to predict trajectories of all other agents in the scene in addition to the ego's. This extended version presents similar planning performance than regular Urban Driver. For comparisons with other methods, we evaluated MBAPPE with both Urban Driver and GameFormer [Huang et al., 2023], the highest performing open-source method on nuPlan designed for both prediction and planning.

5.3.2 Ablation study over the prior

An ablation study over the choice of prior is presented table 5.1. Continuity constraints are the one described section 5.2.4.

Prior		Metrics			
Learned	Crafted	CR ↓	DA ↓	EP ↑	Score ↑
-	-	6%	4%	31%	26%
✓	-	11%	3%	88%	65%
-	✓	6%	4%	95%	82%
✓	✓	**5%**	**2%**	**96%**	**86%**

Table 5.1: Ablation over the prior.

We can see from results of Table 5.1 that MCTS without prior is inefficient. Exploration being unguided, the expansion phase does not create node leading to a good reward *a priori*. Following P_l for the first steps of the simulation allowed to significantly improve the exploration phase by guiding the MCTS to stay within the driving area. Indeed, thanks to continuity constraints, a good beginning of the trajectory allows to stay on the road and reach acceptable metrics. Interestingly,

leveraging only P_l leads to an increase of the collision rate: if the MCTS first actions differ from the prior's, there will be a mismatch between the guidance it provides and the actual scenes which can lead to collisions.

Leveraging P_h allows the MCTS to prioritize exploration of the most common behavior on average (staying at around the same velocity with a null steering angle), therefore minimizing collisions and optimizing overall progress. Notably, using only this naive prior without any kind of learning already yields very good performance, highlighting the power of guided exploration in the MBAPPE method. Finally, leveraging $P_h + P_l$ allows to prioritize this kind of behavior while starting with a better heads up and leads to best results on this set of experiments.

5.3.3 Ablation study over continuity constraints

An ablation study over the choice of continuity constraints is presented Table 5.2. For these experiments, prior is $P_h + P_l$.

Constraints		Metrics			
Tree	Node	CR ↓	DA ↓	EP ↑	Score ↑
-	-	7%	4%	95%	79%
✓	-	8%	3%	96%	82%
-	✓	8%	2%	94%	82%
✓	✓	**5%**	**2%**	**96%**	**86%**

Table 5.2: Ablation over continuity constraints.

It becomes apparent that when applied separately, continuity constraints offer only marginal improvements to our method. A possible explanation is that the handcrafted identity prior already directs the MCTS towards a form of constrained exploration similar to what is achieved through node constraints. However, utilizing both node and tree constraints independently does enhance the exploration process. Importantly, the combined effects of these constraints not only substantially increase performance but also ensure a consistent selection of actions, both within a single tree and across multiple trees that correspond to sequential planning steps.

5.3.4 Comparison with state-of-the-art methods

We compare MBAPPE's performance with other state-of-the-art method on the validation scenario of the Val14 benchmark [Dauner et al., 2023]. See Table 5.3.

Baselines: Urban Driver [Scheel et al., 2022] utilizes PointNet [Qi et al., 2017b] layers to process polyline and employs a MLP following a multi-head attention block to forecast the ego trajectory. GameFormer Planner [Huang et al., 2023] exploits a Transformer to predict all agents trajectories before refining ego planning via non-linear optimization. PlanCNN [Renz et al., 2023] leverages a CNN on rasterized inputs to predicts the ego trajectory. PDM [Dauner et al., 2023]

leverages an improved IDM [Treiber et al., 2000b] model combined with a simple MLP to generate several trajectories which are then scored to return the optimal one. GC-PGP [Hallgarten et al., 2023] categorizes proposed plans according to their traversal of a route-constrained lane graph, and then identifies the most probable cluster center.

Method	CR ↓	DA ↓	EP ↑	Score ↑
Urban Driver [Scheel et al., 2022]	34%	26%	96%	47%
GameFormer Planner [Huang et al., 2023]	6%	4%	98%	84%
PDM-Hybrid [Dauner et al., 2023]	**2%**	**0%**	**99%**	**93%**
IDM [Treiber et al., 2000b]	12%	6%	95%	76%
GC-PGP [Hallgarten et al., 2023]	-	-	-	57%
PlanCNN [Renz et al., 2023]	-	-	-	73%
MBAPPE (Urban Driver)	5%	2%	96%	86%
MBAPPE (GameFormer)	3%	2%	98%	90%

Table 5.3: Val14 benchmark on nuPlan

In our experiments, we found that incorporating a prediction model into MBAPPE consistently leads to enhanced planning compared to the vanilla model. When combined with GameFormer, MBAPPE significantly boosts key metrics, achieving a score of 90% compared to 84% when using non-linear optimization techniques in the GameFormer Planner. The improvement is even more pronounced with Urban Driver MA, increasing the score from 47% to 86%. The scoring gap between MBAPPE (Urban Driver) and MBAPPE (GameFormer) arises from GameFormer's superior prediction of both other agents' trajectories and ego trajectory derived as a prior to guide exploration. However, on this benchmark designed by the PDM team, PDM-Hybrid still outperforms MBAPPE (GameFormer) by 1% to 3% on all key metrics. Nonetheless, given the larger performance gap between all other runner ups, we believe this minimal score difference to be negligible in practical driving scenarios, especially considering the gain in explainability and interpretability of MBAPPE approach compared to PDM-Hybrid. Furthermore, scores of 0% DA and 99% EA are increasingly hard to beat, which may call for alternative metrics in the future to understand performance differences. Additionally, MBAPPE has room for improvement. Specifically, the fixed and hard-coded lane widths in the internal MCTS simulation sometimes leads to nearly impossible situations with extremely narrow turns, hence diminishing overall score. Better outcomes could also be attained by incorporating a more sophisticated learned prior for each node, as suggested in other work [Schrittwieser et al., 2019, Chen et al., 2020b], or by using MCTS results to enhance the prior through continuous learning, creating a self-improving loop [Schrittwieser et al., 2019, Cai and Hsu, 2022].

Thus, MBAPPE not only delivers state-of-the-art performance, but is also an explainable and interpretable operator when applied to predictive models. This dual benefit both refines decision-making policies and provides added adaptability.

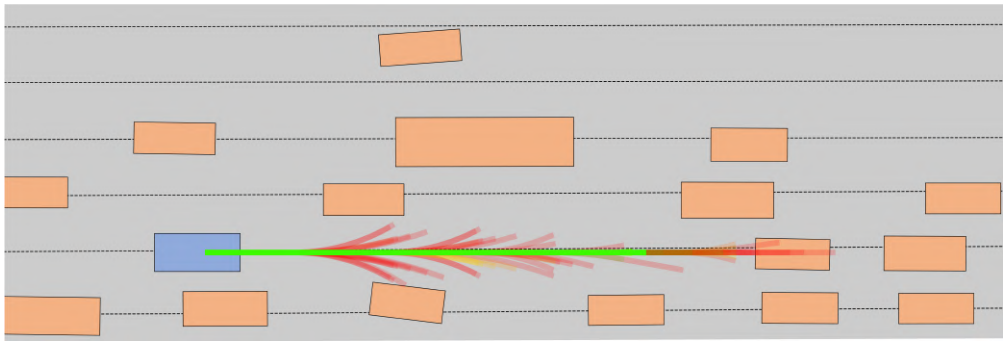
5.4 A Qualitative Analysis

This section presents qualitative visualization and analysis on MBAPPE internal thought process to emphazise on its transparency and explainability.

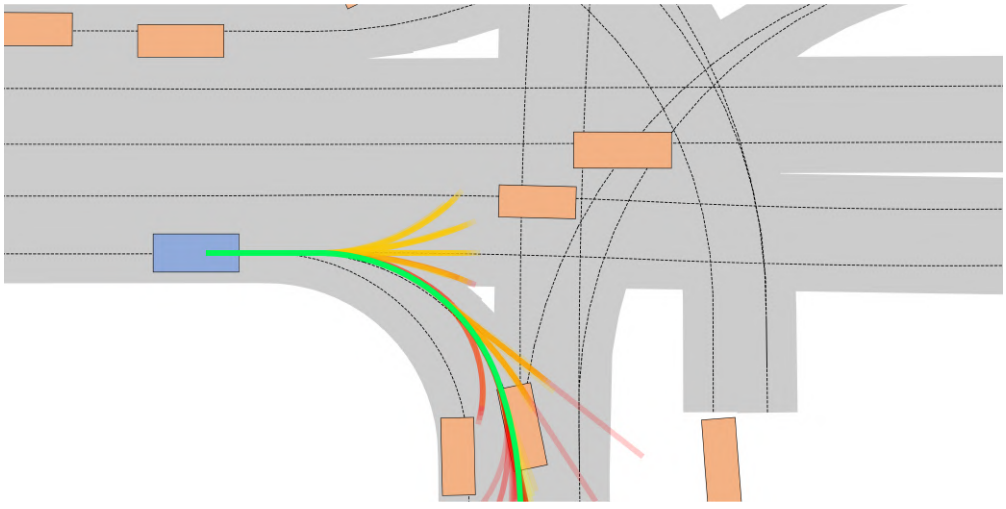
5.4.1 Qualitative observations

Qualitative visualization of the exploration done by MBAPPE at one planning step in different scenarios from the nuPlan challenge are represented in Figure 5.5. The chosen scenarios are *straight in traffic*, *turn right*, *turn left* and *high lateral speed*. We chose to illustrate these particular scenarios as they represent most of the driving time of an agent. Videos representing the MCTS exploration and decision process on this same scenarios can be found on YouTube [1].

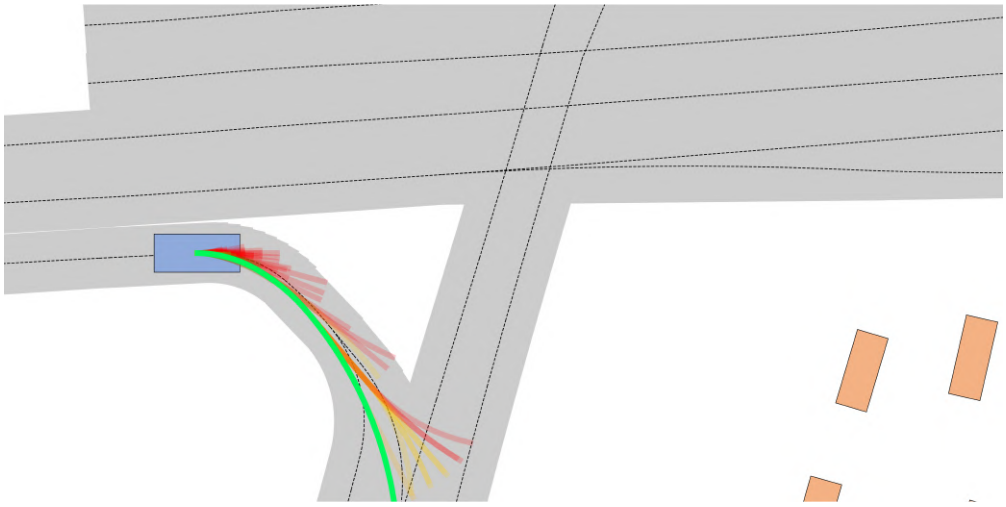
[1]Link: `https://www.youtube.com/watch?v=EzgHDyH7RfI`



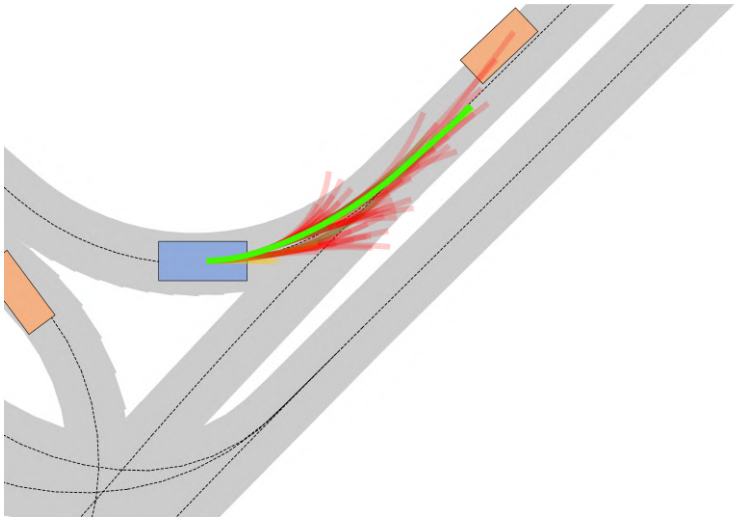
(a) One time step in *straight in traffic* scenario. The trajectory in green yields the highest Q-value. We observe that the decision of going straight and avoid collision is scored better by a significant margin, as all other trajectory are in deep red.



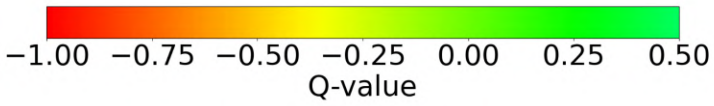
(b) One time step in *turn right* scenario. Different imagined trajectories presenting decent behavior are observable, hence the presence of both green (the highest score) and yellow (good score) trajectories. Yellow trajectories here raise good driving behavior, but does not satisfy the progress part of the reward as the objective is to turn right.



(c) One time step in *high lateral speed* scenario. The MCTS explores several possible planning and finally selects the one following the road, as it yields the highest Q-value.



(d) One time step in *turn left* scenario. The MCTS follows the road and finally select the trajectory leading the ego vehicle to go and wait behind the stopped vehicle.



(e) Color scale for the Q-value representation. A higher Q-value means a better driving behavior as captured by the accumulated reward over the trajectory.

Figure 5.5: Qualitative visualization of the exploration done by MBAPPE in one planning step in different scenarios. We display the bird-eye-view trajectory pieces in xy coordinates. The MCTS explores multiple steering angle and acceleration configurations to correctly take the turn. MBAPPE finally selects the path which maximizes the Q-value (in green).

5.4.2 An explicit and explainable method

A key benefit of MBAPPE is its conceptual simplicity: it requires only basic high-level directives in the form of a reward function (e.g., move ahead, avoid collisions, stick to the route, and remain on the road). Despite its vague prior presented in Section 5.2.4, the method yields highly effective and realistic planning, on par with the state-of-the-art. This approach eliminates the need for specific, hard-to-generalize rules, like basing decisions on the road's curvature or the speed of the car ahead, as well as the use of hardly interpretable neural networks. As a result, our approach is highly flexible, adaptable, and explainable.

Indeed, decisions of the MCTS are explainable and the internal process that led to those decisions can be easily observed and analyzed. Figure 5.6 provides an example of a decision tree of the MCTS in which we can observe several exploration branches and their consequences on the tree expansion. In particular, we observe on the green right branch that internal exploration leading to desirable behavior yields the highest Q-value and further exploration of that branch. When exploration leads to collisions or to the ego leaving its expected route, the Q-value is low and exploration stops, as shown in the red middle and orange left branches. Figure 5.6 shows that MCTS decisions-making process is transparent and explainable, thus leading to an explicit and safe planning.

