Using context-cues and interaction for traffic-agent trajectory prediction

Vyshakh Palli Thazha

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAE001

Thèse de doctorat

Using Context-Cues and Interaction for Traffic-Agent Trajectory Prediction

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à École Nationale de Techniques Avancées (ENSTA) Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique, Données, IA

Thèse présentée et soutenue à Palaiseau, le 5 Janvier 2022, par

VYSHAKH PALLI THAZHA

Composition du Jury :

Alexander Gepperth Professeur, University of Applied Sciences Fulda, Germany	Président du jury
Véronique Cherfaoui Professeur, Université de Technologie de Compiègne (Heudyasic)	Rapporteur
Anne Verroust-Blondet Enseignant chercheur, Inria (RITS)	Rapporteur
Dominique Vaufreydaz Maître de Conférences, Université Grenoble Alpes (LIG Lab)	Examinateur
David Filliat Professeur, ENSTA Paris (U2IS)	Directeur de thèse
Javier Ibañez-Guzmán Expert Autonomous Systems, Renault Group	Co-directeur de thèse

Abstract

Autonomous vehicle navigation in urban areas involves interactions with the different road-users or traffic-agents like cars, bicycles, and pedestrians, sharing the same road network. The ability of autonomous vehicle to observe, understand and predict the behaviour of these traffic-agents is very important to gain a good situation understanding prior to deciding what manoeuvre to follow. While this is achieved to various degrees of success using model-based or data-driven methods, human drivers remain much more efficient at this task, instinctively inferring different agent motions even in previously unseen and challenging situations. Moreover, context plays a very important role that enables us humans to understand what is being perceived and make finer predictions. The need to increase situational awareness of autonomous vehicles, as well as for safety related driving assistance functions, stimulates our goal to exploit contextual information to predict the future trajectories of the observed traffic-agents in different conditions.

Over the past years, machine learning has proven to be efficient at solving a wide variety of problems, particularly those associated to machine perception. This thesis therefore focuses on developing machine learning models to exploit contextual information in order to observe and learn the trajectories of different interacting traffic-agents as perceived from an autonomous vehicle. While most models proposed in the past rely on a single sensor and model-based techniques, the current approaches often rely on the use of multiple sensors and process their outputs using different machine learning methods. The approach proposed in this thesis follows these trends by combining information from different sensors to predict the trajectories of the observed traffic-agents using machine learning, as well as integrating contextual information and interactions into the prediction

process.

The thesis gradually builds a machine learning architecture based on a theoretical formulation and experimentation. Our approach is based on an LSTM encoder-decoder model that accepts data from different inputs. Trajectory observations from 3D LiDAR point-cloud data and semantic information from map-masks are used. Map masks represent areas where the traffic-agents can operate or not, in a binary manner. The information on pedestrian attention to oncoming vehicles obtained from camera images is also exploited to enrich the sequence prediction system. The goal is to feed the model with context-cues and semantic information to enhance the prediction of the traffic-agent trajectories, by knowing whether or not the agents are aware of the presence of the subject vehicle and including knowledge on areas where they are likely to navigate. Moreover, interactions of the autonomous vehicle with traffic-agents often govern its behaviour as the vehicle navigates. A mechanism to incorporate this information to the machine learning model is also developed as an interaction-aware trajectory prediction system enhanced by context-cues.

Machine learning architectures are built using datasets acquired from the perception sensors of a vehicle navigating in the expected workspace. As datasets play an important role in solving machine learning problems, available annotated datasets for autonomous navigation were reviewed according to their availability of sensor data and contextual information. Experiments were performed for our models to learn, and gradually build the resulting architecture. Their performance are demonstrated using the well-known NuScenes dataset acquired in urban settings. The performance of the proposed approach were compared with model and data-driven approaches, demonstrating that the incorporation of multiple contextual information and agent interactions provides a substantial performance increase.

Résumé

La navigation autonome des véhicules dans les zones urbaines implique des interactions avec les différents usagers de la route ou agents de la circulation partageant le même réseau routier comme les voitures, les vélos et les piétons. La capacité du véhicule autonome à observer, comprendre et prédire le comportement de ces agents est très importante pour acquérir une bonne compréhension de la situation avant de décider de la manœuvre à suivre. Bien que cela soit réalisé à divers degrés de succès en utilisant des méthodes basées sur des modèles ou des données, les conducteurs humains restent beaucoup plus efficaces dans cette tâche, déduisant instinctivement différents mouvements d'agent même dans des situations inédites et difficiles. De plus, le contexte joue un rôle très important qui permet à nous les humains de comprendre ce qui est perçu et de faire des prédictions plus fines. La nécessité d'accroître la connaissance de la situation des véhicules autonomes, ainsi que des fonctions d'aide à la conduite liées à la sécurité, stimule notre objectif d'exploiter ces informations contextuelles pour prédire les trajectoires futures des agents observés dans différentes conditions.

Au cours des dernières années, l'apprentissage automatique s'est avéré efficace pour résoudre une grande variété de problèmes, en particulier ceux associés à la perception. Cette thèse se concentre donc sur le développement de modèles d'apprentissage automatique pour exploiter des informations contextuelles afin d'observer et d'apprendre les trajectoires de différents agents en interaction. Alors que la plupart des modèles proposés dans le passé reposent sur un seul capteur et des techniques basées sur un modèle, les approches actuelles reposent souvent sur l'utilisation de plusieurs capteurs et traitent leurs sorties à l'aide de différentes méthodes d'apprentissage automatique. L'approche proposée dans cette thèse suit

ces tendances en combinant les informations de différents capteurs pour prédire les trajectoires des agents observés à l'aide de l'apprentissage automatique, ainsi qu'en intégrant des informations contextuelles et des interactions dans le processus de prédiction.

La thèse construit progressivement une architecture d'apprentissage automatique basée sur une formulation théorique et des expérimentations. Notre approche est basée sur un modèle d'encodeur-décodeur LSTM qui accepte les données de différentes entrées. Des observations de trajectoire à partir de données de nuages de points LiDAR 3D et d'informations sémantiques à partir de masques de carte sont utilisées. Les masques de cartes représentent des zones où les agents peuvent opérer ou non, de manière binaire. Les informations sur l'attention des piétons aux véhicules venant en sens inverse obtenues à partir des images des caméras sont également exploitées pour enrichir le système de prédiction de séquence. L'objectif est d'alimenter le modèle avec des indices contextuels et des informations sémantiques pour améliorer la prédiction des trajectoires en sachant si les agents sont conscients ou non de la présence du véhicule et en incluant des connaissances sur les zones où ils sont susceptibles naviguer. De plus, les interactions du véhicule autonome avec les agents de la circulation régissent souvent son comportement lorsque le véhicule navigue. Un mécanisme pour incorporer ces informations au modèle d'apprentissage est également développé aboutissant à un système de prédiction de trajectoire intégrant les interactions et des indices contextuels.

Les architectures d'apprentissage sont construites à partir de jeux de données acquis à partir des capteurs de perception d'un véhicule. Étant donné qu'ils jouent un rôle important dans la résolution des problèmes d'apprentissage, les jeux de données annotés disponibles pour la navigation autonome ont été examinés en fonction de la disponibilité des données des capteurs et des informations contextuelles. Sur cette base, nos expériences ont permis de valider nos modèles et de construire progressivement leur architecture. Leurs performances sont démontrées à l'aide du célèbre jeu de données NuScenes acquis en milieu urbain. Les performances de l'approche proposée comparées aux approches basées sur des modèles et des données démontrent que l'ajout de multiples informations contextuelles et des interactions d'agents permet une augmentation substantielle des performances.

– – – ● ● ● – – – ⋆ ⋆ ⋆ – – – ● ● ● – – –

To my Amma and Achai,

without whom I wouldn't be the person I am today

– – – ⋆ ⋆ ⋆ – – – ● ● ● – – – ⋆ ⋆ ⋆ – – – ● ● ● – – – ⋆ ⋆ ⋆ – – –

Acknowledgements

This thesis is easily my biggest undertaking to date and it has not been an easy trip. Completing this long and arduous journey could have never been accomplished without the presence and support of many people. First and foremost, I would like to express my gratitude to my advisors David Fillat and Javier Ibañez-guzmán without whose continuous support and push I would never have completed this journey successfully. Thank you for guiding me, correcting me when I was wrong and pushing me when I didn't believe in myself.

I am indebted to the people at ANRT, Renault and ENSTA for funding this thesis for three and a half years and making this research possible. Huge gratitude to my ENSTA colleagues and friends for the lively coffee breaks, intense babyfoot sessions, loud Teeworlds breaks and the heated Nerf battles that kept the otherwise stressful PhD life interesting and fun. Thank you Olivier, Alex, Julien, Francois, Nathalia and the others. My PhD mates Florence, Adrien, Hugo and Clement, I am grateful for your support at different points of my thesis and motivating me. Special thanks to Thibault, Tim and Gabriel for being the best and for all the workout sessions, insightful scientific discussions and fun that we had together. Thank you for making my PhD life better in general.

Colleagues at Renault - Mathieu, Edouard, Imane, You Li and others, I would like to thank you guys for the memorable times at the company and the innumerous coffee break discussions we had.

I cannot thank my friends enough for being my life support throughout my time in Paris. Thank you for the amazing memories in the city as well as for the different trips that we organised. Thank you Paola, Vishnu, James, Moad,

Kapil, Maria, Aida, Adnene, Hamzia, Raza, Caro, Mel, Eli, my flatmates Philippe and Reem and the others for each happy moment I spent with you guys. Special thanks to Laure for being the best, for putting up with all my nonsense and also proof-reading my manuscript. Without you guys, living through this pandemic and completing my thesis would have proven to be difficult. Friends who made my short period in India memorable and helped me overcome tough times in the last year - Unni, Santhu and Achu, you guys are awesome!

Last but not the least I would like to share my gratitude to my family for being there always and supporting me even from far away - Achai, Manna, Ammu, Raj Uncle, Raviappan and everyone else. Big love to both my grandmas.

Contents

Chapter 1

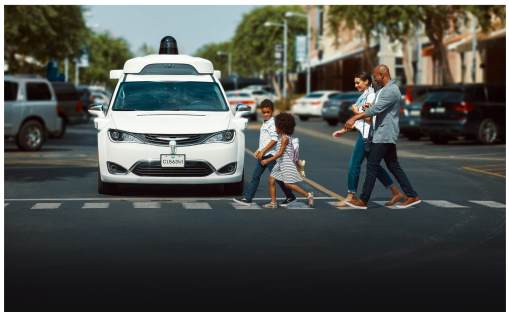
Introduction

Contents

The continuous population growth and economic improvements across the globe has meant that the transportation of people and goods is today a major daily activity. As a consequence, we have witnessed an expansion of road networks, the emergence of traffic jams, and pollution. Nevertheless, people appreciate their freedom to move despite the growing safety concerns and environmental impacts. Technological advances in sensing, computing, communications, and recent progress in machine learning are enabling the development of autonomous systems for transportation applications (Maleki et al., 2021; Sun et al., 2019). These have increasing capabilities that allow them to autonomously interact and evolve in complex environments. The deployment of autonomous vehicles evolv-

ing in existing road networks has the potential to improve the life of mankind in different ways (Maleki et al., 2021). The perceived benefits include:

- Commuters, as they drive, can improve their productivity and their quality of life because the driving task will be done by the autonomous system;
- Transportation Accessibility: Autonomous vehicles should provide means to people that have reduced access to transportation. Example, the elderly, the differently-abled, minors etc. could take this transport in an independent manner;
- Safety: Most accidents are due to human error, mainly distraction. When vehicles are under computer control, they are expected to improve safety.