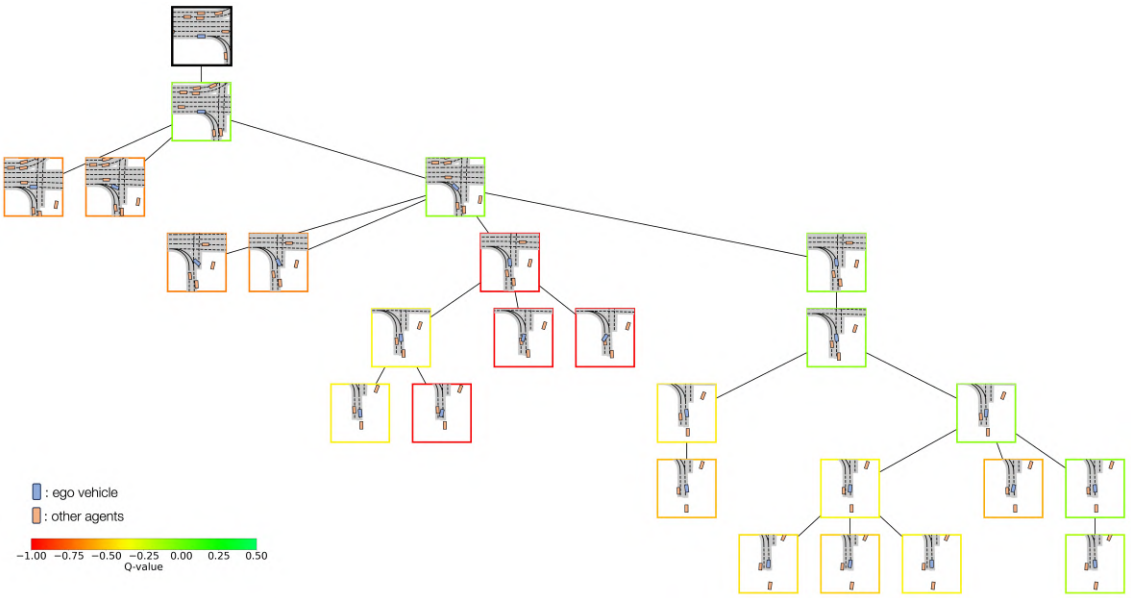


Figure 5.6: A subset of a decision tree obtained with MCTS exploration. Nodes are colored according to their Q-value. The root node correspond to the present state of the vehicle in the nuPlan simulator. We observe that the orange left branch exploration leads to the ego leaving the expected route, hence the low Q-value. The red middle branch exploration leads to a collision, thus explaining the low Q-value. The green right branch exploration presents the expected behavior and therefore has the highest Q-value. The explored planning can also be observed in Figure 5.5.

5.5 Conclusion

As a conclusion, this chapter introduced MBAPPE, a novel approach extending MCTS for planning within a partially learned environment in the context of autonomous driving. Through ablation studies, we highlighted the advantages of incorporating the designed priors and continuity constraints into the MCTS tree. Comparative analysis using a public benchmark on the nuPlan simulator revealed that MBAPPE is an effective refinement operator for planning models, consistently outperforming vanilla models across all evaluation metrics. Finally, we emphasized on the interpretability and explainability provided by this technique, which is a critical attribute for ensuring the safety and reliability of autonomous vehicles. In terms of future work, as MBAPPE improves planning model capabilities, one could fine-tune the prior network similarly to the approach used in AlphaGo [Silver et al., 2016]. This would enable the network to better emulate the MCTS output, thereby refining its priors and initiating a cycle of self- improvement. Better results could also be achieved with a more complex learned prior inferred for each node as well as learning a bootstrapped value network to estimate node expected returns in addition to the current reward [Schrittwieser et al., 2020, Chen et al., 2020c]. However this would require more network inferences and could harm the execution time.

Altogether, and even at the current stage of development, MBAPPE is an efficient and interpretable method for autonomous driving, whose explainability and robustness offers a novel and promisable approach not only applicable on simulators, but also real life autonomous driving on commercial cars. Nonetheless, this PhD thesis aims at more than research on commercial applications of autonomous vehicles and defends the idea that research in autonomous driving can be applied to greater causes and generate social benefits. Therefore, chapter 6 introduces an approach leveraging connected and autonomous vehicles for traffic dissipation.

Chapter 6

Traffic forecasting for RL-based traffic dissipation

Contents

This chapter aims at going further with the idea of autonomous driving and utilize this concept to serve a greater cause and generate social benefits. In particular, it introduces a novel approach leveraging Connected and Autonomous Vehicles (CAVs) for traffic dissipation via a hierarchical speed planner, whose design includes supervision, reinforcement learning, and mathematical modelization. The presented work was conducted at the University of California, Berkeley as a visiting PhD candidate during the Fall 2023 semester, under the supervision of Professor Maria Laura Delle Monache. My main contribution is the first authored [Chekroun et al., 2024] research of Section 6.4 which present an efficient method for real time mesoscale traffic forecasting whose output is to be leveraged by an RL-based speed planner (co-authored [Wang et al., 2024]) inside a hierarchical control framework (co-authored [Lee et al., 2024]) for mixed traffic autonomous driving for traffic dissipation.

6.1 Autonomous driving for traffic dissipation

This section introduces the notion of traffic dissipation by leveraging CAVs. This research takes place in the context of the CAV-in-the-loop Lagrangian Energy Smoothing (CIRCLES) consortium[1] to which I was a member as a supervised student of Pr Delle Monache. The whole consortium, composed of Pr Bayen, Pr Piccoli, Pr Seibold, Pr Sprinkle, Pr Work, Dr Lee and Pr Delle Monache is shared between University of California Berkeley, Rutgers University, Temple University and Vanderbilt University and seeks to apply CAVs for traffic dissipation in real life scenarios, in particular traffic dissipation in mixed autonomy traffic, *i.e.* some vehicles are autonomous and other are manually controlled.

6.1.1 A hierarchical control framework

The CIRCLES consortium introduced the MegaController [Lee et al., 2024], a control framework for the mixed autonomy traffic flow problem. The designed control framework is a hierarchical structure coordinating a centralized speed planner, focusing on macroscopic traffic flow optimization, and microscopic decentralized vehicle controllers which are either acceleration-based or ACC-based. The Speed Planner and Vehicle Controller are intrinsically connected, with the former advising the latter of upcoming events. Simultaneously, the vehicles are designed to relay their observations to a central server for the purpose of compiling data. The architectural framework of the MegaController is represented in Figure 6.1. The MegaController was implemented and tested in an open road field operational test called the MegaVanderTest (MVT) for which 100 CAVs were deployed on the I-24 interstate highway in the US on November 2022.

6.1.1.1 A centralized speed planner

The centralized speed planner, described in [Wang et al., 2024, Lee et al., 2024], is designed to optimize the overall traffic via macroscopic speed guiding for all the CAVs by utilizing traffic data

[1] https://circles-consortium.github.io

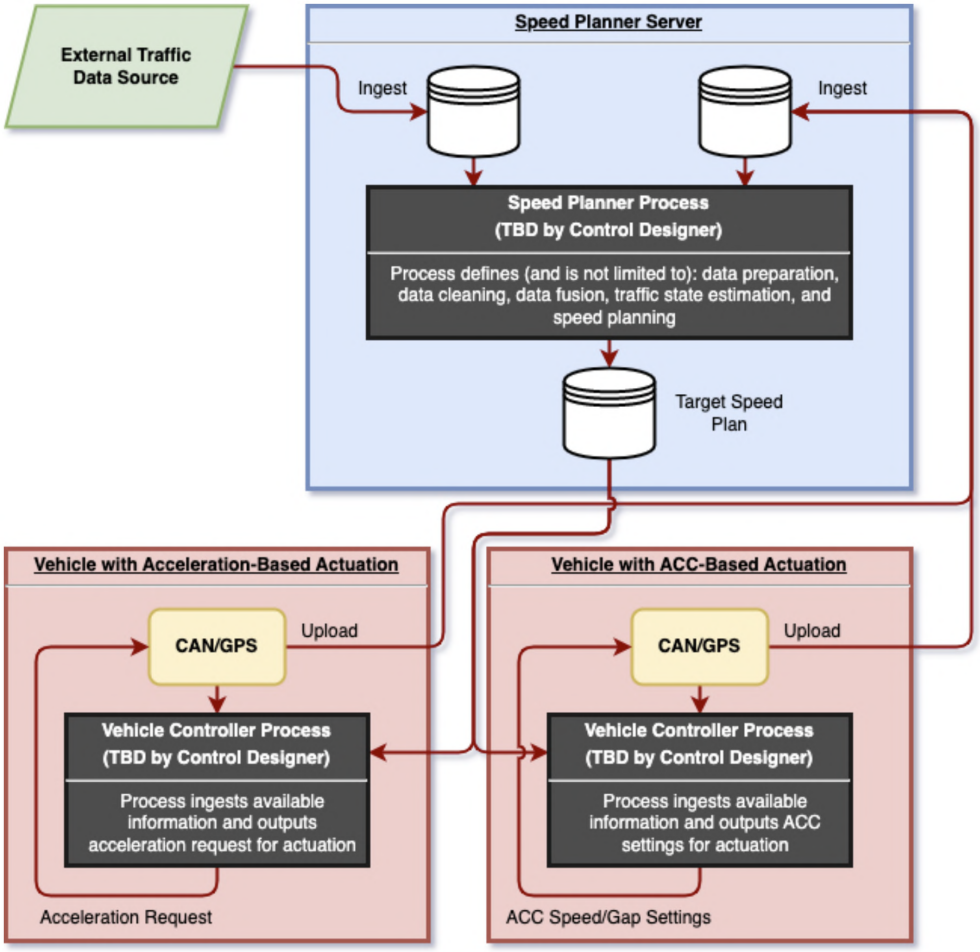


Figure 6.1: Architectural framework of the MegaController. The hierarchical and modular nature of the design allows for greater flexibility in design decisions and dealing with varied sensing and actuation capabilities of the heterogeneous fleet. The blue box represents the centralized Speed Planner unit, and the red boxes represent decentralized Vehicle Controllers, which are vehicle-dependent (that is, each vehicle has a different control architecture and thus requires a different control paradigm). The components work in concert to achieve higher level goals of flow smoothing. Figure and caption from [Lee et al., 2024].

from homogeneous sources which are the mesoscale INRIX traffic data [Reed, 2019], presented in Section 6.3.1, and microscopic data from the CAVs. It is important to note that each vehicle makes its post approximately every 1 second, and a server-side process inserts new data from INRIX approximately every 60 seconds which have a delay of up to 3 minutes, and that while the INRIX data provides a single speed across all lanes, the CAVs pings provide lane-level speed information. Figure 6.2 represents the implementation of the speed planner we tested in the MVT.

The sequence of events of a Speed Plan publication can be summarized as follows:

1. Each new INRIX update is integrated with historical INRIX data and then fed to the prediction module.

2. Vehicle observations from the previous 60 seconds are retrieved and combined with the

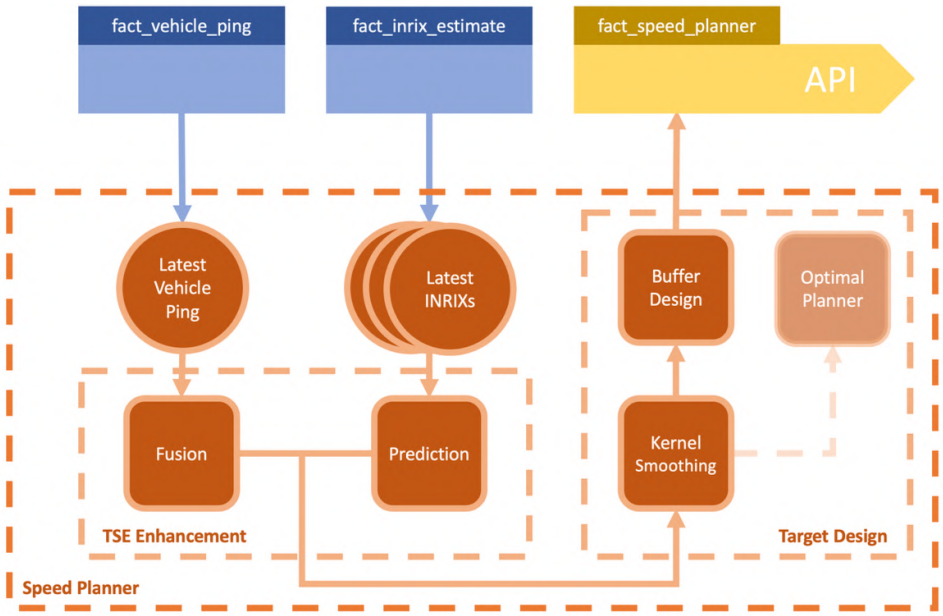


Figure 6.2: Data pipeline for the Speed Planner: 1.For each update, past INRIX data from the database are fetched as the input of the prediction module. 2. Fetch the vehicle observations of the previous 1 minute. Fuse the INRIX prediction with vehicle observation to obtain the lane level Trattic State Estimation (TSE). 3. Smooth the obtained lane level TSE with the forward average kernel. 4. Input the smoothed TSE into the bottleneck identification module. 5. If there is any standing bottleneck identified, design the corresponding buffer segment in the smoothed TSE as the target speed profile. Else use the smoothed TSE as the target speed profile. 6. Publish. Figure and caption from [Wang et al., 2024].

INRIX prediction to derive an estimation of traffic conditions at the lane level.

3. The lane-level Traffic State Estimation (TSE) is smoothed using a forward kernel average.
4. Bottleneck identification is performed on the smoothed lane-level TSE.
5. In the event that a stationary bottleneck is detected in the lane, a deceleration zone is prescribed as a buffer segment, as represented in Figure 6.3. For all other areas, the smoothed lane-level TSE is utilized as the lane-level Speed Plan.
6. Publish the Speed Plan for all lanes.