(a) *Waymo*

(b) *Cruise*

Figure 1.1: *Autonomous Vehicle Companies and their vehicles*

The Urban DARPA Grand Challenge demonstrated successfully in 2006 that vehicles can navigate under full computer control (Thrun et al., 2006). From that point on, interest on the topic continued to increase and considerable progress has been achieved on the operation of Autonomous Vehicles on public road networks. Most vehicle OEMs have prototypes or road-maps for this domain. Numerous start-ups have emerged providing very competitive results. Alliances between vehicle OEMs and Start-ups running multi-million dollar projects have been announced. It appears that a new industry has emerged with major players like Waymo (way, 2020), Motional (mot, 2020), Cruise (cru, 2020) etc. Images of their vehicles are shown in Figure 1.1. Whilst the expectations in the beginning were

high, driving automation has proved to be much harder. It was predicted that some form of autonomous vehicle would already be available by 2020. However, this is not the case, much work is still required (Said et al., 2021).

The research in this thesis addresses the perception problem as applied to fully autonomous vehicles (The driver is no longer in the vehicle control loop). In order to provide a coherent concept, this is equivalent to the SAE Level 4 Automation Level. This is explained in the next section. Perception is considered one of the major mentally complex tasks for any human being and fundamental for autonomous driving. Different developments and demonstrations have shown that the perceived world needs not only to be represented digitally but also to facilitate its interpretation by a machine.

In this chapter, initially we look into the major characteristics of autonomous vehicles. This is followed by the role of context in the perception systems when applied to autonomous navigation. The problem addressed in this thesis together with the purpose and objectives are described in the next section. The contributions of the thesis are described next, and an overview of the content of this manuscript.

1.1 Autonomous Vehicles

Over the past years, the Society of Automotive Engineers (SAE) has defined 5 levels of driving automation which are used today as a standard. These are shown in Figure 1.2 (SAE, 2018).

Levels 0, 1 and 2 provide information to the driver or act on the vehicle with the driver always being part of the control loop. Level 2 autonomy with a control of the longitudinal and lateral motion but with human driver monitoring the situation is available on today's high-end vehicles (E.g. Tesla, Audi etc.). In Level 3, we have partial automation and the computer will fully control some tasks and with the driver having the ability to take over whenever necessary. Levels 4 and 5, the vehicle is under complete computer control without any driver intervention. The differences between Levels 4 and 5 is that in Level 4, the vehicle is only able to navigate autonomously in a predefined area. Whilst for Level 5, the vehicle

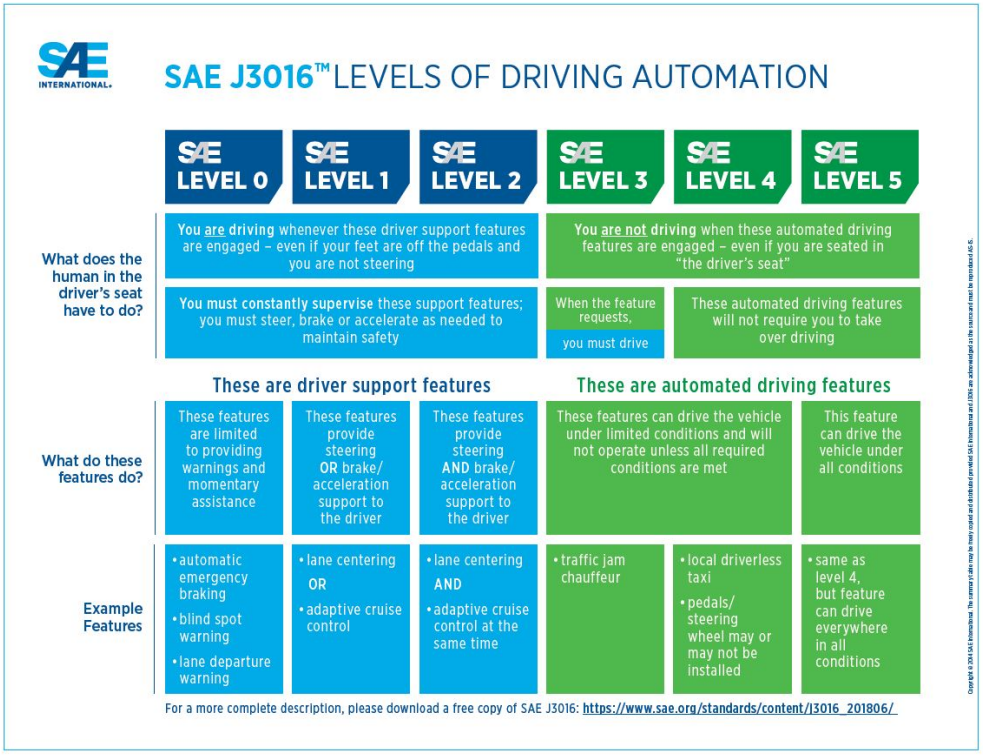


Figure 1.2: *Levels of driving automation as defined by SAE (SAE, 2018)*

can drive anywhere. At the time of writing, NAVYA, a French company, is for example building and selling Level 4 electric powered shuttles and cabs which are operating as public service vehicles in different constrained sites in Europe (nav, 2019; 2020, 2020). Waymo has driven more than 20 milions miles autonomously in public roads. They are testing robot-taxis without a safety driver in Phoenix, Arizona for the past year (way, 2020). Fully autonomous vehicles are undergoing testing in several parts of the world, but none are yet available to the general public.

1.1.1 Architecture and Sensors

To navigate autonomously, a vehicle needs to perceive its surrounding environment, build a digital representation in order to be aware of its situation, and then decide the action to take. This decision acts on the vehicle resulting in its motion

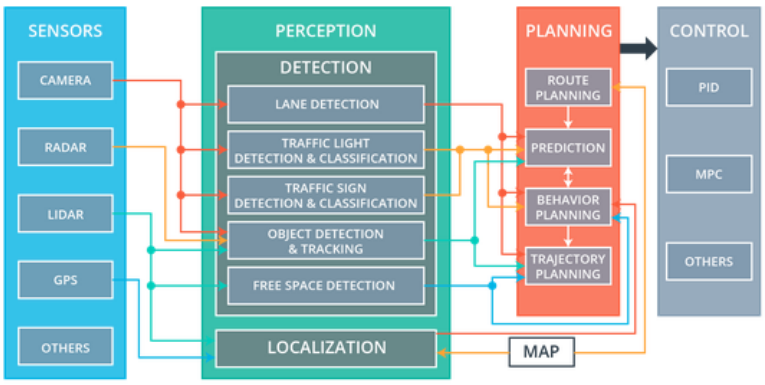


Figure 1.3: *Example of a system architecture of an Autonomous Vehicle (Raack, 2018)*

under full computer control. These typical stages involved in autonomous driving are called sensing, perception, planning and control. Figure 1.3 shows a typical functional architecture used in autonomous vehicles. A set of vehicle on-board sensors acquire data which is then processed to provide a digital representation of the environment. For example, lane detection, object detection and classification, object tracking, free space detection etc. This information is used to perform risk assessment, trajectory planning and predictions, which results on the control of the vehicle motion.

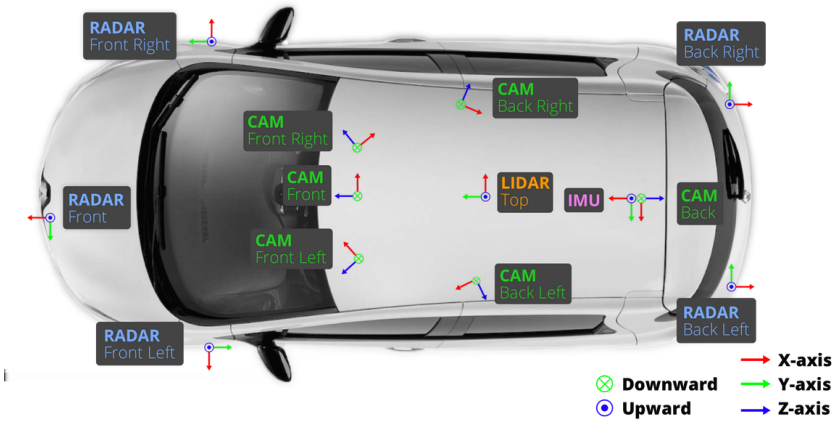


Figure 1.4: *Sensor setup on a Renault Zoe used for building the NuScenes dataset (Caesar et al., 2019)*

Perception provides the means for the vehicle to know what is going on within its operating environment. In an autonomous driving scenario, there are numerous mobile entities that appear around the ego-motion vehicle: pedestrians, other vehicles, cyclists etc. A major challenge resides in perceiving all these elements in a continuous and precise manner without false positives (i.e.,reporting non-existent obstacles) or false negatives (i.e., missing real obstacles). For this purpose, several active and passive exteroceptive sensors can be used, these include different types of video cameras, LiDARs[1], and RADARs[2]. State-of-the-art detection algorithms provide bounding boxes or segmentation for each class of objects detected. Currently, there are many algorithms that can detect these objects using vision data from cameras or point-cloud data from LiDARs (Lang et al., 2018; Chen et al., 2017a; Zhou and Tuzel, 2018).

However, there is no single sensor technology capable to provide precise and complete spatio-temporal information on all what is surrounding the vehicle, each has its own advantages and disadvantages (Kocic et al., 2018). Ongoing solutions thus combine different types of sensors through a multiple-sensor fusion process that enables the exploitation of their major features in a coherent manner, thus reducing the uncertainty associated to each one of them. As an example, Figure 1.4 shows the sensor setup on a Renault Zoe which is used for recording the NuScenes dataset (Caesar et al., 2019). It uses 6 cameras, 5 RADARs, and one LiDAR to record synchronised sensor data for research and development of perception and navigation algorithms.

1.1.2 Multi-Sensor Fusion and Trajectory Prediction

The work in this PhD will not concern directly the traffic-agent detections. It focuses on the processes beyond perception, i.e., multi-sensor fusion trajectory prediction and the need of incorporating additional information to improve the prediction.

Sensor Fusion is defined as "combining raw or processed sensor data such that the resulting information is, in some sense, better than what would be possible

[1]LIght Detection And Ranging
[2]RAdio Detection And Ranging

when these sources were used individually" (Fung et al., 2017; Kocic et al., 2018). It reduces the uncertainty associated with each of the sensors used. It balances the disadvantages of each individual sensor type and improves the robustness and the overall reliability of the system. Sensor fusion techniques can be primarily split into three categories (Fung et al., 2017):

- Low-level fusion or raw data fusion combines several sources of raw data to produce new data that is expected to be more informative than the inputs;
- Intermediate-level fusion or feature level fusion combines various features such as edges, corners, lines, textures, or positions into a feature map that may then be used for segmentation and detection;
- High-level fusion, also known as decision fusion, combines decisions from several experts. Methods of decision fusion include voting, fuzzy-logic, and statistical methods.

In this thesis, the focus is on developing high-level fusion of individual object detection.

Sensor fusion can be performed among different sensors, but also applied to the same sensor from different points of time in order to filter noise and improve precision of the information. To drive autonomously, it is necessary to anticipate the evolution of their environment based on their perceived information.

Trajectory prediction of traffic-agents in the context of autonomous vehicles account for a big part of risk-assessment and motion planning. This step beyond perception is similar to how a human would assess the situation while driving (Endsley, 2015) - Will the vehicle in front turn right? Will the pedestrian cross the road? - Questions that the driver answers passively while manoeuvring the ever-changing driving environment. By contrast, for autonomous systems, to carry out the same is a complex task. The appropriate combination of input sensors have to be chosen, complementary information must be provided and the right models implemented to achieve the accuracy needed for the vehicle to understand its situation and navigate safely (Endsley, 2015). These models have to perform tem-

poral sequence prediction. This has become an important area of research which includes multiple applications to Natural Language Processing, Weather Forecast, Economics, or Intelligent Systems.