TSE Enhancement module. The TSE enhancement module is made of two submodules: an INRIX prediction module designed to overcome INRIX latency, and a data fusion module integrating real-time data from on road vehicles enabling more detailed, lane-level TSE with enhanced time-space precision. For the MVT, the implemented prediction module presented in [Lee et al., 2024, Wang et al., 2024] is a neural network based on self-attention trained to identify and predict movement of congestion frontiers within INRIX data by inferring binary vectors indicating whether the corresponding INRIX segment is a congestion frontier. However, this design of prediction module is being reconsidered and a forecasting-based method [Chekroun et al., 2024] for

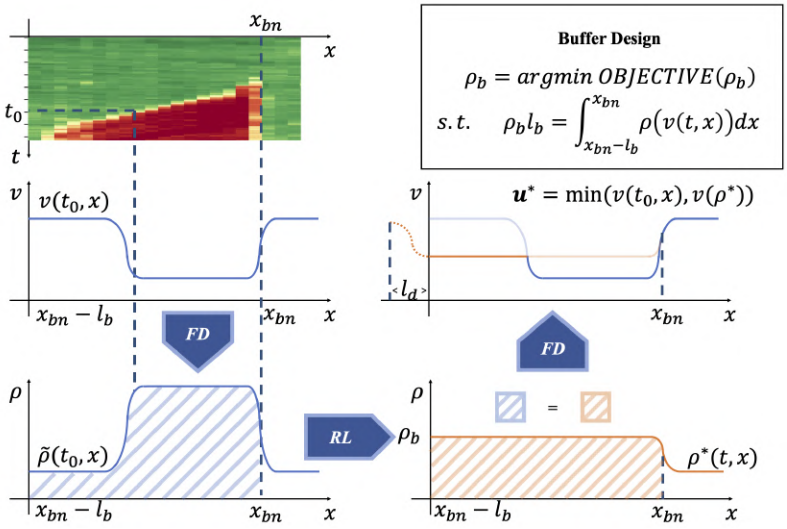


Figure 6.3: Obtain target speed profile: once a standing bottleneck is identified, the speed profile $v(t,x)$ will be converted to the density profile $\tilde{\rho}(t,x)$ by using a calibrated mapping from speed to density called a fundamental diagram (FD). RL policy selects the desirable density for the buffer area ρ_b based on $\tilde{\rho}(t,x)$, which is the critical parameter to determine a desirable density profile $\rho^*(t,x)$. The target speed profile is obtained by converting $\rho^*(t,x)$ to the speed profile and taking the minimum value at each position. Figure and caption from [Wang et al., 2024].

prediction presented in Section 6.4 will be implemented in the next MVT. After the prediction phase, the fusion module further segment INRIX data into smaller sub-segments by computing each vehicle's average speed over the past update interval and overwriting the TSE of the corresponding sub-segment. As vehicle data comes with lanes information, a distinct TSE is generated for each lane.

Target design. The enhanced TSE is leveraged to generate a target speed profile for CAVs. It is made of two submodules: a kernel smoothing which process the enhanced TSE at each time step using a chosen kernel to improve the fuel consumption in a high density traffic flow by synchronizing the driving speeds of all automated vehicles, and a buffer design which leverage a RL based algorithm to infer a buffer area upstream of the standing bottleneck [Jang et al., 2024, Wang et al., 2024]. The target speed generated by the RL algorithm is leveraged in a mathematical model of traffic (a Partial and Ordinary Differential Equation, PDE-ODE) to generate an identification of traffic density. Finally, the kernel smoothing is fed this information to generate the velocity of the next time step. Contrarily to human drivers who tend to accelerate to close the distance between their vehicle and the one in front, our proposed desired speed profile aims to slow down in advance of the right amount to create a gap between successive vehicles. This approach considers the information from the TSE which are the presence of congestion in the nearby downstream area. The proposed desired speed profile is adaptive to traffic states and is robust, as it relies on a single hyper-parameter, as detailed in [Wang et al., 2024].

Respective example outputs of TSE and Target design with interpretations are represented in Figure 6.4.

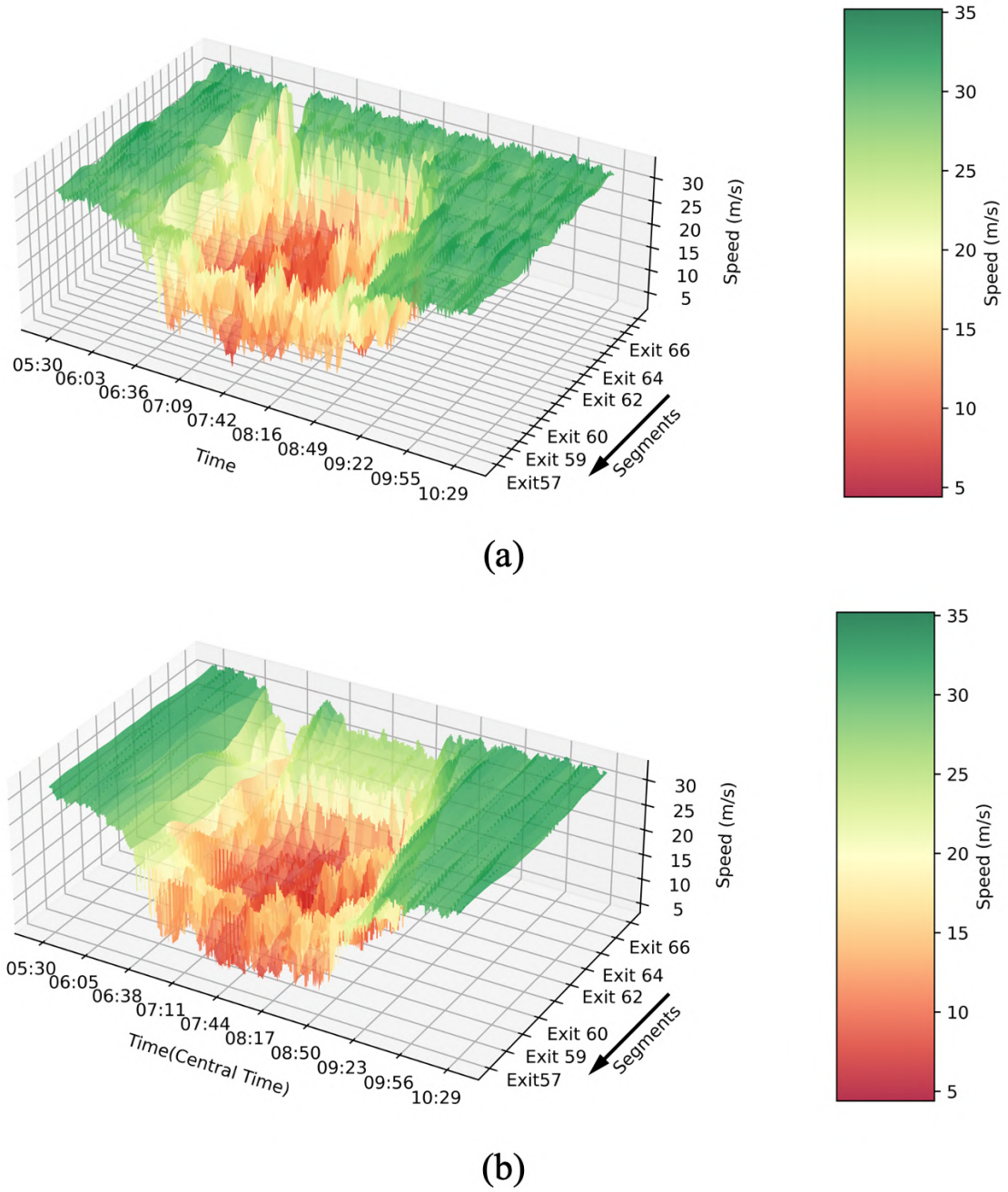


Figure 6.4: Speed surfaces of **(a) Enhanced TSE** and **(b) Target Speed**. In (b) the overall speed surface is smoother. Upstream of the congestion (Exit 62 – Exit 66 in the figure), our system designs a smooth speed gradient for vehicles about to enter the queue. The goal is to guide the traffic to slow down in advance. (Arrow indicates the traffic flow direction). Figure and caption from [Wang et al., 2024].

6.1.1.2 Decentralized vehicle controllers

Mesoscopic traffic information from the Speed Planner is then put in context with microscopic observation of each CAVs local state, such as the distance to the vehicle in front, relative speed, etc. The decentralized vehicle controller [Hayat, 2024, Lee et al., 2024] is designed to improve the traffic flow while ensuring the safety of both the AV and other vehicles around it and is inputed target speeds from the Speed Planner and input data from the in-vehicle network. Due to physical differences between vehicles of the fleet, two types of controller have been implemented for the MVT.

Acceleration-based Controller is an explicit, mathematically-defined controller defined in [Lee et al., 2024]. It aims at reaching a target speed without getting trapped in stop-and-go waves of congested traffic. It is actually made of two distinct controllers, one designed for base control *i.e.* to use under normal and emergency operating conditions, and a lane-change recovery one to behave smoothly during cut-ins and cut-outs.

ACC-based controller was deployed on 97 of the 100 vehicles from the fleet during the MVT. Rather than generating an acceleration to actuate as the acceleration-based controller does, the ACC-based controller provides outputs setpoints for the CAV native ACC system, an assistant driving system controlling a vehicle longitudinal movements based on given setpoints. Hence, safety assurance and lane-change are handled by the stock ACC. The ACC-based controller is RL-based and trained with Proximal Policy Optimization (PPO) [Schulman et al., 2017b]. The underlying MDP is presented in [Lee et al., 2024].

6.1.2 An open-road field experiment with 100 CAVs

The presented hierarchical control framework was implemented and deployed on a fleet on 100 CAVs to improve traffic efficiency on a freeway in which a small portions of vehicles were automated, others being driven by unaware drivers. The MVT is the largest field experiment to leverage CAVs to regulate traffic flow, and deployed algorithms from diverse fileds of transportation, mathematical modelization and computer science, from model based control to reinforcement learning. The control strategy adopted is hierarchical, as an upper level control speed planner generate target speeds following mesoscopic analysis which are then fed to onroad vehicles for low level control via a microscopic analysis of the scene. The dataset obtained from the experiment has been combined with datasets from I-24 MOTION [Gloudemans et al., 2023], a four mile section of the I-24 interstate highway equiped with a technology called I-24 Mobility Technology Interstate Observation Network designed to produce ultra-high resolution trajectory data of all vehicles on the roadway via 276 cameras on $\tilde{3}5$ meters roadside poles to avoid occlusion. Therefore, the MVT experiment was able to generate a large data resource for further study on the interaction of control vehicles on bulk traffic flow.

If the MVT was the first experience of this kind at that scale, the CIRCLES consortium plans on reiterating the experiment to gather more data on different highways, and improve existing technology. In particular, the current implementation of the prediction submodule of the TSE Enhancement Module in the Speed Planner has ben reconsidered toward a traffic forecasting module

able to infer the evolution of the mesoscale INRIX traffic in real-time.

6.2 Related work on traffic forecasting

Traffic forecasting is a key challenge in transportation research. With the rise of autonomous vehicles, the need for precise traffic predictions becomes more important. Accurate forecasts can greatly impact urban planning, public safety, and the overall effectiveness of transportation systems. By predicting future traffic patterns, decision-makers such as policymakers and city planners can better allocate resources, implement infrastructure improvements promptly, and develop effective traffic management strategies. This forward-thinking approach can help in reducing traffic congestion through systems designed to distribute traffic more evenly. Traffic information is relevant on several levels of granularity, on a scale from micro to macro, each presenting its own interest. Micro-scale traffic information captures detailed, vehicle-level data, such as individual speeds, positions, and behaviors, providing a high-resolution view of traffic conditions at specific locations. It is often used for fine-grained analyses, like understanding the dynamics of a single intersection, or collaborative planning to enable an energy-efficient driving [Antonio and Maria-Dolores, 2022, Delle Monache et al., 2019, Stern et al., 2018]. On the other hand, macro-scale traffic information focuses on aggregated, high-level data that provides an overall picture of traffic flow across broader areas. This can include metrics like average speeds, traffic volumes, and congestion levels, and is generally employed for long-term planning and large-scale traffic management [Derrow-Pinion et al., 2021]. Mesoscale traffic information occupies the middle ground between micro-scale and macro-scale. Specifically in our use case, it focuses on how groups of vehicles interact with each others on segmented portions of a single highway and how it impact average speeds across these distinct sections. These three types of information offer valuable insights but differ in their level of detail and computational requirements.

6.2.1 Rule-based traffic forecasting

Traffic forecasting has long been explored via rule-based methods. In particular, some research extended the Kalman Filter for traffic estimation via ensemble methods [Yuan et al., 2015] or Kalman recursions in dynamic state-space [Portugais and Khanal, 2014]. Alternative modelizations, such as particle filters [Ren et al., 2010] or spatial copulas [Ma et al., 2019], have also been leveraged to this extent. However, these methods suffer from performance decays when unexpected events provoke non predictable changes or if the allocation to a traffic pattern is inaccurate.

6.2.2 Learning-based traffic forecasting

6.2.2.1 Recurrent neural networks

RNNs, and in particular LSTMs [Hochreiter and Schmidhuber, 1997b], are lighter deep-learning based methods for forecasting able to effectively to capture and model sequential data via a sophisticated memory mechanism. Key components of LSTM networks are represented in Figure

3.4, in Chapter 3. During training, the LSTM network learns to adjust the parameters of its gates and the cell state in a way that allows it to capture long-range dependencies and patterns in sequential data. This enables LSTMs to excel in time series prediction where understanding context and dependencies over time is crucial. However, LSTM remains limited to capturing both spatial and temporal patterns in series prediction. Some work extended LSTM with self-attention on the temporal dimension to exploit temporal structure in videos[Yao et al., 2015], or to merge the cell and output states to improve performance for classification of long sequences of text [Jing, 2019a].

6.2.2.2 Coupling RNNs and convolutions

The advent of deep learning has addressed several shortcomings of rule-based methods. By learning from data, these models can account for unpredictable yet regular behaviors. Laña et al. [Laña et al., 2019] employed Spiking Neural Networks to achieve long-term pattern forecasting, adapting these predictions to real-time situations. For short-term forecasting, the Graph Convolution Network (GCN) has emerged as a potent tool. Guo et al. [Guo et al., 2020] utilized a GCN for traffic forecasting, integrating it with a latent network to glean spatial-temporal features. Mallick et al. [Mallick et al., 2022] enhanced the capabilities of GCN by incorporating ensembling methods, leveraging Bayesian hyperparameter optimization and generative modeling. However, despite their efficiency, these deep models consist of computationally demanding operations, making them unsuitable for real-time forecasting.

6.2.2.3 Other learning-based approaches

Incorporating an analysis of spatial dependencies has been explored through a different method. ConvLSTM replaces LSTM state-to-state and input-to-state transition with convolutions [Shi et al., 2015, Lin et al., 2020] to bring LSTM the computational capability to analyze spatiotemporal series. Methods relying on attention are able to learn how each data point interacts with each other at each timestep. In particular self-attention has been successfully leveraged with LSTM for diverse forecasting tasks [Li et al., 2020b, Deng et al., 2019, Jing, 2019b], and Transformers [Vaswani et al., 2017b, Wen et al., 2022b] recently showed promising results in data series [Wu et al., 2020]. Some methods leverages both convolution and self-attention to reach state-of-the-art results on some datasets [Lin et al., 2020]. Other methods combine LSTM and graph neural network [Jiang and Luo, 2021, Zhao et al., 2020, Bogaerts et al., 2020] to forecast and quantify how roads and intersections impact one another through graph modelization. However, these methods are by design for macro-scale road systems and are unfit for meso modelizations.

6.3 Traffic forecasting on INRIX data

6.3.1 Data acquisition

This research utilizes mesoscale data obtained from INRIX traffic services [Reed, 2019], which includes average speeds across multiple lanes on 21 segments of the I-24 interstate highway in

Figure 6.5: The Target Road Segment of CIRCLES: I-24 Westbound in Nashville, Tennessee, seen within the highlighted region. Figure from [Chekroun et al., 2024].

Nashville, TN, as depicted in Figure 6.5.