1.1.3 Machine learning for Autonomous Vehicles

In recent years, there has been a rapid expansion in computational power and public datasets that reoriented the approaches used to solve the perception and prediction problems. It has triggered an increased use of machine learning, in particular deep learning algorithms, in perception, sensor fusion, and trajectory prediction problems that have proved to be often much more efficient than model-based methods (Lefèvre et al., 2014). Test benches and applications in the automotive domain have demonstrated the clear advantage of machine learning methods applied to perception over model/geometry-based methods (Russell and Norvig, 2009; Maleki et al., 2021; Fujiyoshi et al., 2019). The methods applied in this thesis are also centred on the use of machine learning.

To train data-driven methods, annotated datasets are needed. For the automotive domain, it is necessary to take them from moving vehicles operating in public roads to take into account all the contextual information. Several datasets have emerged applicable to the vehicle navigation domain. One of the earliest ones is the Victoria Park dataset (Guivant et al., 2000) that has been used extensively to solve the Simultaneous Localisation and Mapping (SLAM) problem. This was completed by the New College datasets (Smith et al., 2009). The publication of the KITTI dataset facilitated the application of machine learning and the benchmarking of vehicle navigation algorithms since 2013 (Geiger et al., 2013). Recent datasets, like NuScenes (Caesar et al., 2019) and Apolloscapes (Wang et al., 2019) include richer information and sensor configurations similar to those found in SAE level 4 autonomous vehicles. The performance attained were much higher in terms of accuracy and complexity of the work environments by comparison to model-based methods (Fujiyoshi et al., 2019). The success of data-based methods has attracted research on several neural networks architecture such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) or Long Short-Term Memory (LSTM) that we will seek to exploit in our research. These data driven approaches are detailed in Chapter 3.

1.2 Role of Context in Perception for Autonomous Navigation

As introduced before, perception systems for autonomous vehicles address different functions - environmental image acquisition, object detection and classification, tracking and trajectory prediction, to name the most important. The accuracy with which the perception system solves each of these tasks depends not only on the data acquired by the sensors installed but also on additional information given to or acquired by the system. The latter helps to provide a "context" to the perception algorithms so as to identify different scenarios.

Context is part of a process of interacting with an ever-changing environment composed of "re-configurable, migratory, distributed, and multi scale resources" (Coutaz et al., 2005). It provides additional information that facilitates the interpretation of data. As an example, for vehicle navigation, the perception problem does not stop at detecting the vehicles in front of the subject vehicle. We should also determine whether or not it is relevant for the driving decision. For this purpose, we can take advantage additional information. In the case of autonomous vehicles we use maps: we can for example project the perceived vehicle onto an HD-map (high definition map), which makes it possible to infer whether the vehicle is within our lane or whether its path will cross our planned path.

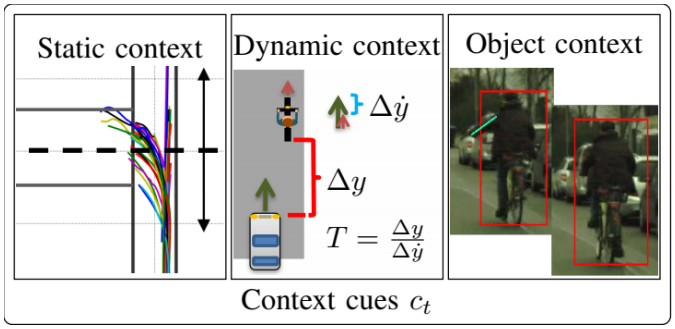


Figure 1.5: *A schematic overview of the position and context features used in Pool et al. (2019). The static context is the distance to the intersection where the cyclist might turn left. The dynamic context is the time for the vehicle to overtake the cyclist assuming they maintain their pace. The object context indicates the confidence, of a trained detector, whether the cyclist is lifting his/her arm.*

Context could be obtained from different sources such as HD-maps of the surrounding environment or information extracted from the sensory data like pedestrian head-orientation or gait (see Figure 1.5 for example taken from (Pool et al., 2019)). One of the goals of this thesis is to explore how this information can help in trajectory prediction in an autonomous navigation scenario. For example, the trajectory of a pedestrian could be influenced by whether he/she has spotted the vehicle or not, and whether there is a pedestrian crossing or not. HD-maps are an important tool for providing data on whether road markings or traffic signals are present in the environment. This, in addition to perception algorithms that detect the state of these signs, becomes a strong tool in solving these use-cases (Ma et al., 2018; William et al., 2019).

Wirthmüller et al. (2020) also explores the possibility of using daytime, day of the week, weather, location and traffic density as contextual cues. They classify such information as *external conditions* and the more studied cues such as lane markings (Bartoli et al., 2018), map information, intersection distances or topologies (Klingelschmitt and Eggert, 2015), traffic rules (Gindele et al., 2013), and intentions (Schneemann and Heinemann, 2016) as *situation context*. Figure 1.6 shows the utilisation of distance of pedestrians to curbside and traffic light status as situation context for predicting their movement. In this thesis, the focus is on using situation context.

1.3 Problem Formulation

To summarise this introduction, machine learning has contributed very much to perception applied to autonomous vehicle navigation by enabling large progress on object detection. However, models rely often on a single sensor. As a result, there are shortcomings in the form of false positives and false negatives and other detection problems, unacceptable for safety critical navigation tasks that can be addressed by sensor fusion algorithms. They play an important role by merging the information gathered from different sensors and producing a detection or a track which is robust and more efficient than using a single sensor. Even then, conventional fusion algorithms still have several limitations due to limited input information and can be improved by taking advantage of various form of context

Figure 1.6: *Example intersection scenario. Dashed green line denotes a rectangular approximation to the curbside in view. Orange arrows denote relative distance of a pedestrian from the two curbsides, which can indicate pedestrian intention. Pedestrian traffic light status is highlighted in orange, which also influences pedestrian movement (Habibi et al., 2018)*

information.

In this thesis, we explore the application of deep learning methods to a multiple sensor fusion process and the inclusion of context to achieve Multi-Object Trajectory Prediction for autonomous navigation. The purpose is therefore to enhance the performance of perception systems applied to highly automated driving via multi-sensor fusion techniques and the exploitation of contextual information. We focus on the tracking of perceived entities such as pedestrians and vehicles, in order to predict their motion, within the context of urban and peri-urban environments. For this purpose, two typical sensors are used - video cameras and LiDARs - and contextual data coming form maps of the environment or the sensor themselves.

The main objectives are therefore:

- To understand the use of machine learning algorithms for multi-sensor data fusion through a critical state-of-the-art review and identify the most recurrent problem in perception, and trajectory prediction of several traffic-agents;

- To propose a framework with multi-modal input for the trajectory prediction problem, to identify and implement the architecture and systems to incorporate context into the prediction model;
- To validate and analyse the performance of the proposed framework using data acquired from a vehicle navigating on public roads.

1.4 Contributions

Our work has resulted in the following publications:

- Palli Thazha, V., Filliat, D., and Ibañez-Guzmán, J. (2019). Applying map-masks to Trajectory Prediction for Interacting Traffic-Agents. 3rd Edition Deep Learning for Automated Driving (DLAD) workshop, IEEE International Conference on Intelligent Transportation Systems (ITSC'19)
- Palli-Thazha, V., Filliat, D., and Ibañez-Guzmán, J. (2020). Trajectory prediction of traffic agents: Incorporating context into machine learning approaches. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–6

1.5 Thesis overview

In the remainder of this manuscript, chapters 2 and 3 introduce the technology, the sensors, the software architectures and methods that form the perception systems for autonomous vehicles. Chapters 4, 5 and 6 demonstrate the methods developed during this thesis and illustrates the related experiments and results. More precisely:

- Chapter 2 presents the state-of-the-art research on trajectory prediction methods, deep learning based prediction and works that have inspired the ideas behind this thesis. It summarises the different prediction methods, the data and sensors that are generally exploited for trajectory prediction and the evaluation metrics adopted to compare their performances.

- Chapter 3 introduces the different neural networks and deep learning techniques used in perception systems for self-driving cars. It also presents the important datasets that help push research on perception, navigation and decision making in intelligent vehicles.
- Chapter 4 presents an experiment conducted to study the classification of traffic-agents based on their trajectory. Different classes of traffic-agents have distinct dynamic properties and their motion pattern can be used as the feature of each class.
- Chapter 5 discusses the methods developed for trajectory prediction of multiple classes of traffic-agents using map as a context cue.
- Chapter 6 describes the proposal of exploiting pedestrian head orientation (pedestrian attention) along with map information and historical trajectory information for predicting the interactions at pedestrian crossings.
- Finally, Chapter 7 summarises the proposed methods and elaborates the possible future directions in which this work could be extended.

Chapter 2

Trajectory Prediction: A Literature Review

Contents

Trajectory prediction in the context of autonomous vehicles is an important task as it facilitates situation understanding for vehicle navigation purposes. To study the prediction of a traffic-agent trajectory, several factors must be considered, such as the observation of past trajectories of each traffic-agent, observing the surrounding environment to identify risk, identifying cues that will affect the agent

behaviour, or the effects of the likely motion of the ego vehicle. A reliable prediction system contributes to safe decision-making. This reduces the risk associated with vehicle navigation under full computer control. Autonomous navigation can be regarded as inherently risky. This is owing to the fact that the traffic-agents encountered by the vehicle in public road networks are highly interactive and dynamic in nature. In this chapter, we look into trajectory prediction of different traffic-agents by exploring the methods that have been tested and proven, the data, and different context cues that help achieve this purpose. For this, the work found in the literature is categorised and compared according to the prediction methods, data involved and datasets exploited. Data from vehicle proprioceptive sensors, exteroceptive sensors (e.g. cameras, LiDAR and RADAR) as well as cartographic information are studied.

Prediction methods are classified into model-based and data-driven. The former follows classical methods represented by physics models, whilst the latter addresses those dependent on machine learning, in particular deep learning. Moreover, works are classified based on whether the observer (the ego-motion vehicle) is at rest or in motion.

2.1 Prediction Methods

Prediction methods can be split into two broad categories - Physics / Model-based and Data-based. Physics/Model-based prediction methods deal with the kinematics or dynamics of the moving agent be it a car, a pedestrian or a bike. These movements can be approximated by different models and the trajectory can be predicted. In contrast, prediction using data-based approaches learn from examples. They use multiple datasets to understand how different traffic-agents behave in different scenarios. In this section, we compare and analyse the different methods available in the literature to determine their strengths and weaknesses as well as the assumptions under which solutions are proposed.

2.1.1 Model-based Prediction Methods

Trajectory prediction for pedestrians, vehicles and other traffic-agents has been studied using model-based approaches: Constant Velocity (Schöller et al., 2019), Constant Curvature (Horst and Barbera, 2006), Linear Regressions, Kalman Filters (Kalman, 1960), Monte Carlo Simulation (Danielsson et al., 2007), Time-series methods and Hidden Markov Models (Firl et al., 2012) are some examples. Most of these methods are limited to short-term predictions. Some offer long-term prediction but their accuracy is affected by the lack of context information and ignoring the interaction between different classes of traffic-agents. A survey on motion prediction can be found in Lefèvre et al. (2014) where the authors analyse different trajectory prediction methods based on model completeness and vehicle collision risk assessment.

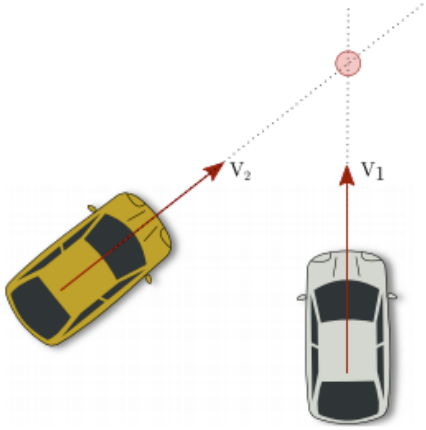


Figure 2.1: *Trajectory prediction with a constant velocity motion model (Miller and Huang, 2002). Illustrations from Lefèvre et al. (2014)*

Several methods rely on the use of dynamic models for predicting the behaviour of vehicles. Some models include representation of the longitudinal and lateral forces, or the road banking angle (Brännström et al., 2010). However, these methods are better suited for the design of vehicle control systems, as hese dynamic models require more observations and model parameters, whilst for applications like trajectory prediction, simpler bicycle models are preferred (Chiu-Feng Lin et al., 2000; Jihua Huang and Han-Shue Tan, 2006).