This data spans mileposts MM66 to MM59, covering an 11.4 km road fraction divided into 21 segments, with a sampling rate of 3,600 data points per day. An example of typical morning traffic is shown in Figure 6.6.

While INRIX traffic data updates every minute, there can be a slight lag of up to three minutes in data generation. The focus of this study is to enhance the accuracy and timeliness of traffic forecasting, especially considering the brief delays in data updates. The objective is to develop a predictive model that effectively forecasts traffic patterns in three-minute intervals, leveraging the minute-by-minute data refreshment to anticipate and manage traffic conditions more efficiently.

6.3.2 Modelization as a data series problem

We modeled this data series problem in the following way. At every time-step t, we note v_t^i the average velocity over the lanes on the whole $i \in [0, 20]$ segment. Hence, the studied data series can be seen as follows:

$$\mathrm{V}_t = \begin{bmatrix} \mathrm{v}_t^0 \\ \mathrm{v}_t^1 \\ \vdots \\ \mathrm{v}_t^{20} \end{bmatrix},$$

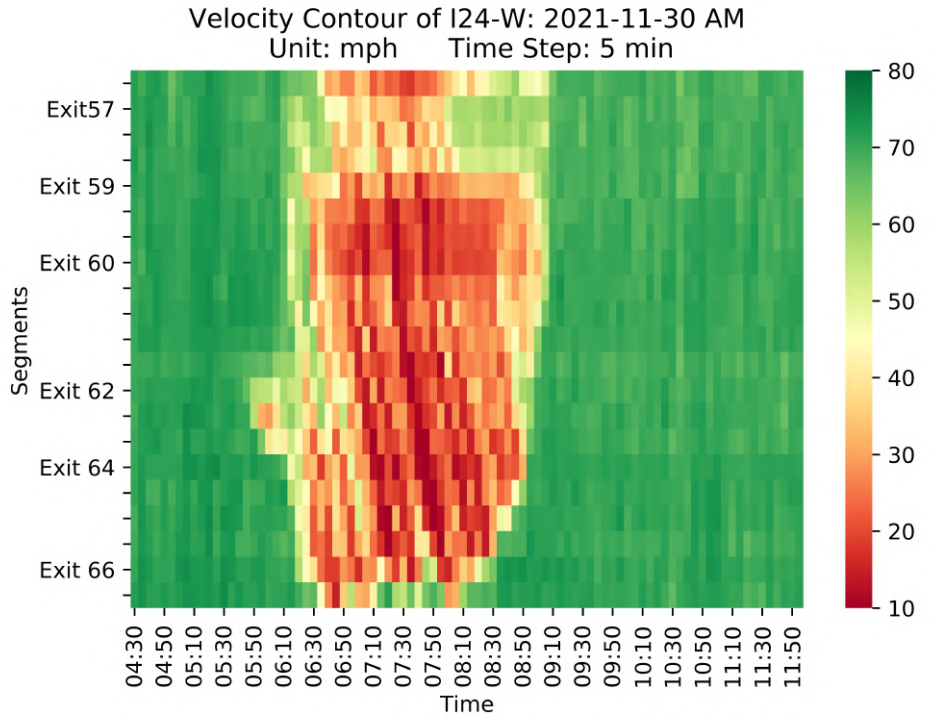


Figure 6.6: In the red contour of the figure, one observes the chronological progression of congestion on the specified segments. A notable persistent bottleneck is evident at Exit 59. This congestion initiates at approximately 6:00 a.m., likely attributable to the augmented commuting demand upstream, and it fully resolves by around 9:00 a.m. Figure and caption from [Chekroun et al., 2024].

We also define

$$\forall (t,k) \in \mathbb{N}^2, \mathrm{I}_t^k = V_t \oplus \cdots \oplus V_{t+k} = \begin{bmatrix} \mathrm{v}_t^0 & \cdots & \mathrm{v}_{t+k}^0 \\ \mathrm{v}_t^1 & \cdots & \mathrm{v}_{t+k}^1 \\ \vdots & & \vdots \\ \mathrm{v}_t^{20} & \cdots & \mathrm{v}_{t+k}^{20} \end{bmatrix}$$

The concatenation of k consecutive velocity vector starting at time t.

Therefore, our final model should be fed with I_{t-s}^s to output I_t^3, s being the chosen sequence length used as input.

6.3.3 Training and validation datasets

The training set is composed of 504,000 data points (every minute for 350 days). We also built two validation sets, also represented in Figure 6.7:

- The Easy Validation set: made of 86,400 data points (every minute for 60 days), it mostly represents common traffic as most of it is smooth, with some discrete congestion and traffic shockwaves. There is usually at least one bigger traffic bottleneck every day between 6 am

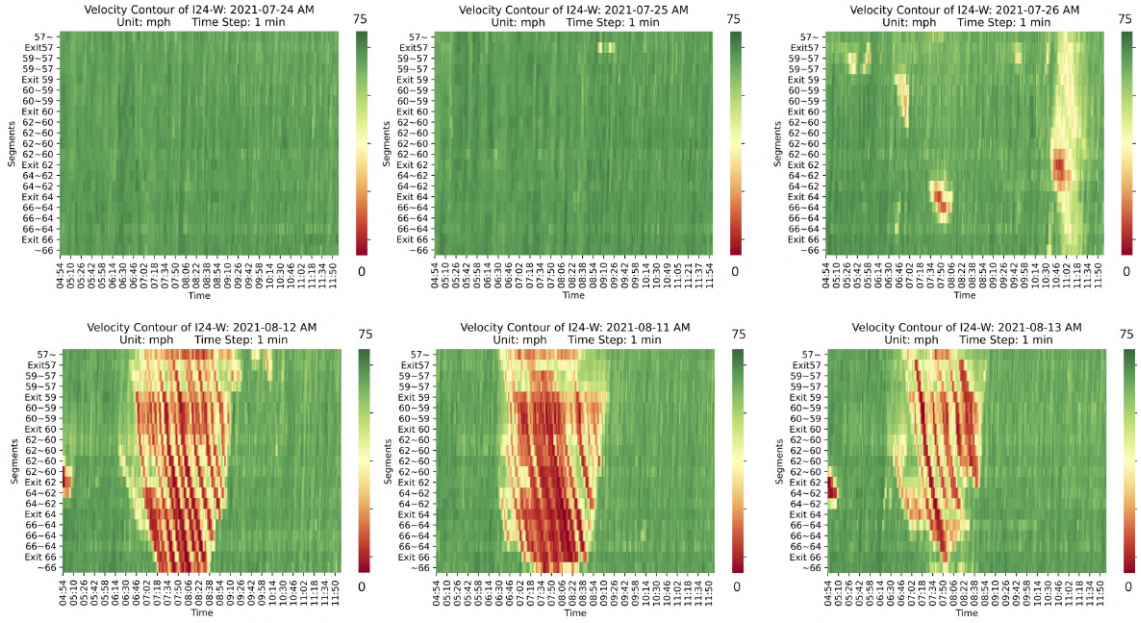


Figure 6.7: Illustrative representation of the validation datasets. **Top Row:** Three representative snapshots from the Easy Validation set, showcasing common traffic patterns with periodic congestion and the prominence of temporal dependencies. **Bottom Row:** Three exemplar visuals from the Hard Validation set, highlighting moments of intense congestion, significant vehicle interactions, and the emphasis on spatial dependencies.Figure and caption from [Chekroun et al., 2024].

and 10 am. Traffic is mostly fluid in this dataset, interactions between vehicles are almost negligible, and metrics mostly represent a model capability to capture temporal dependencies.

- The Hard Validation set: the composition of four 440 minutes of highly congested traffic bottlenecks (1,760 total data points). Metrics are evaluated independently and averaged on those three sets to obtain the Hard metric. Traffic being highly congested, interactions between vehicles are consequent, and validations metrics on this dataset represent a model capability to capture spatial dependencies.

6.4 Real-time mesoscale traffic forecasting

Our primary emphasis is on single-step traffic prediction, which involves forecasting traffic conditions just one minute ahead. Subsequently, we expand our approach to solve the problem of multi-step traffic forecasting, which involves predicting traffic conditions several minutes into the future.

6.4.1 One-minute INRIX prediction

While LSTM is a correct baseline for both accuracy and inference time, they do not qualify as an optimal solution for our data. Indeed, at a given time t, a traffic bottleneck at position k will impact both the short and long-term v_t^k, but also the neighboring segments $v_t^{k-\epsilon}$ and $v_t^{k+\epsilon}$. Hence, studied data presents spatio-temporal relationships, while standard LSTM mostly focuses on temporal relationships. To overcome this limitations, self-attention can be a powerful tool. Indeed, self-attention can intuitively capture the dynamic dependencies between different segments of the road network, recognizing how traffic conditions on one segment affect others. By attending to relevant spatial and temporal patterns, self-attention enables traffic forecasting models to adapt and predict congestion, flow changes, and bottlenecks. This intuitive capacity to capture inter-dependencies makes self-attention a valuable asset in improving the accuracy and reliability of mesoscale traffic forecasting, ultimately contributing to more effective traffic management strategies and reduced congestion. Self-attention is mathematically represented in Equation 3.1.

Therefore, we designed a Self-Attention LSTM (SA-LSTM) whose output gate is augmented with a self-attention layer on the spatial dimension. Our SA-LSTM is represented in Figure 6.8.

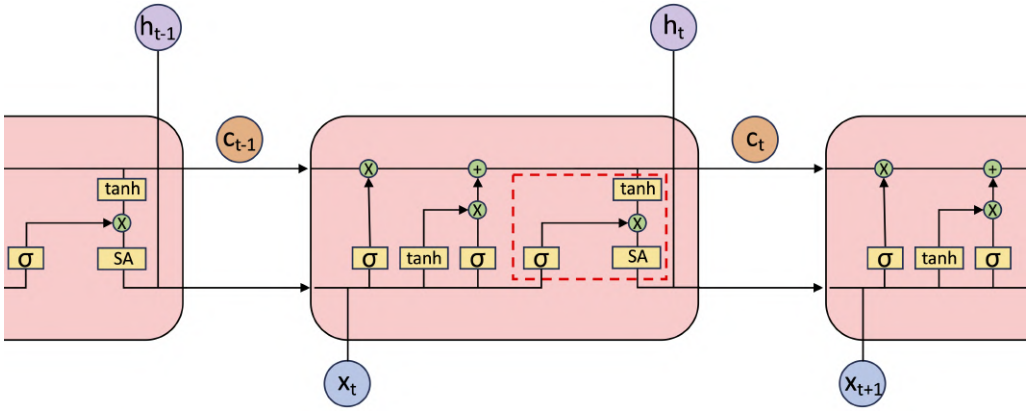


Figure 6.8: A single cell from an SA-LSTM network. The SA-LSTM is an LSTM in which the output gate, in red, is augmented with self-attention.

To train the network to focus on more fine-grained spatial information without further increasing the computational time of operations at inference, we leveraged the Laplacian Pyramid loss [Denton et al., 2015]

$$\mathrm{Lap}_n(x,x') = \sum_{j=0}^{n} 2^{2j} \left| L^j(x) - L^j(x') \right|_1$$

where $L^j(x)$ is the j-th level of the Laplacian pyramid representation of x [Ling and Okada, 2006]. It is a convolution-based loss able to weights the details at fine scales by capturing multi-scale information. The Laplacian Pyramid build process is represented in Figure 6.9. It is used in addition to the MSE loss generally used for LSTMs.

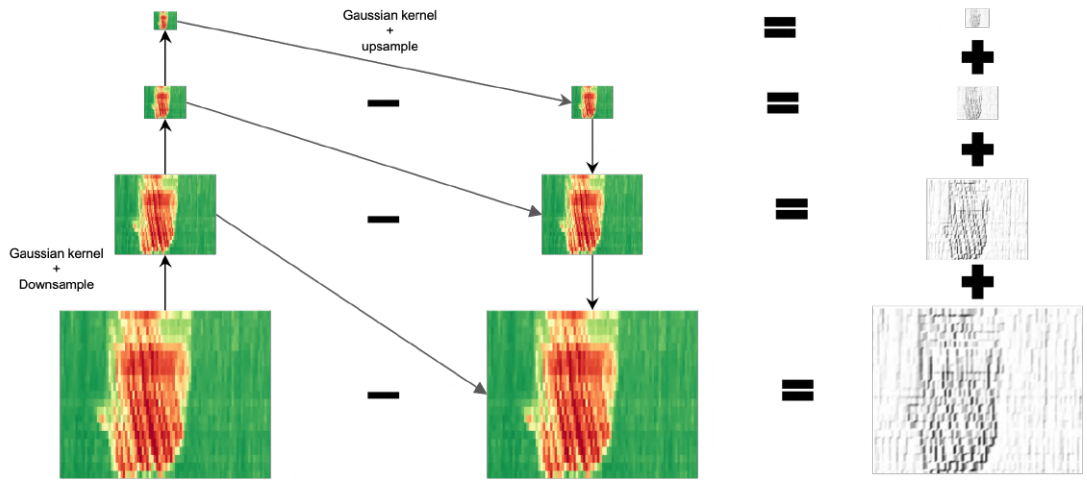


Figure 6.9: Representation of the build process of a Laplacian Pyramid.

6.4.2 *n*-step SA-LSTM

Section 6.4.1 studied one-minute forecasting. In practice however, we aim to forecast up to three-minutes, *i.e.* have access to $V_{t+1}, V_{t+2}, V_{t+3}$. *n*-step forecasting is classically done via recursive inference over the data. Therefore, inferring I_t^3 is made in three successive inferences. First, the network is fed with I_{t-s}^s and outputs $\tilde{v}^t$. Then, the network is fed with $I_{t-s+1}^s \oplus \tilde{v}^t$ to infer $\tilde{v}^{t+1}$, and so on. However, such methods suffer from accumulation error, as inaccuracies in each inference will weigh on the next ones. Also, total inference time is at least n times the inference time of a single network inference. This method is therefore unfit for real-time inference. Another method is the all-at-once technique, in which a single LSTM is fed I_{t-s}^s and trained to output I_t^{t+n}. While significantly faster and offering better long-term forecasting, this method can lead to underperformance on short term forecasting compared to 1-step LSTM which is not desirable in our case of study.

We designed the *n*-step SA-LSTM, a highly supervised multi-layer SA-LSTM represented in Figure 6.10, to take the best of both world: a fast method resilient to accumulation error offering good short term and long term forecasting.

An *n*-step SA-LSTM is a n layers SA-LSTM where:

- Each layer output is constrained via a loss to converge toward V_{t+i};
- Each i-th layer input is the concatenation of the network input $v_{t-k-1}...v_t$ concatenated with previous layer output $(\tilde{v}_{t+j})_{j \leq i}$. Layer i also takes h_{i-1}, c_{i-1} as input.

Therefore, each layer have the same input and output dimensions but contains a different number of cells - which is equal to the input sequence length. Indeed, if input sequence length is 8, first layer will have 8 cells, second layer 8 + 1 cells as we add the previous layer output, and so on. The training of a *n*-step LSTM is sequential layer-wise as each layer is trained independently until convergence.