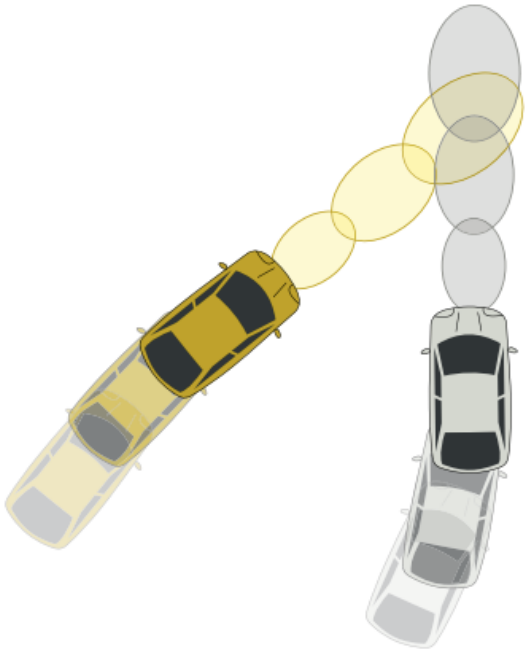


Figure 2.2: *Trajectory prediction with a constant velocity motion model and Gaussian noise simulation. Ellipses represent the uncertainty on the predicted positions (Ammoun and Nashashibi, 2009). Illustrations from Lefèvre et al. (2014)*

Kinematic models describe a vehicle's motion based on the mathematical relationship between the parameters of the movement (e.g. position, heading, velocity), without considering the forces that affect the motion (Lefèvre et al., 2014). The simplest models are Constant Velocity (CV) and Constant Acceleration (CA), applied in the case of straight motion. Miller and Huang (2002) proposes a cooperative vehicle collision warning system and implements the CV model to determine a collision between interacting vehicles. This is illustrated in Figure 2.1. Ammoun and Nashashibi (2009) exploits knowledge acquired through shared information via wireless communication links to predict the trajectories of the surrounding vehicles. These are utilised to identify the configurations of possible collisions between vehicles. Figure 2.2 shows a CV model with Gaussian noise to predict collisions. In the same work, the Constant Turn Rate and Velocity (CTRV) and Constant Turn Rate and Acceleration (CTRA) models allow to represent turns (Ammoun and Nashashibi, 2009; Hillenbrand et al., 2006).

Several works on human motion prediction utilise physics-based models like in Pellegrini et al. (2009); Althoff et al. (2013); Elnagar (2001). Elnagar (2001), for example, propose a framework for predicting future positions and orientation of moving obstacles in a time-varying environment by implementing Kalman filtering techniques. Pellegrini et al. (2009) introduces a dynamic social behaviour model, inspired by models developed for crowd simulation and Althoff et al. (2013) predicts trajectories through a dynamic model by computing the possible occupancy areas for consecutive time intervals applicable to all the agents participating in the situation of interest. Tonoki et al. (2017) uses a model-based method for pedestrian prediction. They record pedestrian trajectory data using an environmental Laser Range Finder (LRF) with an Extended Kalman filter (EKF) and construct pedestrian movement models using a Vector Auto Regressive (VAR) model. The output is the pedestrian state represented by the position, speed and direction.

This section provides relevant work done with the use of model-based techniques. We focus our attention to obtain better performance by using data.

2.1.2 Data-based Prediction Methods

In our work, learning from data refers to developing machine learning algorithms (deep learning) to infer motion predictions. The approaches introduced in this section are data-driven and thus involve machine learning, particularly deep learning networks like Recurrent Neural Networks (RNNs), Long-Short Term Memories (LSTMs) and Convolutional Neural Networks (CNNs). RNNs and LSTMs are employed in sequence prediction problems whilst CNNs are mainly implemented to interpret images. For completeness, these networks are explained in detail in Chapter 3. Such trajectory prediction methods have been applied to data acquired by different sensors. Some solely with vision-based detection outputs, others with 3D bounding-boxes obtained from LiDAR 3D point-clouds. Much research has gone into utilising data to learn the motion patterns of different classes of traffic-agents.

Altché and de La Fortelle (2018) utilises LSTMs to predict the trajectory of a single target vehicle on the NGSIM video dataset (Li et al., 2019). It predicts the longitudinal and lateral trajectories of vehicles on a highway. Milan et al. (2016) propose an end-to-end learning approach for online multi-target tracking

using RNNs to predict the state of each target step by step. This is shown in Figure 2.3. They use LSTMs to achieve data association by working on vision data obtained from the 3D MOTChallenge dataset. Nikhil and Morris (2019) implement a CNN-based approach to predict trajectories of surrounding vehicles. This model applies highly parallelisable convolutional layers to handle temporal dependencies instead of using recurrent networks. The trajectory histories are embedded to a fixed size tensor and stacked convolutional layers used to ensure temporal consistency. Jawed et al. (2019) also utilise a similar architecture based on convolutional layers.

These works do not take into account the surrounding environment and context which could improve the prediction in a real driving scenario. But it is a good way to start building a base for prediction architectures.

2.1.2.1 Interaction and Context Aware Models

In this section, we group data-based approaches based on whether the interaction with other traffic-agents is taken into consideration and whether they utilise context cues. Interaction could be between same or different classes of traffic-agents or with the environment. Interactions with the environment are generally modelled through context cues. When the future trajectory of a traffic-agent is predicted over a long period, the interaction between several agents and with the environment needs to be taken into account. These approaches are mainly data driven. There are several examples in the literature that try to solve the prediction problem involving interactions between traffic-agents. Some focus either on pedestrians or vehicles while others try to solve cases with interactions between multiple classes of agents.

Several works take into account the interaction between vehicles for predicting their trajectories. Hu et al. (2019) employ a generative model to jointly predict the sequential motion of each pair of interacting vehicles. The inputs are trajectories and environment information in the form of location and velocity of surrounding vehicles. Kim et al. (2017) generate occupancy grids and utilise RNNs to predict the trajectory of vehicles on highways. They focus on solving vehicle trajectories on highways and do not utilise context cues in their research, but

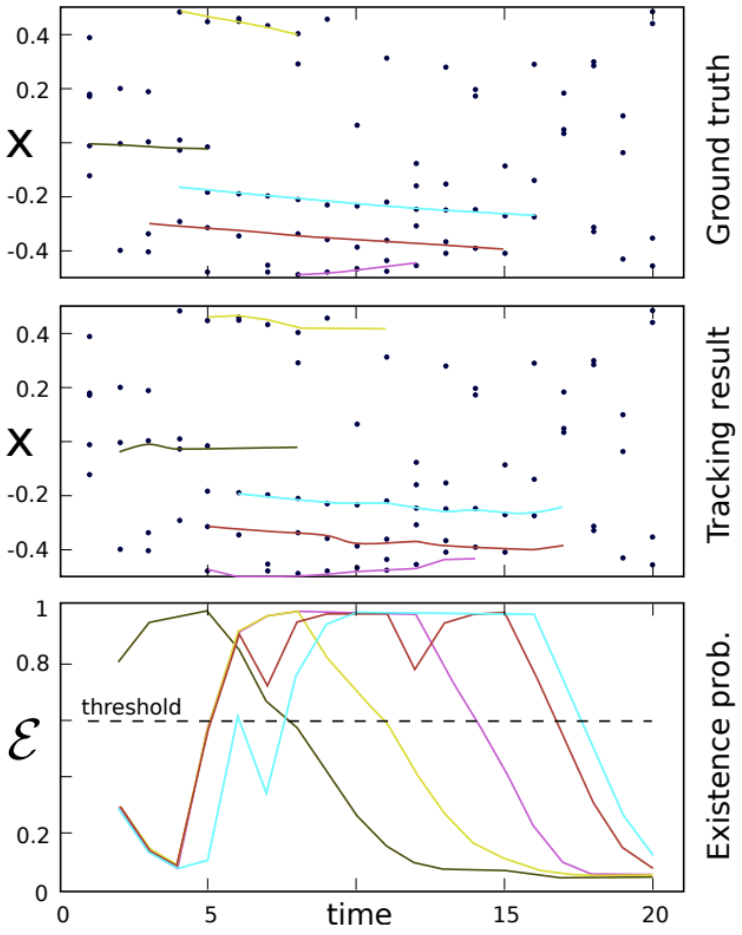


Figure 2.3: *Results of A. Milan's tracking method on a 20-frame long synthetic sequence with clutter. Top: Ground truth (x-coordinate vs. time). Middle: Reconstructed trajectories. Bottom: The existence probability ϵ for each target. Note the delayed initiation and termination, e.g. for the top-most track (yellow) in the middle. (Milan et al., 2016)*

their limited application scenario implicitly put constraints on the motion of vehicle in this particular context. Deo and Trivedi (2018) implement an LSTM encoder-decoder architecture that utilises convolutional social pooling to learn inter-dependencies between vehicles. Here the spatial configuration of the agents in a scene is embedded into a spatial grid around the vehicle ego-motion. This grid is passed into convolutional and pooling layers to obtain the social context encoding. This, along with an LSTM encoding of the agents' trajectory, is con-

catenated and passed through a decoder to obtain the final predicted trajectory, as illustrated in Figure 2.4. An extension of this work is done by Messaoud et al. (2019) where they implement a non-local social pooling module. The non-local multi-head attention mechanism captures the relative importance of each vehicle despite the inter-vehicle distances to the target vehicle, while the local blocks represent nearby interactions between vehicles. A similar architecture is implemented by Park et al. (2018) with the decoder producing the most likely trajectories over an occupancy grid using the beam search technique.

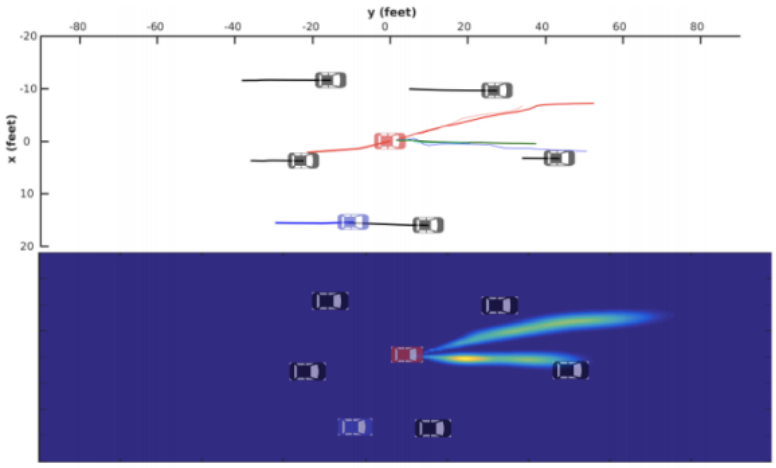


Figure 2.4: *Illustration of Multi-modal predictions of future motion of the surrounding vehicles, along with prediction uncertainty shown here for the red vehicle. Blue is the ego-motion vehicle (Deo and Trivedi, 2018)*

Kotseruba et al. (2016) present a novel dataset for a critical aspect of autonomous driving, the joint attention that must occur between drivers and of pedestrians, cyclists or other drivers. They also show how visual complexity of the behaviours and scene understanding is affected by various factors such as different weather conditions, geographical locations, traffic and demographics of the people involved. Malla et al. (2020) introduce TITAN (Trajectory Inference using Targeted Action priors Network), a new model that incorporates prior positions, actions, and context to forecast future trajectory of agents and future ego-motion. Alahi et al. (2016) tries to learn general human movement and then predict their future trajectories. They solve the pedestrian motion prediction problem by im-

plementing a pooling-based LSTM architecture and jointly predict the trajectories of all the people within the sensor's field of view. LSTMs have proven to be effective when sequences have to be predicted. This is the reason why researchers adopt LSTMs for trajectory prediction problems. But sequences alone cannot completely solve the problem of human trajectory prediction. Context cues play an important role also. These works show the importance of interaction awareness in solving trajectory prediction. We will take inspiration from these works in the later sections to solve pedestrian crossing scenarios.

Context cues like head-orientation and distance from curbside, pedestrian crossings, static environment information and maps can improve trajectory prediction.The following works are examples for such cases:

Kooij et al. (2014) propose a Dynamic Bayesian Network throughout a situational awareness and spatial layout perspective to predict pedestrian paths. For this purpose, they employ the following inputs and cues: pedestrian head orientation, distance between vehicles and the pedestrian at expected point of closest approach, and distance of the pedestrian to curbside. They are able to predict changes in the observed pedestrian dynamics - mainly the stopping probability. Lee et al. (2017) apply a Conditional Variational AutoEncoder (CVAE) based RNN encoder-decoder to make prediction for interacting agents. They take past trajectories as input along with a feature map generated from scene elements like roads and sidewalks. These feature maps provide semantic scene information to the prediction model. In an urban scenario, such context information is important to account for the behaviour of various traffic-agents. Habibi et al. (2018) input relative distance to curbside and state of pedestrian traffic lights as additional information. This provides context to predict the pedestrian path. Ridel et al. (2019) implicitly model pedestrian interactions with vehicles to predict pedestrian behaviour. They exploit pedestrian head orientation as an input along with pedestrian location and past trajectories to an LSTM encoder. The final trajectory is predicted using an LSTM decoder. Another study by Ma et al. (2018) generates 4D graphs to model interactions and classes of the interacting agents. Two dimensions for instances and their interactions, one for time series and another for high-level categorisation. In the graph, all valid instances and categories of traffic-

agents are denoted as nodes. All relationships in spatial and temporal space are represented as edges. An LSTM architecture (with two main layers - one for instances and one for categories) is designed to generate trajectories from these 4D graphs. The idea behind using a category layer is that traffic-agents in the same class will have similar dynamic properties and reactions to other agents and the environment. The results are shown in Figure 2.5. It can be observed that they predict trajectories for different classes of traffic-agents in highly interactive urban environments. However, they do not take into account any context information.

Figure 2.5: *Illustration of TrafficPredict (TP) method on camera-based images (Ma et al., 2018). Green lines are the ground truth and the proposed method is shown in pink. We observe that the environment is highly diverse and dense.*

2.2 Data used in Perception Research

Predicting the stochastic behaviour of pedestrians, or the more organised behaviour of cars or bicycles depends a lot on the available type of data. Studies have been done on passive sensors like monocular cameras and infrared cameras, and active sensors like LiDAR and RADAR.

The methods also differ based on whether the observer is static or dynamic. Prediction of traffic-agents from cameras on highways is an example of a static observer, whereas sensors on an autonomous vehicle account for a dynamic observer.

Having multiple types of sensors helps in accounting for the shortcomings of a single sensor. Hence it is important to study sensor fusion algorithms to improve trajectory prediction of traffic-agents. In this section, we categorise the methods based on the sensor input - whether they utilise Active Sensors or passive ones. Works are also differentiated based on the state of the observer - Static or Dynamic. Finally, research that involves sensor fusion is discussed.

2.2.1 Perception Techniques based on Active Sensors

Active sensors utilised in autonomous vehicle research mainly include LiDARs, RADARs and Ultrasound sensors. They have lots of benefits like high-precision distance measurement and usability during day and night and are, in most cases, not affected by extreme sunlight. Some disadvantages include higher costs for the hardware, high power consumption, increase/decrease in precision of measurement based on the number of layers in the case of a LiDAR, and difficulty to interpret data when compared to cameras.

Both LiDAR and RADAR use light-waves. They offer similar information to an autonomous vehicle, with a few notable differences. LiDAR is more accurate due to its higher frequency waves and hence utilised to detect nearby objects. It is able to perceive smaller objects and provides a more accurate measurement of its shape. This quality becomes a challenge during extreme weather conditions. LiDAR is inaccurate in snow, rain, and fog, as it detects those small particles in the air. RADAR has the ability to see through these by using low frequency radio waves. However, they are less precise in direction and return a lot of noise. It has a lower cost when compared to LiDARs. Both sensors complement each other in most autonomous vehicles.

Among the active sensors, RADARs are implemented mainly for car and bike detection. For example, Park et al. (2018) work with Delphi long range front radars to record vehicle trajectories and implements an LSTM encoder-decoder architecture to predict future vehicle trajectories. Laser sensors (Figure 2.6) have been more widely used than RADAR for object detection, segmentation and trajectory prediction.

Figure 2.6: *Velodyne LiDARs - A Velodyne HDL-64E, an HDL-32E, a Puck, and an Ultra Puck*

Earlier works involved single-layer LiDARs. Oñoro et al. (2013) work on a single-layer LiDAR for vehicle and pedestrian segmentation. Guerrero-Higueras et al. (2019) utilise a 2D LiDAR and Convolutional Neural Networks (CNNs) for tracking people using a mobile robot. With improvement in LiDAR hardware, multi-layer LiDARs were introduced and most autonomous vehicles now have 32- or 64-layer LiDARs. This has paved the way for better feature detection using these sensors. Premebida et al. (2009) tested a four-layer LiDAR in combination with an RGB camera for detecting pedestrians in an urban scenario. Works described in Dewan et al. (2016); Engelcke et al. (2017); Li (2017) and Levinson and Thrun (2014) also utilise LiDAR point-cloud data for object detection and subsequent tracking of the detected objects.

Among the existing datasets, that will be reviewed more thoroughly in the next chapter, KITTI (Geiger et al., 2013), NuScenes (Caesar et al., 2019), Appoloscapes (Wang et al., 2019), FORD campus (Pandey et al., 2011), Oxford Robotcar (Maddern et al., 2017a), Waymo (Sun et al., 2019) and Standford Track collection (Teichman et al., 2011), offer a wide range of point-cloud data collected using Velodyne HDL-64/32 and SICK LiDAR sensors.

There exist studies which use LiDARs for detection and tracking. Most

of these only use data from a single sensor and do not take cues from other sources. This limits the possibility of achieving a complete observation of the environment. Context cues and other sensors have to be used to complement the information obtained from this data. In the next section, we summarise the works that use passive sensors and how these can be used to augment the information that we obtain from active sensors.

2.2.2 Perception Techniques based on Passive Sensors

Passive sensors, typically different types of cameras, are employed in most autonomous vehicles. These provide a semantic understanding of the environment which the vehicle is traversing. Research has provided a multitude of vision-based solutions through deep learning for semantic segmentation, object detection, depth estimation and trajectory prediction.

Cameras have the upper hand when it comes to scene understanding as they can "see" the environment. They are feature rich. The hardware cost is also relatively very low when compared to active sensors. But adding software licensing fees and intense development and training costs to this has made camera systems for autonomous vehicles more expensive in recent years. When it comes to depth sensing, cameras do not perform well as the depth has to be inferred from relative position and size of objects from the camera. Only a depth estimate can be obtained and not accurate distances as compared to active sensors. Other factors that trouble a camera-based system are extreme lighting conditions, time of day and rain. Hence, it can be deducted that for a robust autonomous system, a combination of active and passive sensors will be required to overcome the shortcomings of each other.

Methods proposed in Nikhil and Morris (2019), Alahi et al. (2016), Gupta et al. (2018), Milan et al. (2016), Fernando et al. (2018b) focus on data obtained from vision sensors. They implement techniques like Convolutions, Generative Adversarial Networks (GANs), RNNs or LSTMs on various information obtained from these images. Altché and de La Fortelle (2018), Nikhil and Morris (2019), Jawed et al. (2019), Milan et al. (2016), Messaoud et al. (2019), Deo and Trivedi (2018) work on the sequential position information obtained from image or video

datasets, while Fernando et al. (2018b); Kooij et al. (2014) work on raw images for pedestrian detection first and then predicts their future movement. Chandra et al. (2019) developed graph LSTMs to process the spatial coordinates of the road-agent defined by the set of vertices of the detected vehicle on an image. Nikhil and Morris (2019) use sequential pedestrian position and do not use any context cues. Alahi et al. (2016), Gupta et al. (2018) and Fernando et al. (2018b) also focus on pedestrian trajectory prediction with social interaction. They do not take into account other context cue besides the position of the pedestrians. Chandra et al. (2019) focus on predicting the behaviour of vehicles. They predict both spatial coordinates as well as whether a road-agent is going to exhibit overspeeding, underspeeding, or neutral behavior by modeling spatial interactions between road-agents.

Here again, many datasets like NGSIM US101 (Colyar and Halkias, 2007), UCY (Lerner et al., 2007), ETH pedestrian dataset (Pellegrini et al., 2009), ImageNET (Deng et al., 2009) or Microsoft COCO (Lin et al., 2014), NuScenes (Caesar et al., 2019) and Apolloscape (Wang et al., 2019) provide images or videos for learning such applications.

2.2.3 Perception Techniques based on Sensor Fusion

Sensor Fusion is defined as the combining of sensory data or data derived from sensory data such that the resulting information is, in some sense, better than what would be possible when these sources are used individually (Fung et al., 2017; Kocic et al., 2018). This helps to overcome physical limitations of sensing systems, balances the disadvantages of each individual sensor type and improves robustness and the overall reliability of the system.

Zhao et al. (2019) propose to fuse multiple image information using a Multi-agent tensor fusion method which exploits the past trajectories of multiple dynamic interacting agents, and a scene/image containing a static context by using data from the ETC-UCY and Stanford drone dataset. The model decodes recurrently to multiple agents' future trajectories, using adversarial loss to learn stochastic predictions. MV3D (Chen et al., 2017b) combines information from images and LiDAR data for object detection, classification and 3D bounding box

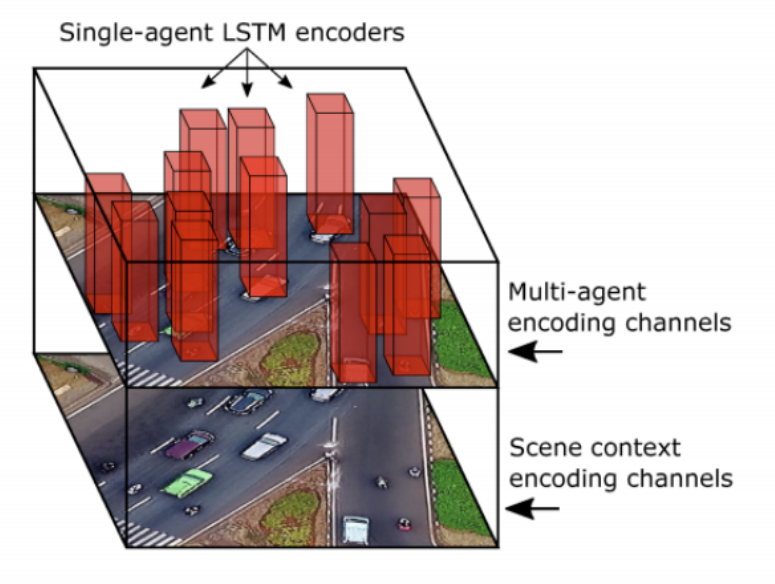


Figure 2.7: *Multi Agent tensor encoding in Zhao et al. (2019) encodes agent behaviour and context through feature maps*

regression. The network takes the bird's eye view and front view of LIDAR point-cloud as well as an image as input. It first generates 3D object proposals from bird's eye view map and project them to three views. A deep fusion network is introduced to combine region-wise features obtained via ROI pooling for each view. The fused features are exploited to jointly predict object class and do oriented 3D box regression. Casas et al. (2018) introduce a one-stage detector and forecaster that exploits both 3D point-clouds produced by a LiDAR sensor as well as dynamic maps of the environment. They claim better accuracy than the respective separate modules while saving computation time, which is critical to reduce reaction time in self-driving applications. Dimitrievski et al. (2019) employ camera and LiDAR data fusion to solve the association problem, for pedestrian tracking, where the optimal solution is found by matching 2D and 3D detections to tracks using a joint log-likelihood observation model. They utilise a behavioural motion model and a non-parametric distribution as state model, to accurately track unpredictable pedestrian motion in the presence of heavy occlusion.