- Epochs $0 \rightarrow N$: layer_1 is trained, other layers are frozen and loss is only on $\tilde{v}_{t+1}$

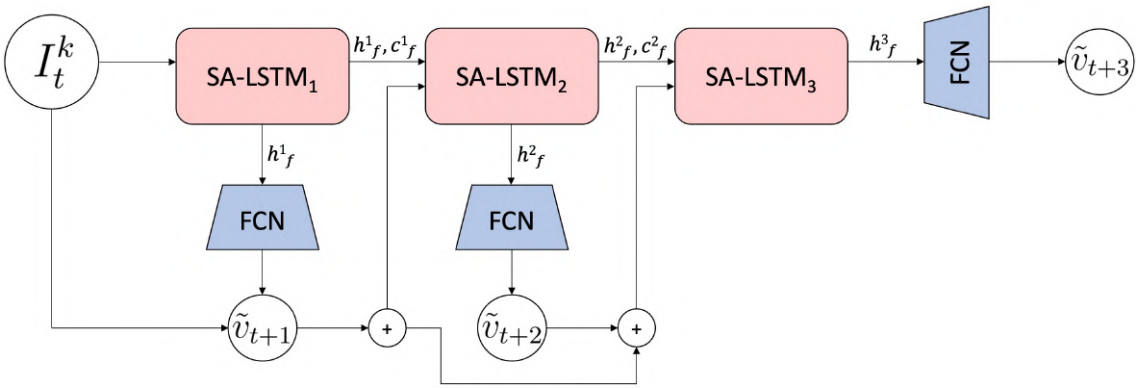


Figure 6.10: Each i-th layer of the 3-step SA-LSTM is trained to infer the forecasting at time $t+i$ through a shared weight fully connected network (FCN). We note h^i_f and c^i_f the outputs of the last cell of the i-th SA-LSTM.

- Epochs $N \longrightarrow 2\times N$: layer$_2$ is trained, other layers are frozen and loss is only on $\tilde{v}_{t+1}$

$\vdots$

- Epochs $(n-1)\times N \longrightarrow n\times N$: all layers are unfrozen and the network is fine-tuned.

6.5 Experimental results

Unless specified otherwise, all presented models have been trained with an AdamW [Loshchilov and Hutter, 2017] optimizer set with a learning rate of 0.01 and a scheduler to make the learning rate decrease by a factor of 10 when validation metrics stagnate or increase over 3 consecutive validations. Training aims to minimize the Mean Square Error (MSE) between the prediction $\tilde{y}$ and the ground truth y *i.e.*, the value $\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$. For ablation studies, training seeds are fixed, and gradient descent is not stochastic: every batch contains the whole dataset and hence is an epoch.

6.5.1 One-minute forecasting

Ablation study over self-attention. Comparison between LSTM and SA-LSTM is presented in Table 6.1. We observe that LSTM and SA-LSTM are on-par on the Easy validation set, and SA-LSTM is significantly better on the Hard dataset. Hence, adding a self-attention layer to an LSTM allows for enhancing the quality of spatial dependencies predictions with no degradation of temporal dependencies.

Ablation study over Laplacian pyramid loss. Experimental results are presented in Table 6.2. Training with this loss gave better results on one-minute predictions, particularly on the Hard dataset with a significant observed improvement. Indeed, this loss allows the model to focus the training on high-frequency details, which are more important in the Hard set. We observed the optimal depth to be 3 for our chosen hyper-parameters setting. Accuracy degrades for deeper

		Validation set		
Method	Self-Attention	Easy	Hard	Time (ms)
LSTM	-	0.66	5.71	0.2
	✓	0.64	**4.53**	0.5

Table 6.1: Ablation study of LSTM and SA-LSTM on INRIX data for traffic forecasting. Metric is MSE scaled by $\times 10^3$. Time is inference time measured as the mean over 50,000 inferences after a warmup of 1,000 inferences.

depth than 3 because of the required pre-processing on tensors used for the Laplacian Pyramid loss during training only: we need the dimension of the inputs of this loss to be a multiple of 2^{depth}, and going to deep adds substantial empty padding. Therefore most of the input becomes 0, which improves training metrics but significantly decreases validation metrics.

		Pyramid depth				
Method	Validation	0	1	2	3	4
SA-LSTM	Easy	0.64	0.64	**0.63**	**0.63**	0.65
	Hard	4.48	4.31	3.94	**3.59**	4.12

Table 6.2: Ablation study of SA-LSTM trained with Laplacian Pyramid Loss using several depths on INRIX data for traffic forecasting. Metric is MSE and scaled by $\times 10^{-3}$. Inference time is unchanged compared to SA-LSTM, as the core network is the same. Next experiments will fix the Laplacian Pyramid loss depth at 3.

Comparison with state-of-the-art methods. To validate our method, we compare inference time and validations accuracy with state-of-the-art spatio-temporal forecasting methods in Table 6.3. ConvLSTM [Shi et al., 2015] is a type of LSTM in which state-to-state and input-to-state transitions are replaced with convolutions. Self-Attention ConvLSTM [Lin et al., 2020] is a ConvLSTM whose transitions have been augmented with self-attention layers. Transformers [Vaswani et al., 2017b] leverage attention to transform an input sequence into an output one by weighting how each elements of the input sequence interact one with each other. Interestingly, convolution-based methods trained with the Lap_3 loss led to a drop in accuracy on the easy validation set, while self-attention only methods experimentally benefit from it. SA-LSTM yields the best metrics on the Hard validation set and is comparable with the best method on the Easy one. Moreover, inference time is significantly lower than other methods designed for spatio-temporal forecasting and stays well under the intended millisecond.

Notably, both ConvLSTM and SA-ConvLSTM results on the Easy dataset degrade when training with a Laplacian Pyramid Loss but improve on the Hard dataset, as seen in the ablation in Table 6.3. More generally, we observed the Laplacian Pyramid Loss to improve all methods on the Hard validation dataset, however, SA-LSTM trained with Laplacian Pyramid Loss still outperforms other variations. Our intuition is that ConvLSTM-based models are by design highly focused on spatial dependencies and less on temporal ones than regular LSTM-based methods.

Method	Lap_3 Loss	Validation Set Easy	Validation Set Hard	Time (ms)
LSTM	✗	0.66	5.71	**0.2**
	✓	**0.61**	4.09	
ConvLSTM	✗	0.63	5.15	3.7
	✓	0.68	5.13	
SA-ConvLSTM	✗	0.72	4.19	4.1
	✓	0.76	3.94	
Transformers	✗	0.65	5.03	1.8
	✓	0.64	4.71	
SA-LSTM	✗	0.64	4.52	0.5
	✓	0.63	**3.58**	

Table 6.3: Comparison of different forecasting methods and ablation study over the Lap_3 loss on INRIX data for traffic forecasting. Metric is MSE scaled by $\times 10^3$.

Training with this loss worsens the spatial-dependency/ temporal-dependency analysis trade-off and over-advantages the analysis of spatial predictions over temporal ones.

Overall, the model offering the best results on both the Easy and Hard datasets, so on temporal-focused and spatial-focused prediction, is the SA-LSTM. An example heatmap prediction from this network and corresponding traffic curve in different scenarios are represented in Figure 6.11 and Figure 6.14, in Section 6.6.

6.5.2 n-step forecasting

We compared different multi step forecasting methods and compared metrics on $t+1$, $t+2$ and $t+3$. We also compare running time, as we want our solution to run under the millisecond.

Method	Validation set	$t+1$	$t+2$	$t+3$	Total time (ms)
Recursive	Easy	**0.63**	0.83	1.24	1.8
	Hard	**3.58**	6.51	10.67	
All-at-once	Easy	0.70	**0.82**	**0.96**	0.5
	Hard	4.31	5.58	**7.43**	
n-step	Easy	**0.63**	0.83	1.03	0.9
	Hard	**3.58**	**5.41**	7.56	

Table 6.4: Comparison of different multi step forecasting methods. Metrics are MSE scaled by 10^3. Underlined running times are the one acceptable for our application case.

The most optimal results for $t+1$ are achieved using the recursive and 3-step methods. This is expected since the LSTM weights dedicated to this inference were trained specifically for 1-step forecasting using real INRIX data. In contrast, the underperforming all-at-once method is

not as finely tuned for 1-step predictions. However, from $t+2$ onwards, the accumulation errors begin to impact the recursive method, which then gets surpassed by both the all-at-once and 3-step approaches, leading to similar performance metrics. This gap becomes even more pronounced at $t+3$, where both the all-at-once and 3-step methods significantly outpace the recursive method. It's worth noting that the 3-step LSTM offers the best overall results for both single-step and multi-step predictions while maintaining sub-millisecond inference times. This highlights the method's proficiency in 1-step forecasting and its resistance to cumulative errors.

For our use case, n-step SA-LSTM appears as the best trade-off between inference time and both single and multi-step inference: $t+1$, $t+2$ and $t+3$ predictions are on-par with our best results overall while being faster than any other forecasting method except Vanilla LSTM. An example heatmap prediction from this method and corresponding traffic curve in different scenarios are represented in Figure 6.12 and Figure 6.15.

6.6 Qualitative observations

This section presents qualitative observations and analysis of some of our results. Presented qualitative representation comes in two forms of different granularity.

6.6.1 Heatmaps

We first present a comparison of ground truth and inferred speed profile plotted has heatmaps for different forecasting methods and for both single step forecasting and multi-step forecasting. These heatmaps bear 3D information and represent mean velocity of all the vehicles on each segment of the studied part of the I-24 highway at each time step. While this kind of representation can give an overall insight on the quality of the inference, it is in practice hardly analyzable with the naked eyes. This subsection present heatmaps for single step Lap_3 SA-LSTM, 3-step Lap_3 SA-LSTM, and for the example of a failing case a 3 minute inference via the recursive method with the Lap_3 SA-LSTM.

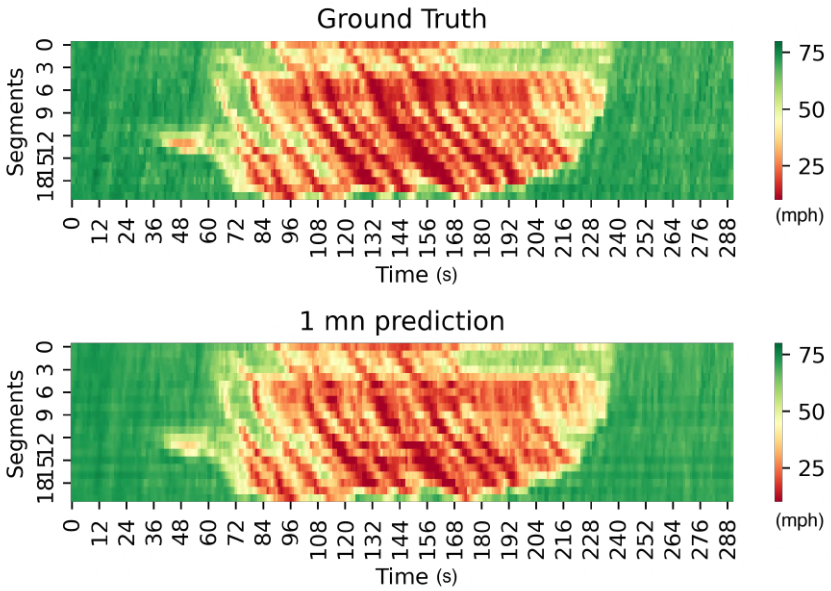


Figure 6.11: Comparison of heatmap generated from Lap_3 SA-LSTM one-minute traffic prediction with an heatmap generated from ground truth data. We observe inference to be as expected in both fluid and congested setup. Speeds are in miles/hour, time is in minute.

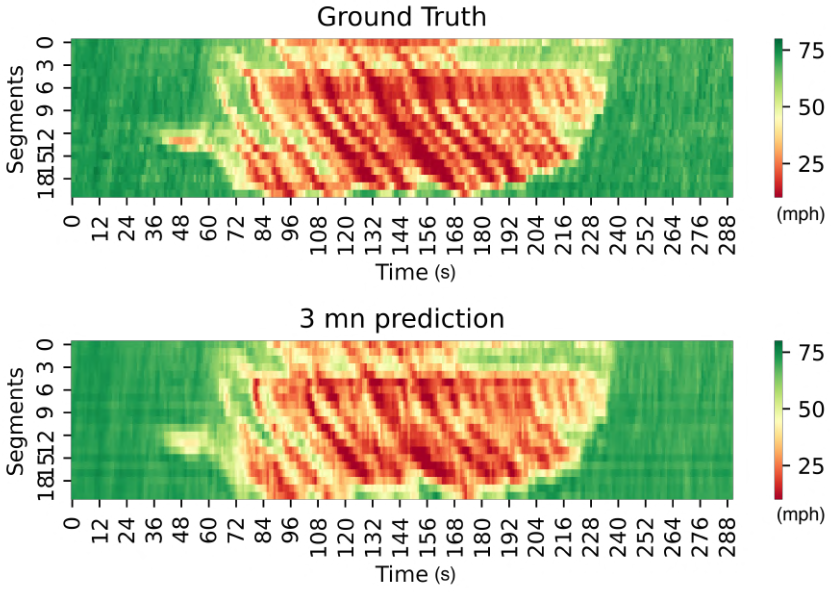


Figure 6.12: Comparison of heatmap generated from 3-step Lap_3 SA-LSTM three-minute traffic prediction with an heatmap generated from ground truth data. We observe inference to be as expected in both fluid and congested setup. Speeds are in miles/hour, time is in minute.

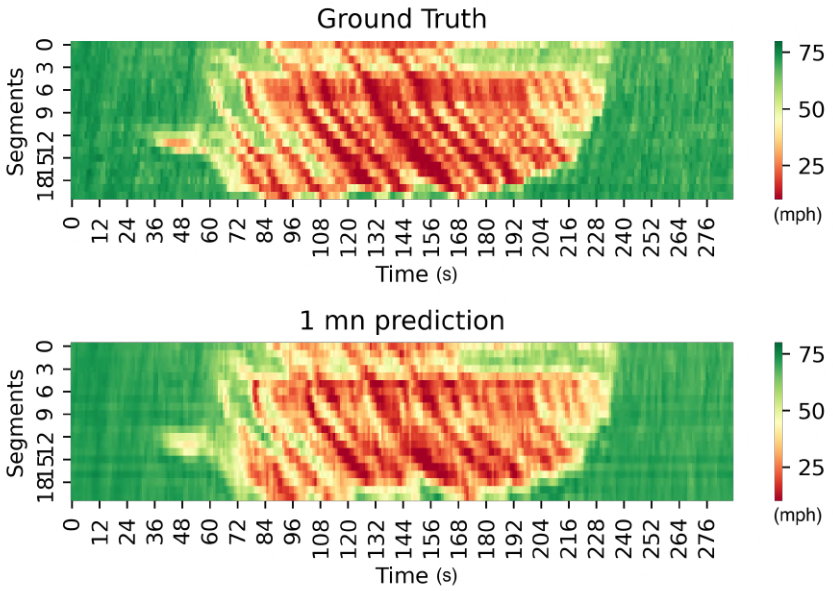
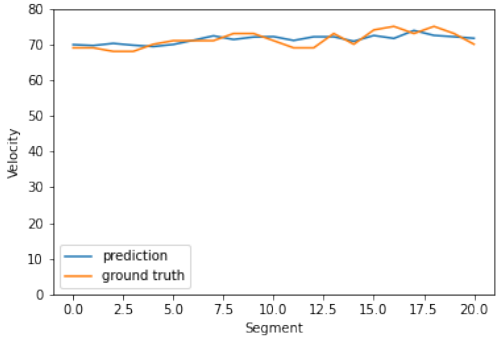


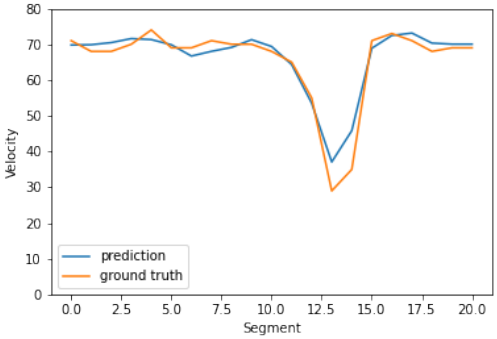
Figure 6.13: Comparison of heatmap generated from recursive Lap_3 SA-LSTM three-minute traffic prediction with a heatmap generated from ground truth data. We observe some blurr in the figure. This is due to the loss of accuracy caused by accumulation error.