2.2.4 Observer position

The prediction problem changes in complexity based on whether the observer is static or dynamic. A static observer problem is inherently much easier to solve as it does not have to compensate for the ego-motion of the sensor. Literature has a lot of research which tries to solve static observer trajectory prediction problems. For example, the work done by Milan et al. (2016) mainly focus on a static observer, specifically a static camera observing the motion of people in a crowded environment. Altché and de La Fortelle (2018) also work on a static observer setup by working on the NGSIM US101 dataset while Deo and Trivedi (2018) utilise the I-80 dataset in addition to it. Example data from NGSIM dataset is shown in Figure 2.8. They study the trajectory of vehicles on a highway observed through aerial ortho-rectified photos and videos. Hu et al. (2019) collected a dataset at a single-lane roundabout in Berkeley, California. The data was recorded by a drone from bird's-eye view. This allowed them to study the behaviour of vehicles at roundabouts.

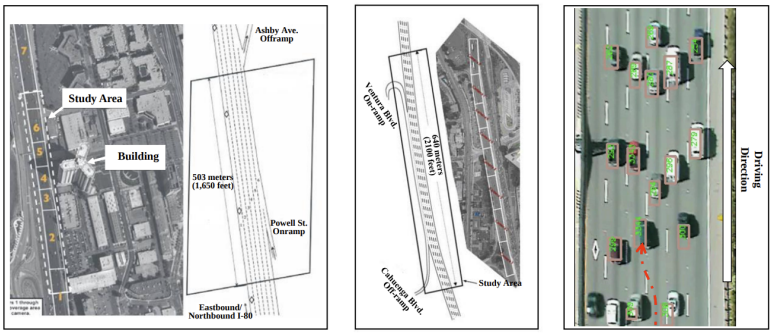


Figure 2.8: *Overview of traffic environments on NGSIM datasets (Li et al., 2019)*

In most autonomous driving scenarios, the ego-motion vehicle will be moving and the study of trajectory prediction of traffic in such a setting becomes extremely important. Such a study requires data recorded from moving sensors. Nikhil and Morris (2019); Alahi et al. (2016); Gupta et al. (2018) work on publicly available datasets - ETH and UCY which provide over 1500 pedestrian trajectories in varied crowd settings recorded from moving cameras. Jawed et al. (2019) manipulate the Udacity dataset which consists of 30,000 image frames recorded at 20 frames per second. The data is captured with a camera mounted on the

windshield of a car while driving in Mountain View, California. Kim et al. (2017) predict future trajectory of the surrounding vehicles over the occupancy grid map. The experiments are conducted on the data collected from a highway driving scenario and they show that the proposed method offers better prediction accuracy over the existing Kalman filter-based methods. Park et al. (2018) collected a large set of vehicle trajectory data from several hours of highway driving around Seoul, South Korea. The test vehicle was equipped with Delphi long range front radars. Kooij et al. (2014) created a dataset consisting of 58 sequences recorded using a stereo camera (baseline 22 cm, 16 fps) mounted behind the windshield of a vehicle.

2.3 Evaluation Metrics

Two classifier evaluation indexes applied in this thesis - Sensitivity and Precision. These are defined as:

- Sensitivity is an evaluation of a classifier's positive finding capability.

$$Sensitivity = \frac{TruePositives}{TotalPositives} \tag{2.1}$$

- Precision is introduced to find the ratio of true positives in all positive labelled samples.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.2}$$

To evaluate the prediction accuracy of trajectory prediction methods, the most commonly used metrics are described in Gupta et al. (2018); Zhao et al. (2019): Average Displacement Error (ADE) and Final Displacement Error (FDE) in pixels with respect to each time-step t within the prediction horizon:

1. Average Displacement Error (ADE): Average distance between ground truth and our prediction over all predicted time-steps for the traffic-agent.

$$ADE(i) = \frac{1}{T} \sum_{j=1,2...T} \sqrt{(\hat{x}_i^j - x_i^j)^2 - (\hat{y}_i^j - y_i^j)^2} \tag{2.3}$$

$$ADE = \frac{1}{n} \sum_{i=1,2,...n} ADE(i) \tag{2.4}$$

2. Final Displacement Error (FDE): The distance between the predicted final destination and the true final position of the agent.

$$FDE(i) = \frac{1}{n} \sqrt{(\hat{x}_i^T - x_i^T)^2 - (\hat{y}_i^T - y_i^T)^2} \tag{2.5}$$

$$FDE = \frac{1}{n} \sum_{i=1,2,...n} FDE(i) \tag{2.6}$$

where n is the total number of interacting agents in the test set, x_i^j and y_i^j denote the coordinates of the i^{th} agent in the predicted time-step j and T denotes the final predicted time-step.

Using these metrics, an important factor to be chosen is the time horizon in which the prediction will be performed. This depends on the class of the object under consideration because different traffic-agents move at a different pace. Pedestrians are typically predicted for a short time-period of 3s or for a long time-period of 10s (Fernando et al., 2018b). Vehicles and bikes, on the other hand, are often predicted for 1, 3 or 5s (Deo and Trivedi, 2018; Park et al., 2018).

2.4 Conclusion

This chapter introduces the related works that have inspired the research presented in this manuscript. Studies involving trajectory prediction, intention, tracking and sensor fusion for interacting traffic-agents are classified based on the prediction methods - whether they are model-based, data-based and/or interaction/context-aware models. Classification based on the input sensors that are utilised is also summarised. They are differentiated as active sensors, passive sensors, sensor fusion and position of the observing sensor. Table 2.1 summarises the most relevant

works based on the methods, sensors and context cues in predicting the trajectories of traffic-agents.

Paper	Model-based	Data-based	Interaction	Context	LiDAR	Image	Fusion
Tonoki 2017 (Tonoki et al., 2017)	✓	-	✓	-	✓	-	-
Ammoun 2009 (Ammoun and Nashashibi, 2009)	✓	-	✓	-	-	-	-
Altché 2018 (Altché and de La Fortelle, 2018)	-	✓	✓	-	-	✓	-
Kooij 2014 (Kooij et al., 2014)	-	✓	✓	✓	-	✓	-
Milan 2016 (Milan et al., 2016)	-	✓	✓	-	-	✓	-
Alahi 2016 (Alahi et al., 2016)	-	✓	✓	-	-	✓	-
Kim 2017 (Kim et al., 2017)	-	✓	✓	✓	✓	✓	-
Lee 2017 (Lee et al., 2017)	-	✓	✓	✓	✓	✓	✓
Deo 2018 (Deo and Trivedi, 2018)	-	✓	✓	-	-	✓	✓
Park 2018 (Park et al., 2018)	-	✓	✓	-	-	-	-
Nikhil 2019 (Nikhil and Morris, 2019)	-	✓	✓	-	-	✓	-
Jawed 2019 (Jawed et al., 2019)	-	✓	✓	-	-	✓	-
Hu 2019 (Hu et al., 2019)	-	✓	✓	-	-	✓	-
Chandra 2019 (Chandra et al., 2019)	-	✓	✓	-	-	✓	-

Table 2.1: *Comparison of Trajectory Prediction literature.*

Results show that a fusion of data and information from different sensors and the context cues will provide better trajectory prediction. This will aid in solving problems like occlusion, missing detections, crowded scenes and other problems that will arise in a difficult urban scenario. We take pointers from these works and propose new techniques to improve such a prediction of trajectories.

The next chapter, will introduce basic deep learning concepts and networks that we utilise in our research. Relevant datasets with regards to autonomous vehicle navigation are also presented and critiqued.

Chapter 3

Deep Learning for Intelligent Vehicles

Contents

Machine learning is a branch of artificial intelligence (AI) focused on building applications that learn from data and improve their accuracy over time without being programmed to do it (IBM, 2020). The behaviour of the system is not explicitly programmed but is learned from going through examples of similar scenarios. By gathering knowledge from experience, this approach avoids the need for human operators to formally specify all the knowledge that the computer needs (Russell and Norvig, 2009). It is a data driven approach.

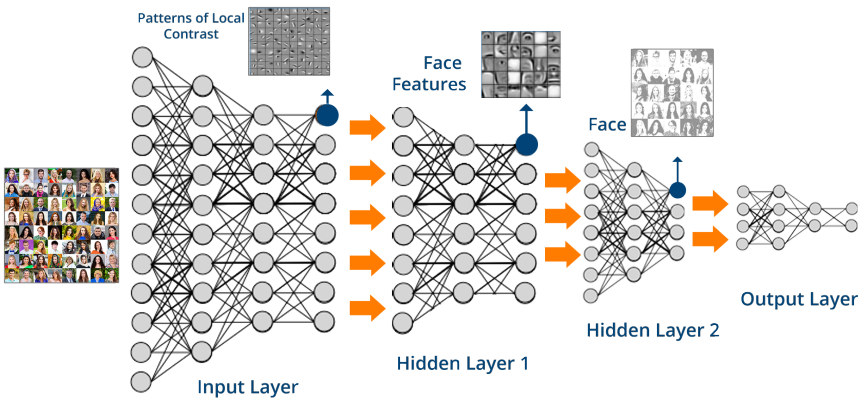


Figure 3.1: *Deep learning architecture(Olah, 2015)*

The emergence of Deep Learning has completely changed the potential of Machine Learning techniques. It is a particular form of Artificial Neural Networks, where a hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other (Figure 3.1), the graph is deep, with many layers, hence the deep learning name. Deep learning algorithms have been developed and tested widely in autonomous vehicle perception and navigation. They have replaced classical model based methods and are preferred in Autonomous vehicle navigation. Image segmentation, clustering, classification and other tasks related to autonomous driving have been solved using these networks. They are also used in end-to-end driving architectures which output the steering angle directly without going through the classical steps of detection, classification, tracking, control etc. (Zhou and Tuzel, 2018).

In the next sections, important deep neural network models, used in this thesis, are explained and their applications are discussed. Convolutional Neural Networks, Recurrent Neural Networks, Long Short Term Memories and Gated Recurrent Units are the building blocks of most of the architectures mentioned in our work. The datasets available for Autonomous Vehicle navigation are also discussed in this chapter.

3.1 Convolutional Neural Networks (CNN)

The most basic Neural Networks architecture receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each is fully connected to all neurons in the previous layer and where neurons in a single layer function independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings, represents the class scores (Dertat, 2017).

3.1.1 Structure

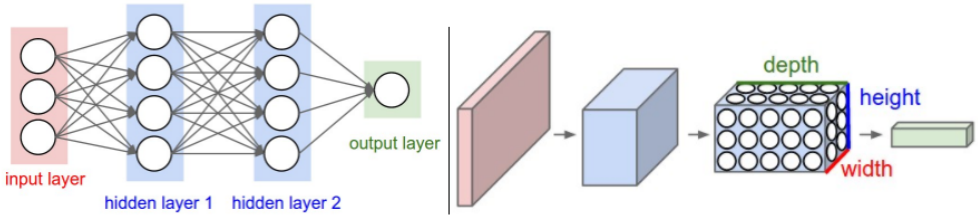
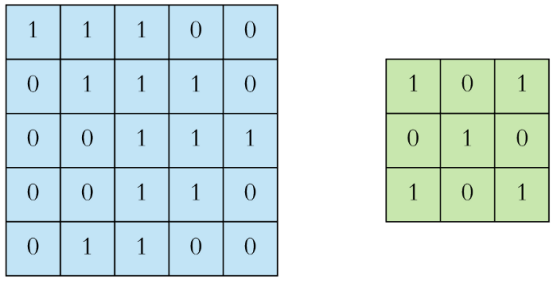
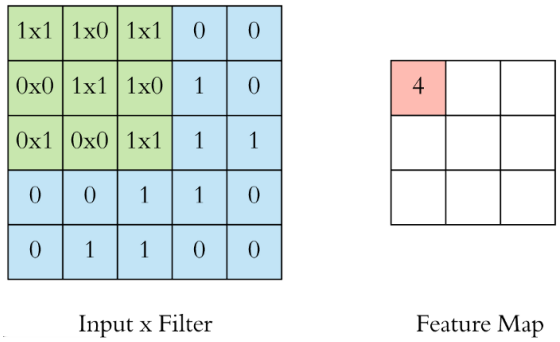


Figure 3.2: *Left: A regular 3-layer Neural Network. Right: A CNN arranges its neurons in three dimensions (width, height, depth), as visualised in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)(Karpathy and Li, 2015).*

For image processing applications (and more generally for spatially arranged inputs), Convolutional Neural Networks (CNN or ConvNet) take advantage of the fact that the input consists of images and they constrain the architecture in a structured manner. In particular, unlike a regular Neural Network (Figure. 3.2, left), the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth (Figure. 3.2, right). Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network. The neurons in a layer will only be connected to a small region of the previous layer, instead of all of the neurons in a fully-connected manner (Karpathy and Li, 2015).



(a) *Input and Kernel for a CNN*



(b) *Kernel multiplication and Feature map*

Figure 3.3: *(a) shows the Input matrix and the Kernel. (b) illustrates the calculation of the feature map by sliding the kernel over the input matrix. Illustrations from Dertat (2017)*

The core component of a CNN is the convolutional layer. Convolution is a mathematical operation which merges two sets of information. A convolution operation is done on the input using a kernel or a convolution filter to produce a feature map. The input is a matrix and the kernel is another matrix which is slid over the input in each location as shown in Figure 3.3. Stride S is the number of elements that we slide the kernel over the input. At each location, an element wise multiplication is performed and the results are summed. This gives us the value for the feature map for that particular location. Sliding the kernel over the whole input gives us the complete feature map. There exist 3D CNNs also, where the kernels move through three dimensions of data (height, length, and depth) and produce 3D activation maps. This is used for example for processing point-cloud data.

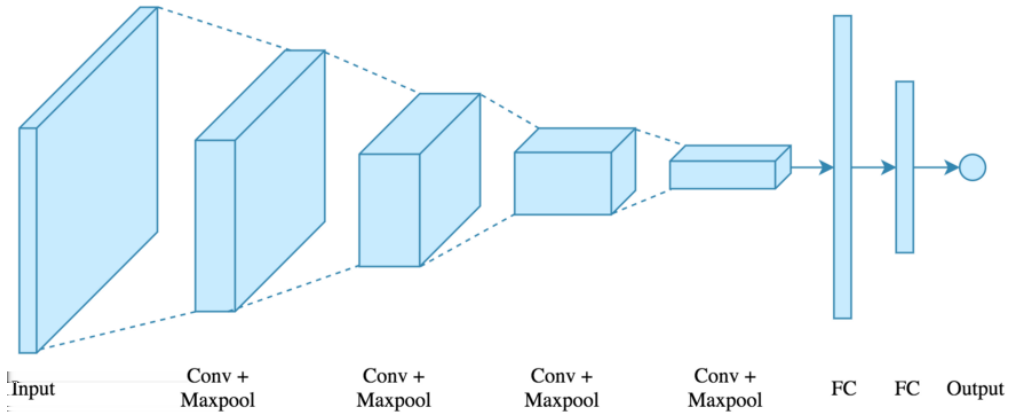


Figure 3.4: *A CNN architecture with 4 convolution + pooling layers, followed by 2 fully connected layers. Input is an image and the output is a score for the class. Illustrations from Dertat (2017)*

In a complete CNN model, as shown in Figure 3.4, after each convolution layers, a Reclified Linear Unit (RELU) layer will apply an elementwise activation function, such as the max(0,x) function, which thresholds the negative values at zero. It leaves the size of the input volume unchanged. Then to reduce the dimensions, a pooling operation is done. This reduces the number of parameters, which in turn shortens training time and avoids overfitting. Pooling layers downsample each feature map independently, reducing their height and width, keeping the depth intact.

The final fully-connected (FC) layers compute the class scores. As with ordinary Neural Networks, each neuron in these layers will be connected to all the neurons in the previous layer. These layers can learn to approximate different functions, however since they contain multiple connections, they will have a high number of parameters. Their training is then more time consuming and energy consuming than convolution layers. An FC layer of size N is represented by the following function:

$$\mathbf{o} = W * \mathbf{h} + b \tag{3.1}$$

with h the input vector of size H, W a weight matrix of size $H * N$ and

b the bias vector of size N.

As an illustrative example, the input images used in CIFAR-10 image classification challenge (Krizhevsky, 2009) consists of a volume of activations, that is a matrix of dimensions 32x32x3 (i.e, width, height and 3 colour channel for the depth of the image). A convolutional layer on this input will transform it into 32x32x12 if 12 filters are used. A pooling layer can downsample this to 16x16x12 and finally an FC layer will compute the class scores, resulting in volume of size 1x1x10. The CNN architecture used in this example, ConvNet reduces the full image into a single vector of class scores. These are arranged along the depth dimension. In this example, there are 10 class scores which classify the image.

3.1.2 Guidelines for the design of CNN Architectures

The literature provides some general guidelines to help choose the parameters of a CNN Architecture. This include:

The input layer (e.g. an image) should be divisible several times by 2. This includes 32 (e.g. CIFAR-10), 64, 96 (e.g. STL-10), or 224 (e.g. common ImageNet ConvNets), 384, and 512 (Olah, 2015). The convolutional (CONV) layers should use small filters (e.g. 3x3 or at most 5x5), using a stride of 1. The pool layers are in charge of downsampling the spatial dimensions of the input. The preferred setting is to use max-pooling with 2x2 receptive fields (kernel size) (i.e. F=2), and with a stride of 2 (i.e. S=2). The preferred method to build a Convolutional Net architecture is to stack a few CONV-RELU layers, follow them with POOL layers, and repeat this pattern until the image has been merged spatially to a small size.

As an example of a typical CNN architecture, the runner-up in 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC'14) was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. This has become the most widely applied architecture. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and features an extremely homogeneous architecture that only performs 3x3 convolutions and

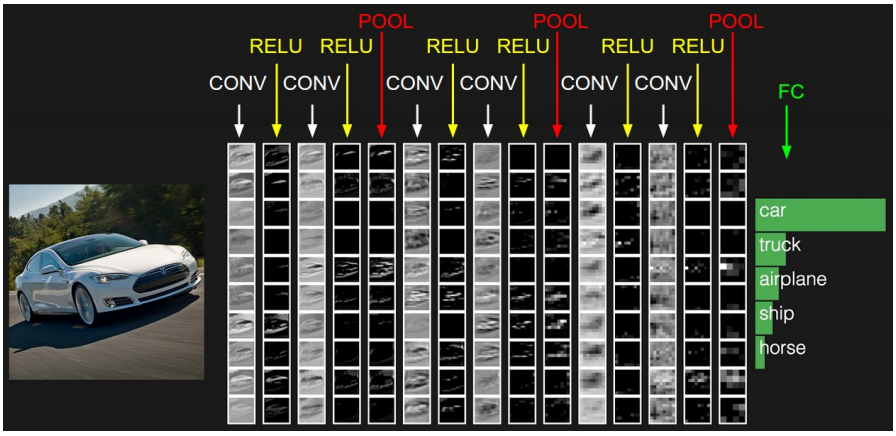


Figure 3.5: *A ConvNet architecture where 5 class scores are shown out of the 10. This is a tiny VGG Net. Illustrations from Karpathy and Li (2015)*

2x2 pooling from the beginning to the end. This is shown in Figure 3.5.

3.1.3 Applications

Convolutional Neural Networks are adopted in a variety of applications such as face recognition, scene labelling, image classification, action recognition, human pose estimation, semantic segmentation etc. In the domain of autonomous driving, which heavily depends on cameras and LiDARs, this has become an indispensable part of the software architecture. They work well for image segmentation and classification, however for other applications, they must be combined with other neural networks such as LSTMs. This is the case for trajectory prediction. The most known convolutional network architectures are LeNet, AlexNet, GoogLeNet, VGGNet and ResNet (Karpathy and Li, 2015).

As relevant application to autonomous vehicle, Treml et al. (2016) propose a deep network architecture for image segmentation that keeps the high accuracy while being efficient enough for embedded devices. Figure 3.6 illustrates the network architecture and Figure 3.7 shows the segmentation results. It consists of Exponential Linear Units (ELU) activation functions, an encoder, followed by parallel dilated convolutions, and a decoder. When applied to the Cityscapes dataset, the network achieves higher segmentation accuracy than others that are

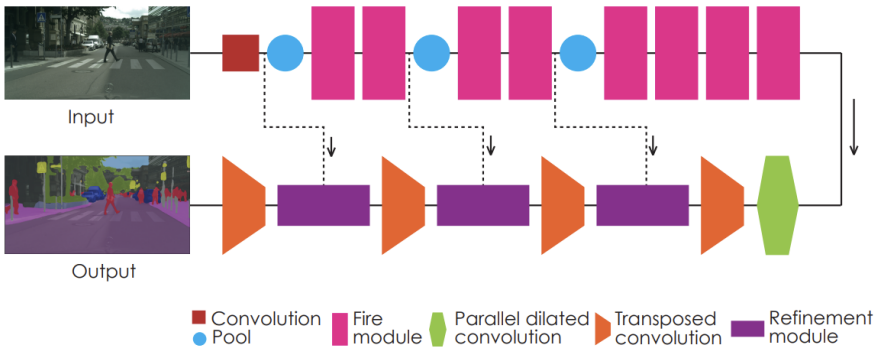


Figure 3.6: *Architecture of network for semantic segmentation proposed in Treml et al. (2016).*

tailored to embedded devices. The frame-rate is sufficiently high for application in autonomous vehicles.

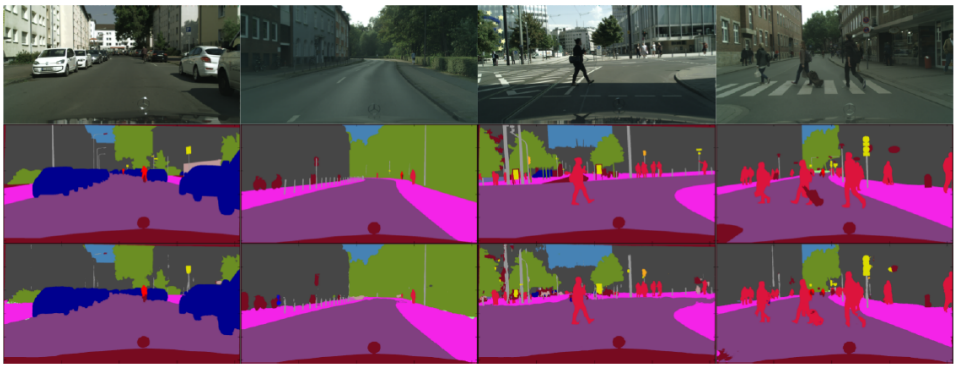
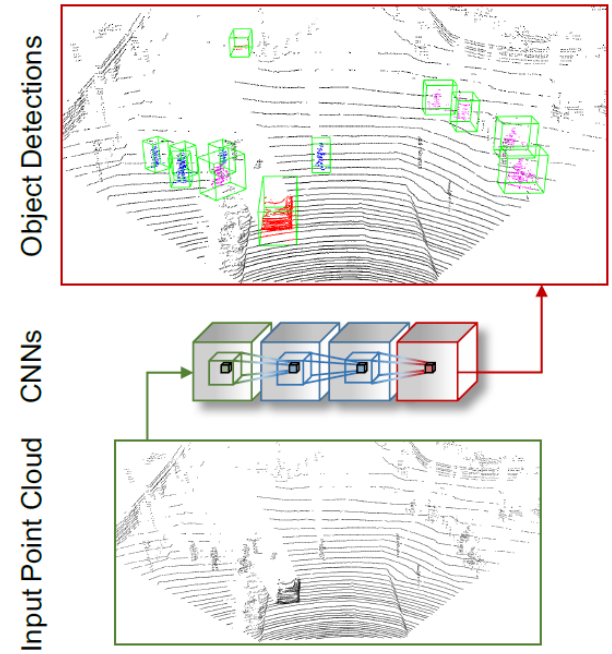


Figure 3.7: *Example output (bottom) of the network with ground truth (centre) on images from the Cityscapes validation set. Treml et al. (2016)*

Convolutional architectures can also be applied to LiDAR point clouds for object detection in autonomous vehicles (Engelcke et al., 2017). They achieve object detection in point clouds at fast speeds with 3D CNNs constructed from sparse convolutional layers, based on a voting scheme. A state of the art was established on the KITTI benchmark. The architecture known as Vote3Deep outperforms methods that use data from both point clouds and images in most test cases. Figure 3.8 illustrates this architecture.