6.6.2 Velocity curves in diverse stages of traffic

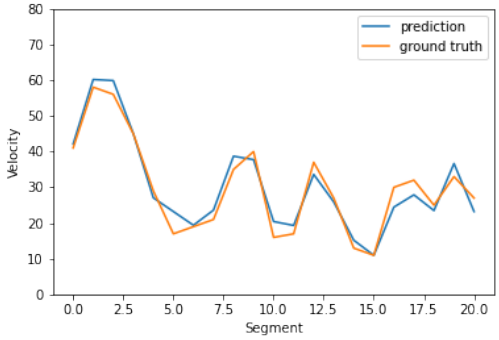
A more granular and easier to analyse type of representation is the plot of velocity curves in different stages of traffic. Contrarily to heatmaps, a velocity curve focuses on a single timestep and represents the relation between segment and mean velocity of the vehicles in it. This subsection present velocity curves in four representative stages of traffic (free flow of timestep 24, bottleneck of time 48, fully congested on time 180, and dissipation stage of timestep 216) for single step Lap_3 SA-LSTM, 3-step Lap_3 SA-LSTM, and for the example of a failing case a 3 minute inference via the recursive method with the Lap_3 SA-LSTM.



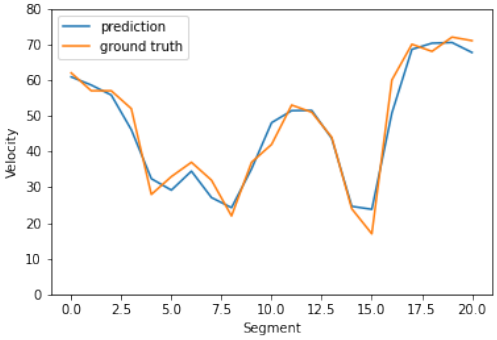
(a) Timestep 24: During the free flow stage, the prediction is able to generate the results around the free flow speed, 70 mile/hr.



(b) Timestep 48: A bottleneck start to form between segments 11 - 15. The prediction presents the same velocity change pattern with an accurate spatial location of the bottleneck as ground truth.

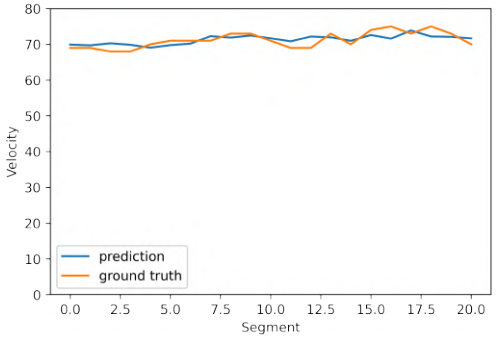


(c) Timestep 180: During the fully congested stage, the model is able to predict the propagation of the upstream shockwaves.

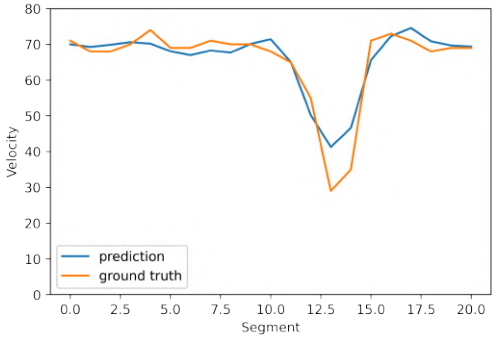


(d) Timestep 216: During the dissipation stage of the congestion, the prediction is able to capture the speed recovery at the bottleneck and upstream.

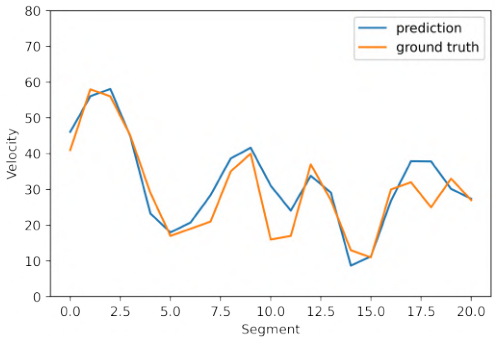
Figure 6.14: Comparison between the ground truth and the one-minute predictions from the Lap_3 SA-LSTM during different stages of the congestion lifecycle. Velocities are in mph.



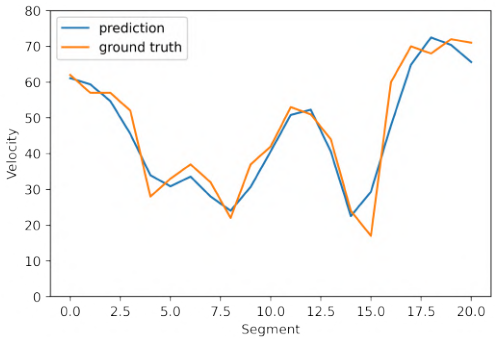
(a) Timestep 24: Prediction of free flow stage align with the ground truth around free flow speed.



(b) Timestep 48: A bottleneck start to form between segments 11 - 15. The prediction presents the same velocity change pattern with an accurate spatial location of the bottleneck as ground truth.

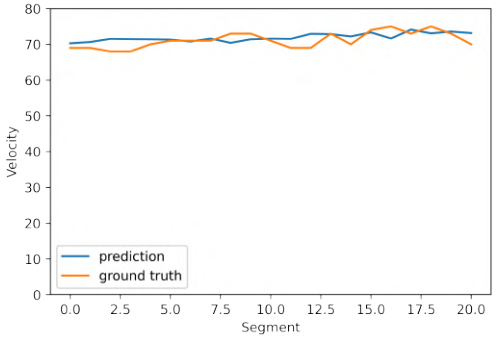


(c) Timestep 180: During the fully congested stage, the prediction captured the pattern of shockwave, while the prediction of absolute speed value has diversion from the ground truth. The predicted locations of the bottleneck and shockwaves are reliable.

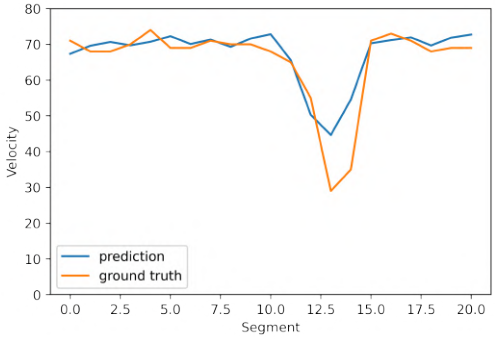


(d) Timestep 216: During the dissipation stage of the congestion, the prediction is able to capture the speed recovery at the bottleneck and upstream.

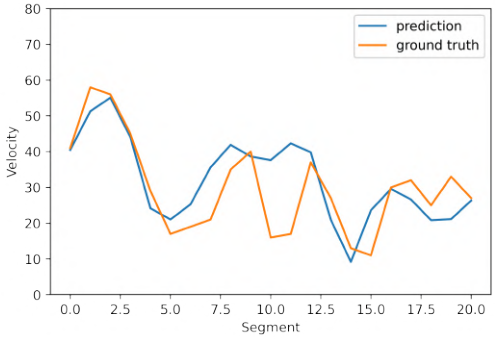
Figure 6.15: Comparison between the ground truth and the three-minute predictions from the 3-step Lap_3 SA-LSTM during different stages of the congestion lifecycle. Velocities are in mph.



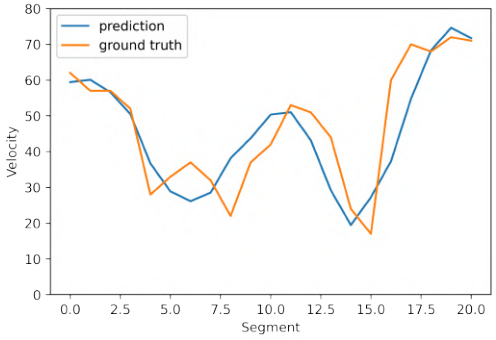
(a) Timestep 24: Prediction of free flow stage align with the ground truth around free flow speed.



(b) Timestep 48: A bottleneck start to form between segments 11 - 15. The prediction presents the same velocity change pattern with an accurate spatial location of the bottleneck as ground truth. The lowest speed at the bottleneck is underestimated.



(c) Timestep 180: During the fully congested stage, the prediction captured the pattern of shockwave, while the prediction of absolute speed value has diversion from the ground truth.



(d) Timestep 216: During the dissipation stage of the congestion, the prediction is able to capture the speed recovery at the bottleneck and upstream.

Figure 6.16: Comparison between the ground truth and the three-minute predictions from the recursive Lap_3 SA-LSTM during different stages of the congestion lifecycle. Velocities are in mph.

6.7 Conclusion

This section introduces the concept of leveraging autonomous driving technologies to generate social benefits, in particular for traffic dissipation. It presents the CIRCLES consortium and the work of this community to build a hierarchical speed planner utilizing diverse methods and technologies from several fields, and the MegaVanderTest, the biggest CAV experiment so far (as of January 2024) where a fleet of 100 vehicles leveraged those technology to gather traffic data and test the designed system. This section focuses in particular on my main contribution of this project, which is the problem of real-time mesoscale traffic forecasting. Hence, we present a fast and accurate method able to extract and analyze both temporal and spatial dependencies of traffic data series. This approach has been analyzed through an extensive ablation study of its components, and compared with state-of-the-art methods for spatio-temporal forecasting to highlight how adapted it is for the studied task. Lastly, we introduced a novel technique for generalization of one-step forecasting method to multi-step forecasting. This method showed to provide the best trade-off inference time on both short-term and long-term forecasting for our considered use case.

Chapter 7

Conclusion

This thesis explored three approaches where expert knowledge, in the form of expert data or supervised network trained on such expert data, are leveraged in a reinforcement learning setup.

Firstly, this dissertation presents the GRI framework and its application to end-to-end autonomous driving with the GRIAD algorithm. GRI is an approach leveraging both expert data and exploration data gathered by an online agent for model-free deep reinforcement learning. GRI lays on the simplifying hypothesis that expert data represents a perfect behavior, and are thus always associated with a constant maximal reward. Data from both sources are sampled indistinguishably by the neural network for the training. GRIAD applied the GRI approach to autonomous driving on the CARLA simulator and trained a reinforcement learning network on the latent space of a pre-trained CNN encoder. Ablation studies and experimental and qualitative analysis of GRI and GRIAD showed the interest of this method. GRIAD won the 2021 CARLA Challenge, and was ranked first on the CARLA leaderboard for around 6 months. It is worth noting that GRIAD has been trained on real data on a Valeo proprietary simulator, was implemented on real vehicles by Valeo engineers and was tested on the road in Créteil, France. However, GRIAD has some limitations, which have been tackled by other research. Adaptive General Reinforced Imitation (AGRI) [Olsen, 2022] aims at generalizing the GRI approach to incorporate an adaptive demonstration on exploration ratio instead of the fixed one. Reinforced InterFuser [Markhus, 2023] proves that GRIAD-like approaches can benefit from sensor fusion approaches and use LiDAR sensors on top of cameras. Nonetheless, while both approaches offer clear and interesting insights on GRIAD limitations they failed to match state-of-the-art performance.

Secondly, this thesis introduces MBAPPE, a method leveraging a supervised neural network trained via IL on expert data to allow and enhance MCTS exploration in a partially-learned environment for mid-to-end autonomous driving on the nuPlan simulator. The supervised neural network guides the MCTS exploration phase by providing a prior on the ego agent trajectory, and predicts trajectories of other agents in the scene to allow a more reliable MCTS simulation phase. Ablation study was conducted to assess the importance of a well designed prior and continuity constraints to limit exploration to comfortable action for the ego agent within the MBAPPE setup. A comparison with state-of-the-art method showed the interest of this approach which can be seen as a fast improvement operator over existing prediction methods. The interpretability and explainability of this approach is also emphasized via qualitative analysis of MBAPPE behavior and thought process on some scenarios. MBAPPE provides state-of-the-art metrics on a nuPlan benchmark, and a preliminary draft of MBAPPE was awarded the Honorable Mention for Innovation Award at the nuPlan challenge 2023. However, MBAPPE still holds some limitations, often related to implementation. In particular, lane width considered by the MCTS during the simulation phase was, at time of publishing and evaluation, not retrieved from the API but were hard coded to the value found in most scenarios instead. This can lead to difficulties in some turning or pickup dropoff scenarios. Moreover, the MCTS output could be leveraged to refine the prior in a continuous learning setup, thus allowing a circle of self-improvement. Also, better results could be achieved with a more complex learned prior inferred for each node [Schrittwieser et al., 2019, Chen et al., 2020b], as well as learning a bootstrapped value network to estimate node expected returns in addition to the current reward, but this would require more network inferences

which severely harmed the execution time in our experiments.

Lastly, this thesis aims at going further with the idea of autonomous driving and utilize it to serve a greater cause than commercial use by introducing a novel approach leveraging CAVs for traffic dissipation in mixed traffic autonomy. It presents the global hierarchical control framework designed by the CIRCLES consortium, composed of decentralized vehicle controllers and a centralized speed planner. In particular, it introduces a real-time mesoscale traffic forecasting method that is leveraged by the RL-based speed planner to generate a target speed for all the connected vehicles in the designated area. Ablation studies are presented to justify the design of the traffic forecasting network. Comparison with state-of-the-art methods shows that this approach outperforms comparable ones for our given task. Finally, single step forecasting is generalized to multi-step forecasting with the novel n-step LSTM approach, demonstrated to offer the best trade-off in accuracy through successive time steps and execution time. However, why experimental results shows our method efficiency for mesoscale traffic forecasting in real time on the INRIX traffic dataset, our validation method fails to demonstrate whether this approach will allow an improved speed planning in the RL loop, on the global system scale. To measure its efficiency on the global pipeline, the developed method is planned on being utilized in the next MegaVanderTest organized by the CIRCLES consortium, which will take place in a couple of years.

Overall, this thesis explored different concepts relative to autonomous driving and tackle them by integrating expertise in reinforcement learning setups via diverse approaches.

Appendix A

Résumé en français

Chapitre 1 : Introduction Ce chapitre introduit la thèse en expliquant le contexte de la conduite autonome et l'importance de l'intégration des connaissances expertes dans les systèmes d'apprentissage par renforcement pour améliorer les performances de ces systèmes. Il décrit également les objectifs de la thèse et donne un aperçu des publications et distinctions obtenues.

Chapitre 2 : Revue de la littérature : des modèles cinématiques à la planification de mouvement non fondée sur l'apprentissage Ce chapitre présente une révue des méthodes existantes de planification de mouvement non fondées sur l'apprentissage. Il explore les modèles fondés sur la cinématique, les interactions entre agents, les approches probabilistes et les modèles hiérarchiques.

Chapitre 3 : Revue de la littérature : des modèles d'apprentissage profond à la planification de mouvement fondée sur l'apprentissage Ce chapitre se concentre sur les approches fondées sur l'apprentissage automatique pour la planification de mouvement. Il aborde les composants des réseaux neuronaux, l'apprentissage par imitation pour la conduite autonome de bout en bout, ainsi que les approches par apprentissage par renforcement pour la planification de mouvement.

Chapitre 4 : Distillation d'expertise dans le RL sans modèle pour la conduite autonome de bout en bout Ce chapitre propose une méthode novatrice pour intégrer des connaissances expertes dans des systèmes d'apprentissage par renforcement sans modèle, appliqués à la conduite autonome de bout en bout. La méthode, appelée GRIAD (General Reinforced Imitation for Autonomous Driving), distille les connaissances expertes dans l'apprentissage d'un agent, en combinant apprentissage par imitation et apprentissage par renforcement. Le chapitre comprend une étude approfondie de l'algorithme dans des environnements de simulation du simulateur Mujoco pour valider l'efficacité de la méthode. Il présente également une évaluation sur le simulateur CARLA, démontrant que l'approche surpassait les méthodes classiques d'apprentissage par renforcement, notamment sur les benchmarks NoCrash et sur le CARLA Leaderboard. Une analyse qualitative des limites de la méthode est également abordée.

Chapitre 5 : Exploitation des connaissances apprises dans le RL fondé sur un modèle pour la conduite autonome de milieu à fin Dans ce chapitre, une approche hybride est développée pour la conduite autonome de milieu à fin. Un réseau neuronal, préalablement entraîné via l'apprentissage par imitation, est utilisé comme guide dans une méthode d'apprentissage par renforcement fondé sur modèle, exploitant l'exploration par recherche d'arbre de Monte-Carlo (MCTS). L'algorithme, appelé MBAPPE (MCTS-Built-Around Predictions for Planning Explicitly), est testé dans l'environnement du simulateur nuPlan. Le chapitre inclut des études d'ablation qui analysent les effets des contraintes de continuité et de l'intégration des priorités apprises. Il met également en avant la capacité du système à fournir une meilleure interprétabilité des décisions prises par l'algorithme, ce qui est crucial pour des applications de conduite autonome sécurisées.

Chapitre 6 : Prévision du trafic pour la dissipation du trafic fondée sur le RL Ce chapitre présente un cadre de contrôle hiérarchique dans lequel un planificateur de vitesse centralisé, entraîné par renforcement, guide une flotte de 100 véhicules autonomes et connectés sur les autoroutes afin de dissiper les embouteillages. Cette recherche, menée par le consortium américain CIRCLES, a conduit à une expérience réelle sur l'autoroute I-24 à Nashville, Tennessee, aux États-Unis. En particulier, ce chapitre développe un module de prévision du trafic à l'échelle mésoscopique en temps réel, qui peut être exploité par le planificateur de vitesse centralisé entraîné par renforcement. Ce module de prévision est capable de prédire l'état du trafic à court terme (une minute) et à moyen terme (plusieurs minutes). Le chapitre présente les résultats expérimentaux de ce module de prédiction de trafic, obtenus à partir de données réelles issues de la base de données INRIX, et démontre son efficacité sur différents types de trafic observés, en le comparant à d'autres méthodes de l'état de l'art en prédiction spatio-temporelle.

Chapitre 7 : Conclusion Le dernier chapitre récapitule les principales contributions de la thèse, les résultats obtenus et propose des perspectives pour des recherches futures dans le domaine de la conduite autonome basée sur l'apprentissage par renforcement.

Appendix B

Bibliography

[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.

[Albeaik et al., 2022] Albeaik, S., Bayen, A., Chiri, M. T., Gong, X., Hayat, A., Kardous, N., Keimer, A., McQuade, S. T., Piccoli, B., and You, Y. (2022). Limitations and improvements of the intelligent driver model (idm). *SIAM Journal on Applied Dynamical Systems*, 21(3):1862–1892.

[Amini et al., 2020] Amini, A., Gilitschenski, I., Phillips, J., Moseyko, J., Banerjee, R., Karaman, S., and Rus, D. (2020). Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150.

[Antonio and Maria-Dolores, 2022] Antonio, G.-P. and Maria-Dolores, C. (2022). Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections. *IEEE Transactions on Vehicular Technology*, 71(7):7033–7043.

[Aydemir et al., 2023] Aydemir, G., Akan, A. K., and Güney, F. (2023). Adapt: Efficient multi-agent trajectory prediction with adaptation. *arXiv preprint arXiv:2307.14187*.

[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.

[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.

[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.

[Bishop, 1995] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

[Bogaerts et al., 2020] Bogaerts, T., Masegosa, A. D., Angarita-Zapata, J. S., Onieva, E., and Hellinckx, P. (2020). A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data. *Transportation Research Part C: Emerging Technologies*, 112:62–77.

[Bojarski et al., 2016] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars. *CoRR*, abs/1604.07316.

[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.

[Caesar et al., 2021] Caesar, H., Kabzan, J., Tan, K. S., Fong, W. K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., and Omari, S. (2021). nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*.

[Cai and Hsu, 2022] Cai, P. and Hsu, D. (2022). Closing the planning–learning loop with application to autonomous driving. *IEEE Transactions on Robotics*, 39(2):998–1011.

[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.

[Carton, 2021] Carton, F. (2021). *Exploration of reinforcement learning algorithms for autonomous vehicle visual perception and control*. PhD thesis, Institut Polytechnique de Paris (IPP).

[Chaslot et al., 2006] Chaslot, G., Uiterwijk, J., Bouzy, B., and Herik, H. (2006). Monte-carlo strategies for computer go. *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence*.

[Chekroun et al., 2023a] Chekroun, R., Gilles, T., Toromanoff, M., Hornauer, S., and Moutarde, F. (2023a). Mbappe: Mcts-built-around prediction for planning explicitly.

[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*.

[Chekroun et al., 2023b] Chekroun, R., Toromanoff, M., Hornauer, S., and Moutarde, F. (2023b). Gri: General reinforced imitation and its application to vision-based autonomous driving. *Robotics*, 12(5).

[Chekroun et al., 2024] Chekroun, R., Wang, H., Lee, J., Toromanoff, M., Delle Monache, M. L., and Moutarde, F. (2024). Mesoscale traffic forecasting for real-time bottleneck and shockwave prediction. *arXiv preprint*.

[Chen et al., 2021a] Chen, D., Koltun, V., and Krähenbühl, P. (2021a). Learning to drive from a world on rails. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15590–15599.

[Chen et al., 2021b] Chen, D., Koltun, V., and Krähenbühl, P. (2021b). Learning to drive from a world on rails. In *ICCV*.

[Chen and Krähenbühl, 2022a] Chen, D. and Krähenbühl, P. (2022a). Learning from all vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17222–17231.

[Chen and Krähenbühl, 2022b] Chen, D. and Krähenbühl, P. (2022b). Learning from all vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17222–17231.

[Chen et al., 2019] Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2019). Learning by cheating. In *Conference on Robot Learning (CoRL)*.

[Chen et al., 2020a] Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2020a). Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR.

[Chen et al., 2020b] Chen, J., Zhang, C., Luo, J., Xie, J., and Wan, Y. (2020b). Driving maneuvers prediction based autonomous driving control by deep monte carlo tree search. *IEEE Transactions on Vehicular Technology*, 69(7):7146–7158.

[Chen et al., 2020c] Chen, J., Zhang, C., Luo, J., Xie, J., and Wan, Y. (2020c). Driving maneuvers prediction based autonomous driving control by deep monte carlo tree search. *IEEE Transactions on Vehicular Technology*, 69(7):7146–7158.

[Chitta et al., 2022] Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., and Geiger, A. (2022). Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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

[Codevilla et al., 2019] Codevilla, F., Santana, E., Lopez, A., and Gaidon, A. (2019). Exploring the Limitations of Behavior Cloning for Autonomous Driving. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9328–9337, Seoul, Korea (South). IEEE.

[Coulom, 2006a] Coulom, R. (2006a). Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

[Coulom, 2006b] Coulom, R. (2006b). Efficient selectivity and backup operators in monte-carlo tree search. volume 4630.

[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.

[Dabney et al., 2018] Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1096–1105. PMLR.

[Dauner et al., 2023] Dauner, D., Hallgarten, M., Geiger, A., and Chitta, K. (2023). Parting with misconceptions about learning-based vehicle motion planning. *arXiv preprint arXiv:2306.07962*.

[Delle Monache et al., 2019] Delle Monache, M. L., Liard, T., Rat, A., Stern, R., Bhadani, R., Seibold, B., Sprinkle, J., Work, D. B., and Piccoli, B. (2019). Feedback control algorithms for the dissipation of traffic waves with autonomous vehicles. *Computational Intelligence and Optimization Methods for Control Engineering*, pages 275–299.

[Deng et al., 2019] Deng, D., Jing, L., Yu, J., and Sun, S. (2019). Sparse self-attention lstm for sentiment lexicon construction. *IEEE/ACM transactions on audio, speech, and language processing*, 27(11):1777–1790.

[Denton et al., 2015] Denton, E. L., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751.

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

[Derrow-Pinion et al., 2021] Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Wiltshire, B., Battaglia, P. W., Gupta, V., Li, A., Xu, Z., Sanchez-Gonzalez, A., Li, Y., and Velickovic, P. (2021). Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information*, CIKM '21, page 3767–3776, New York, NY, USA. Association for Computing Machinery.

[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.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[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].

[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., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[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.

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

[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.

[Fan et al., 2018] Fan, H., Xia, Z., Liu, C., Chen, Y., and Kong, Q. (2018). An auto-tuning framework for autonomous vehicles. *arXiv preprint arXiv:1808.04913*.

[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.

[Fujita et al., 2021] Fujita, Y., Nagarajan, P., Kataoka, T., and Ishikawa, T. (2021). Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 22(77):1–14.

[Galceran et al., 2015] Galceran, E., Cunningham, A. G., Eustice, R. M., and Olson, E. (2015). Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *Robotics: Science and Systems*, volume 1, page 6.

[Gao and Wu, 2021] Gao, Y. and Wu, L. (2021). Efficiently mastering the game of nogo with deep reinforcement learning supported by domain knowledge. *Electronics*, 10(13).

[Gao et al., 2018] Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., and Darrell, T. (2018). Reinforcement learning from imperfect demonstrations. *CoRR*, abs/1802.05313.

[Gloudemans et al., 2023] Gloudemans, D., Wang, Y., Ji, J., Zachar, G., Barbour, W., Hall, E., Cebelak, M., Smith, L., and Work, D. B. (2023). I-24 motion: An instrument for freeway traffic science. *Transportation Research Part C: Emerging Technologies*, 155:104311.

[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.

[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.

[Gulino et al., 2023] Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., et al. (2023). Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *arXiv preprint arXiv:2310.08710*.

[Guo et al., 2020] Guo, K., Hu, Y., Qian, Z., Sun, Y., Gao, J., and Yin, B. (2020). Dynamic graph convolution network for traffic forecasting based on latent network of laplace matrix estimation. *IEEE Transactions on Intelligent Transportation Systems*. IF: 3.

[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.

[H. Caesar, 2021] H. Caesar, J. Kabzan, K. T. e. a. (2021). Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*.

[Ha et al., 2020] Ha, T., Cho, K., Cha, G., Lee, K., and Oh, S. (2020). Vehicle control with prediction model based monte-carlo tree search. In *2020 17th International Conference on Ubiquitous Robots (UR)*, pages 303–308.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

[Hallgarten et al., 2023] Hallgarten, M., Stoll, M., and Zell, A. (2023). From prediction to planning with goal conditioned lane graph traversals. *arXiv preprint arXiv:2302.07753*.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Hayat, 2024] Hayat, A. a. (2024). Traffic smoothing with a single vehicle: Dissipating stop-and-go waves with a single controlled vehicle in dense traffic: Experimental evidence. *IEEE Control Systems Magazine*.

[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.

[Henderson, 1974] Henderson, L. (1974). On the fluid mechanics of human crowd motion. *Transportation Research*, 8(6):509–515.

[Hessel et al., 2018] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[Hessel et al., 2017] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298.

[Hester et al., 2018] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. (2018). Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

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

[Hochreiter and Schmidhuber, 1997b] Hochreiter, S. and Schmidhuber, J. (1997b). Long short-term memory. *Neural computation*, 9:1735–80.

[Hoel et al., 2019] Hoel, C.-J., Driggs-Campbell, K., Wolff, K., Laine, L., and Kochenderfer, M. J. (2019). Combining planning and deep reinforcement learning in tactical decision making for autonomous driving. *IEEE transactions on intelligent vehicles*, 5(2):294–305.

[Horgan et al., 2018] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

[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*.

[Hu et al., 2022] Hu, H., Liu, Z., Chitlangia, S., Agnihotri, A., and Zhao, D. (2022). Investigating the impact of multi-lidar placement on object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2550–2559.

[Huang et al., 2023] Huang, Z., Liu, H., and Lv, C. (2023). Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. *arXiv preprint arXiv:2303.05760*.

[Hudson and Zitnick, 2021] Hudson, D. A. and Zitnick, L. (2021). Generative adversarial transformers. In *International conference on machine learning*, pages 4487–4499. PMLR.

[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.

[Jang et al., 2024] Jang, K., Lichtlé, N., Vinitsky, E., Shay, A., Bunting, M., Nice, M., Piccoli, B., Seibold, B., Work, D. B., Delle Monache, M. L., Sprinkle, J., Lee, J. W., and Bayen, A. M. (2024). Large-scale wave dampening via single-agent reinforcement learning algorithms: Controlling and deploying 100 vehicles on a highway with learning-based algorithms. *IEEE Control Systems Magazine*.

[Jiang and Luo, 2021] Jiang, W. and Luo, J. (2021). Graph neural network for traffic forecasting: A survey. *CoRR*, abs/2101.11174.

[Jing, 2019a] Jing, R. (2019a). A self-attention based lstm network for text classification. In *Journal of Physics: Conference Series*, volume 1207, page 012008. IOP Publishing.

[Jing, 2019b] Jing, R. (2019b). A self-attention based lstm network for text classification. In *Journal of Physics: Conference Series*, volume 1207, page 012008. IOP Publishing.

[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.

[Kendall et al., 2019] Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.-M., Lam, V.-D., Bewley, A., and Shah, A. (2019). Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE.

[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.

[Knox et al., 2023] Knox, W. B., Allievi, A., Banzhaf, H., Schmitt, F., and Stone, P. (2023). Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

[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.

[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.

[Laña et al., 2019] Laña, I., Lobo, J. L., Capecci, E., Del Ser, J., and Kasabov, N. (2019). Adaptive long-term traffic state estimation with evolving spiking neural networks. *Transportation Research Part C: Emerging Technologies*. IF: 3.

[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.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Lecun et al., 2004] Lecun, Y., Cosatto, E., Ben, J., Muller, U., and Flepp, B. (2004). Dave: Autonomous off-road vehicle control using end-to-end learning. Technical Report DARPA-IPTO Final Report, Courant Institute/CBLL, http://www.cs.nyu.edu/˜yann/research/dave/index.html.

[Lee et al., 2024] Lee, J. W., Wang, H., Jang, K., Hayat, A., Bunting, M., Alanqary, A., Barbour, W., Fu, Z., Gong, X., Gunter, G., Hornstein, S., Kreidieh, A. R., Lictlé, N., Nice, M., Richardson, W. A., Shah, A., Vinitsky, E., Wu, F., Shengquan, X., Almatrudi, S., Althukair, F., Bhadani, R., Carpio, J., Chekroun, R., Cheng, E., Chiri, M. T., Chou, F.-C., DeLorenzo, R., Gibson, M., Gloudemans, D., Gollakota, A., Ji, J., Keimer, A., Khoudari, N., Mahmood, M., Mahmood, M., Matin, H. N. Z., McQuade, S., Ramadan, R., Urieli, D., Wang, Y., Xu, R., Yao, M., You, Y., Zachar, G., Zhao, Y., Baig, M. N., Bhaskaran, S., Butts, K., Gowda, M., Lee, J., Pedersen, L., Zhang, Z., Zhou, C., Work, D. B., Seibold, B., Sprinkle, J., Piccoli, B., Delle Monache, M. L., and Bayen, A. M. (2024). Traffic smoothing via connected & automated vehicles: A modular, hierarchical control design deployed in a 100-cav flow smoothing experiment. *IEEE Control Systems Magazine*.

[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.

[Li et al., 2020a] Li, G., Li, S., Li, S., Qin, Y., Cao, D., Qu, X., and Cheng, B. (2020a). Deep reinforcement learning enabled decision-making for autonomous driving at intersections. *Automotive Innovation*, 3.

[Li et al., 2018] Li, G., Mueller, M., Casser, V., Smith, N., Michels, D. L., and Ghanem, B. (2018). Oil: Observational imitation learning. *arXiv preprint arXiv:1803.01129*.

[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.

[Li et al., 2020b] Li, W., Qi, F., Tang, M., and Yu, Z. (2020b). Bidirectional lstm with self-attention mechanism and multi-channel features for sentiment classification. *Neurocomputing*, 387:63–77.

[Liang et al., 2018a] Liang, X., Wang, T., Yang, L., and Xing, E. (2018a). Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European conference on computer vision (ECCV)*, pages 584–599.

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

[Lillicrap et al., 2016] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[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.

[Lin et al., 2020] Lin, Z., Li, M., Zheng, Z., Cheng, Y., and Yuan, C. (2020). Self-attention convlstm for spatiotemporal prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11531–11538. AAAI Press.

[Ling and Okada, 2006] Ling, H. and Okada, K. (2006). Diffusion distance for histogram comparison. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 246–253.

[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.

[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.

[Loshchilov and Hutter, 2017] Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

[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., 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.

[Ma et al., 2019] Ma, X., Luan, S., Ding, C., Liu, H., and Wang, Y. (2019). Spatial interpolation of missing annual average daily traffic data using copula-based model. *IEEE Intelligent Transportation Systems Magazine*. IF: 3.

[Mallick et al., 2022] Mallick, T., Balaprakash, P., and Macfarlane, J. (2022). Deep-ensemble-based uncertainty quantification in spatiotemporal graph neural networks for traffic forecasting.

[Markhus, 2023] Markhus, H. S. (2023). Reinforced interfuser for end-to-end autonomous driving in simulated environments. Master's thesis, NTNU.

[Martin et al., 2021] Martin, J. B., Chekroun, R., and Moutarde, F. (2021). Learning from demonstrations with sacr2: Soft actor-critic with reward relabeling. *arXiv preprint arXiv:2110.14464*.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[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.

[Müller et al., 2018] Müller, M., Dosovitskiy, A., Ghanem, B., and Koltun, V. (2018). Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*.

[Ohn-Bar et al., 2020] Ohn-Bar, E., Prakash, A., Behl, A., Chitta, K., and Geiger, A. (2020). Learning situational driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11296–11305.

[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.

[Olsen, 2022] Olsen, A. S. (2022). Adaptive general reinforced imitation in autonomous driving. Master's thesis, NTNU.

[Pan et al., 2017] Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E., and Boots, B. (2017). Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*.

[Pomerleau, 1988] Pomerleau, D. A. (1988). Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1.

[Portugais and Khanal, 2014] Portugais, B. and Khanal, M. (2014). Adaptive traffic speed estimation.

[Prakash et al., 2021] Prakash, A., Chitta, K., and Geiger, A. (2021). Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). 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.

[Qi et al., 2017b] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017b). 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.

[Rajeswaran et al., 2017] Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. (2017). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

[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.

[Reddy et al., 2019a] Reddy, S., Dragan, A. D., and Levine, S. (2019a). SQIL: imitation learning via regularized behavioral cloning. *CoRR*, abs/1905.11108.

[Reddy et al., 2019b] Reddy, S., Dragan, A. D., and Levine, S. (2019b). Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*.

[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.

[Reed, 2019] Reed, T. (2019). Inrix global traffic scorecard.

[Ren et al., 2010] Ren, S., Bi, J., Fung, Y. F., Li, X. I., and Ho, T. K. (2010). Freeway traffic estimation in beijing based on particle filter. In *2010 Sixth International Conference on Natural Computation*.

[Renz et al., 2023] Renz, K., Chitta, K., Mercea, O.-B., Koepke, A. S., Akata, Z., and Geiger, A. (2023). Plant: Explainable planning transformers via object-level representations. In *Conference on Robot Learning*, pages 459–470. PMLR.

[Ross et al., 2010] Ross, S., Gordon, G. J., and Bagnell, J. A. (2010). No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686.

[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.

[Sadat et al., 2019] Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.-C., Yumer, E., and Urtasun, R. (2019). Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956. IEEE.

[Salles et al., 2022] Salles, D., Kaufmann, S., and Reuss, H.-C. (2022). Extending the intelligent driver model in sumo and verifying the drive off trajectories with aerial measurements. *SUMO Conference Proceedings*.

[Scheel et al., 2022] Scheel, O., Bergamini, L., Wolczyk, M., Osiński, B., and Ondruska, P. (2022). Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, pages 718–728. PMLR.

[Schrittwieser et al., 2020] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

[Schrittwieser et al., 2019] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. (2019). Mastering atari, go, chess and shogi by planning with a learned model. *CoRR*, abs/1911.08265.

[Schulman et al., 2017a] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017a). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[Schulman et al., 2017b] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[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.

[Shao et al., 2023a] Shao, H., Wang, L., Chen, R., Li, H., and Liu, Y. (2023a). Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on Robot Learning*, pages 726–737. PMLR.

[Shao et al., 2023b] Shao, H., Wang, L., Chen, R., Waslander, S. L., Li, H., and Liu, Y. (2023b). Reasonnet: End-to-end driving with temporal and global reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13723–13733.

[Shao et al., 2023c] Shao, H., Wang, L., Chen, R., Waslander, S. L., Li, H., and Liu, Y. (2023c). Reasonnet: End-to-end driving with temporal and global reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13723–13733.

[Shi et al., 2015] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

[Silver et al., 2017] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

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

[Sobh et al., 2018] Sobh, I., Amin, L., Abdelkarim, S., Elmadawy, K., Saeed, M., Abdeltawab, O., Gamal, M., and El Sallab, A. (2018). End-to-end multi-modal sensors fusion system for urban automated driving.

[Stern et al., 2018] Stern, R. E., Cui, S., Delle Monache, M. L., Bhadani, R., Bunting, M., Churchill, M., Hamilton, N., Pohlmann, H., Wu, F., Piccoli, B., et al. (2018). Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, 89:205–221.

[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.

[Sunberg and Kochenderfer, 2022] Sunberg, Z. and Kochenderfer, M. J. (2022). Improving automated driving through pomdp planning with human internal states. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20073–20083.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

[Tan and Le, 2019] Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

[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.

[Toromanoff et al., 2019a] Toromanoff, M., Wirbel, É., and Moutarde, F. (2019a). End-to-end model-free reinforcement learning for urban driving using implicit affordances. *CoRR*, abs/1911.10868.

[Toromanoff et al., 2019b] Toromanoff, M., Wirbel, E., and Moutarde, F. (2019b). Is Deep Reinforcement Learning Really Superhuman on Atari? In *Deep Reinforcement Learning Workshop of 39th Conference on Neural Information Processing Systems (Neurips'2019)*, Vancouver, Canada.

[Toromanoff et al., 2020a] Toromanoff, M., Wirbel, E., and Moutarde, F. (2020a). End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7153–7162.

[Toromanoff et al., 2020b] Toromanoff, M., Wirbel, E., and Moutarde, F. (2020b). End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

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

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

[Vaswani et al., 2017b] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Velodyne, 2018] Velodyne, L. (2018). Velodyne alpha prime. `https://velodynelidar.com/blog/guide-to-lidar-wavelengths/`velodynelidar.com [Online; posted 06-November-2018].

[Wang et al., 2024] Wang, H., Fu, Z., Lee, J., Zinat Matin, H. N., Alanqary, A., Urieli, D., Hornstein, S., Rahman Kreidieh, A., Chekroun, R., Barbour, W., Richardson, W. A., Work, D., Piccoli, B., Seibold, B., Sprinkle, J., Bayen, A. M., and Delle Monache, M. L. (2024). Centralized av speed planner: Hierarchical framework for lagrangian variable speed limit in mixed autonomy traffic. *IEEE Control Systems Magazine*.

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

[Wen and Gong, 2023] Wen, Q. and Gong, Z. (2023). Monte-carlo tree search for behavior planning in autonomous driving.

[Wen et al., 2022a] Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. (2022a). Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

[Wen et al., 2022b] Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. (2022b). Transformers in time series: A survey.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

[Wu et al., 2020] Wu, N., Green, B., Ben, X., and O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *CoRR*, abs/2001.08317.

[Wu et al., 2022] Wu, P., Jia, X., Chen, L., Yan, J., Li, H., and Qiao, Y. (2022). Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132.

[Xu et al., 2018] Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., and Savarese, S. (2018). Neural Task Programming: Learning to Generalize Across Hierarchical Tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3795–3802, Brisbane, QLD. 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., 2015] Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., and Courville, A. (2015). Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*, pages 4507–4515.

[Yoshimoto et al., 2006] Yoshimoto, H., Yoshizoe, K., Kaneko, T., Kishimoto, A., and Taura, K. (2006). Monte carlo go has a way to go. In *AAAI*, volume 6, pages 1070–1075.

[Yuan et al., 2015] Yuan, Y., Scholten, F., and van Lint, J. W. C. (2015). Efficient traffic state estimation and prediction based on the ensemble kalman filter with a fast implementation and localized deterministic scheme. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*.

[Zeng et al., 2020] Zeng, W., Wang, S., Liao, R., Chen, Y., Yang, B., and Urtasun, R. (2020). Ds-dnet: Deep structured self-driving network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 156–172. Springer.

[Zhang et al., 2022] Zhang, C., Guo, R., Zeng, W., Xiong, Y., Dai, B., Hu, R., Ren, M., and Urtasun, R. (2022). Rethinking closed-loop training for autonomous driving. In *European Conference on Computer Vision*, pages 264–282. Springer.

[Zhang and Cho, 2016] Zhang, J. and Cho, K. (2016). Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*.

[Zhang et al., 2021] Zhang, Z., Liniger, A., Dai, D., Yu, F., and Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[Zhao et al., 2020] Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., and Li, H. (2020). T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858.

[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

Deux décennies après le premier défi de conduite autonome, qui n'a vu aucun gagnant réussir à naviguer 240 kilomètres de route désertique dans le désert de Mojave, les évolutions en apprentissage automatique ont permis d'importants progrès dans ce domaine. En particulier, la création de simulateurs open-source a facilité la recherche en matière de conduite autonome en permettant d'outrepasser les contraintes réglementaires et en offrant un moyen abordable de collecter des données. Cela, combiné à la montée des réseaux de neurones, a accéléré le développement de méthodes de plus en plus efficaces. Les recherches récentes en matière de planification de mouvement se concentrent principalement sur l'apprentissage par imitation et, dans une moindre mesure, sur l'apprentissage par renforcement. En apprenant à partir de données, les méthodes d'apprentissage automatique sont plus adaptables que celles basées sur des systèmes de règles, car elles dépendent moins d'une représentation parfaite et cohérente de l'environnement. Néanmoins, les approches par imitation restent limitées dans la compréhension des conséquences à long terme de leurs actions et rencontrent des problèmes de robustesse résultant d'une inadéquation de distribution. En revanche, l'approche par renforcement intègre des informations de retour à long terme et surmonte avec succès les problèmes de distribution en apprenant par essais et erreurs. Cependant, cette approche souffre d'inefficacité d'échantillonnage, d'instabilité pendant l'entraînement et d'un manque de garanties de convergence. Cette thèse vise à synergiser les points forts des deux approches tout en atténuant leurs faiblesses en intégrant des connaissances expertes avec des méthodes d'apprentissage par renforcement profond pour différentes applications liées à la conduite autonome.

MOTS CLÉS

Voiture autonome, Apprentissage par renforcement, Connaissance Experte

ABSTRACT

Two decades after the first autonomous driving challenge, which had no winners successfully navigating a 240 kilometers desert road in Mojave, the advancement of machine learning has brought remarkable progress to this field. Notably, the creation of open-source simulators made research for autonomous driving easier by sidestepping regulatory constraints and providing an affordable way to collect data. This, combined with the rise of neural networks, has expedited development of increasingly efficient methods. Recent research for motion planning mostly focuses on imitation learning (IL) and, to a lesser extent, on reinforcement learning (RL). By learning from data, machine-learning based methods are more adaptable than rule-based ones as they rely less on perfect and consistent representation of the environment. Nevertheless, IL approaches remain limited in grasping the long term consequences of their actions and suffer robustness issues stemming from distribution mismatch. Conversely, RL incorporates long-term return information and successfully overcomes distribution mismatch by learning through trial-and-error. However, it suffers from sample inefficiency, instability during training, and lacks of convergence guarantees. This thesis aims to synergize the strengths of both approaches while mitigating their weaknesses by integrating expert knowledge with deep reinforcement learning methods for different autonomous driving applications.

KEYWORDS

Autonomous driving, Reinforcement learning, Expert knowledge