(a) 3D point cloud detection with CNNs

(b) Reference image

Figure 3.8: *(a) - Results of applying Vote3Deep to an unseen 3D point cloud (KITTI dataset). The model detects cars (red), pedestrians (blue), and cyclists (magenta), even at long range. It assigns the respective bounding boxes (green). (b) - Reference image (Engelcke et al., 2017)*

3.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) help in retaining information in memory. Just like when humans think, we use information from the past to make judgements, the RNNs can retain information from the past and use it later. These networks have loops, which allows information from previous updates to persist.

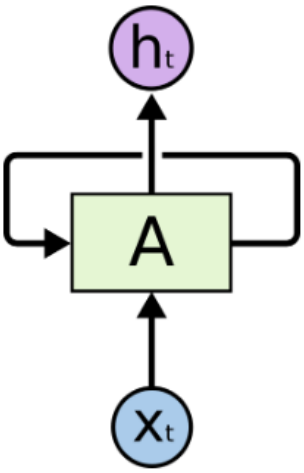


Figure 3.9: *Architecture of an RNN (Olah, 2015)*

Mathematically, RNNs can be represented as follows:

Given the input sequence $[\mathbf{x} = \mathbf{x}_1, ..., \mathbf{x}_T]$ of length T, a single RNN is formed by a repeated application of a function $\mathbf{f_h}$. This generates a hidden state $\mathbf{h_t}$ for each time step t.

$$\mathbf{h}_t = \mathbf{f_h}(\mathbf{x}_t, \mathbf{h}_{t-1}) = \sigma(\mathbf{x_t}\mathbf{W_h} + \mathbf{h_{t-1}}\mathbf{U_h} + \mathbf{b_h}). \tag{3.2}$$

for some non-linearity σ. The *weight matrix* $\mathbf{W}$ and the *bias* $\mathbf{b}$ define different linear transformations. The model output can be defined with different approaches. For example, it can be a function of the hidden state such as:

$$\mathbf{y} = \hat{\mathbf{f}_{\mathbf{y}}}(\mathbf{h_T}) = \mathbf{h_T}\mathbf{W_y} + \mathbf{b_y}. \tag{3.3}$$

A recurrent neural network can be thought of as multiple copies of the same network, each passing its hidden state to a successor. In the last few years, RNNs have been incredibly successful in solving a variety of problems such as speech recognition, language modeling, translation, image captioning etc. RNNs are theoretically designed to encode long-term dependencies however these haven't solved the problem completely. Hochreiter and Schmidhuber (1997) [German] and Bengio et al. (1994) have explored the fundamental reasons why RNNs do not achieve this. By extending the concept, a new architecture has been developed, namely Long Short-Term Memory networks or LSTMs to solve this problem.

3.2.1 Long Short Term Memories (LSTM)

LSTMs are a recurrent network architecture in conjunction with an appropriate gradient based learning algorithm. It is designed to overcome the error backflow problems incurred by RNNs (Graves et al., 2013; Hochreiter and Schmidhuber, 1997).

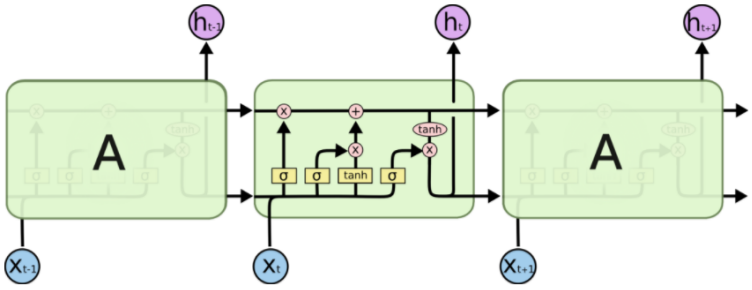


Figure 3.10: *LSTM Architecture(Olah, 2015)*

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single *tanh* layer. LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a

single neural network layer, there are four, interacting in a very special way. This is shown in Figure 3.10. The LSTM has the ability to remove or add information to a cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed of a sigmoid neural net layer and a point-wise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through", while a value of one means "let everything through!". LSTM is defined by setting four gates namely "input", "forget", "output", and an "input modulation gate".

The four gates $(\underline{\mathbf{i}}, \underline{\mathbf{f}}, \underline{\mathbf{o}}, \underline{\mathbf{g}})$, the cell state $(\mathbf{c}_t)$ and hidden state $(\mathbf{h}_t)$ are described by the following equations:

$$\underline{\mathbf{i}} = \sigma(\mathbf{x}_t \mathbf{W}_i + \mathbf{h}_{t-1} \mathbf{U}_i) \tag{3.4}$$

$$\underline{\mathbf{f}} = \sigma(\mathbf{x}_t \mathbf{W}_f + \mathbf{h}_{t-1} \mathbf{U}_f) \tag{3.5}$$

$$\underline{\mathbf{o}} = \sigma(\mathbf{x}_t \mathbf{W}_o + \mathbf{h}_{t-1} \mathbf{U}_o) \tag{3.6}$$

$$\underline{\mathbf{g}} = tanh(\sigma(\mathbf{x}_t \mathbf{W}_g + \mathbf{h}_{t-1} \mathbf{U}_g)) \tag{3.7}$$

$$\mathbf{c}_t = \underline{\mathbf{f}} \odot \mathbf{c}_{t-1} + \underline{\mathbf{i}} \odot \underline{\mathbf{g}} \tag{3.8}$$

$$\mathbf{h}_t = \underline{\mathbf{o}} \odot tanh(\mathbf{c}_t) \tag{3.9}$$

where $\{\mathbf{W}_i, \mathbf{U}_i, \mathbf{W}_f, \mathbf{U}_f, \mathbf{W}_o, \mathbf{U}_o, \mathbf{W}_g, \mathbf{U}_g\}$ represent the weight matrices and σ the sigmoid non-linearity. The cell state $\mathbf{c}_t$ (also referred to as an internal state) is updated in an additive manner.

The work by Milan et al. (2016) on End-to-end Multi Object Tracking (MOT) is a good example of the use of LSTM applied to the autonomous vehicles. The approach is based on recurrent neural networks (RNNs) for temporal prediction and update as well as track management. The combinatorial problem of data association for each frame is solved via LSTMs. The resulting network is shown in Figure 3.11.

The work by Milan et al. (2016) involves machine learning for prediction, data association, state update and initiation and termination of targets. Therefore,

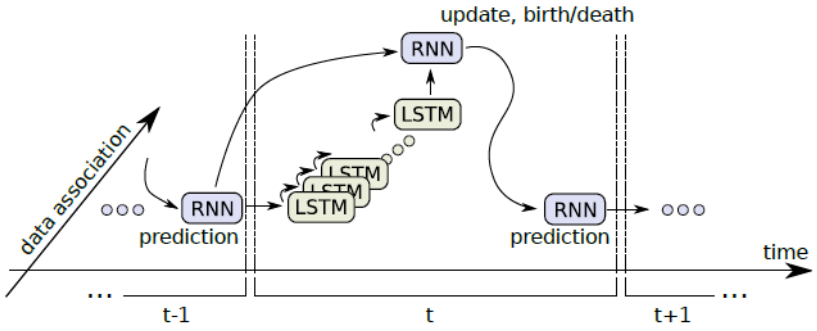


Figure 3.11: *RNN Tracking architecture in Milan et al. (2016)*

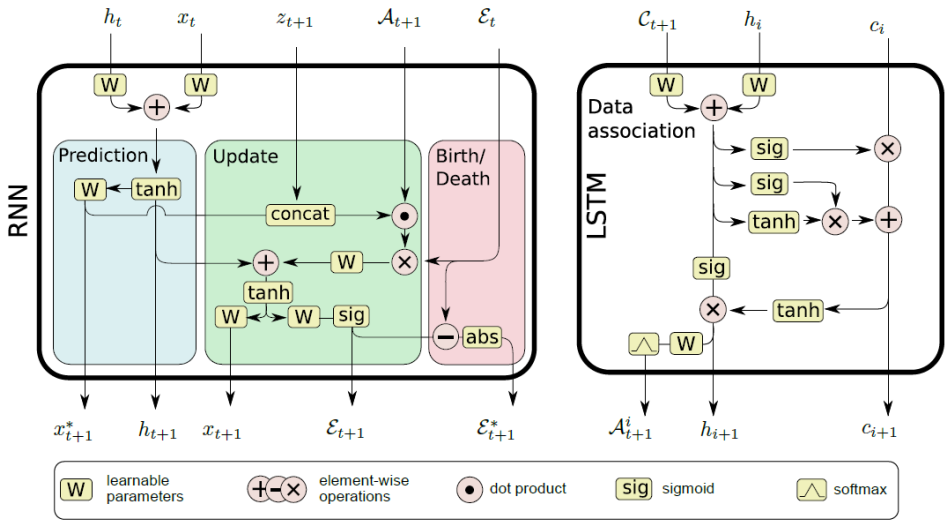


Figure 3.12: *RNN-LSTM architecture used in Milan et al. (2016). Left: An RNN-based architecture for state prediction, state update, and target existence probability estimation. Right: An LSTM-based model for data association.*

is a completely model-free approach. This is illustrated in Figure 3.12. The difficult part in a tracking algorithm is data association, the problem of assigning each detection/measurement to an object or target. The RNN module of this algorithm deals with prediction and update of the state of the targets. It also predicts the birth/death of each target using a Bayesian state estimation method. For this

purpose, it takes as inputs, the current state (x_t), existence probabilities (E_t), measurements vector (z_{t+1}) and data association A_{t+1}. The resulting outputs are - the vector of predicted states for all targets (x_{t+1}), a vector of all updated states (x^*_{t+1}), a vector of probabilities indicated for each target how likely it is a real trajectory (E_{t+1}), and the absolute difference of E_{t+1} to E_t given by E^*_{t+1}. The LSTM module predicts the data association for each set of measurements. This module takes as an input the pairwise-distance matrix $C \in R^{N \times M}$, where $C_{ij} = \left\| x^i - z^j \right\|_2$ is the Euclidean distance between the predicted state of target i and measurement j. This approach by Milan uses the MOTChallenge sequences for training and testing. The results are shown in Figure 3.13.

Figure 3.13: *RNN Tracking output on the MOTChallenge sequence. The colour of each bounding box indicates the person identity (Milan et al., 2016).*

3.2.2 Gated Recurrent Units(GRU)

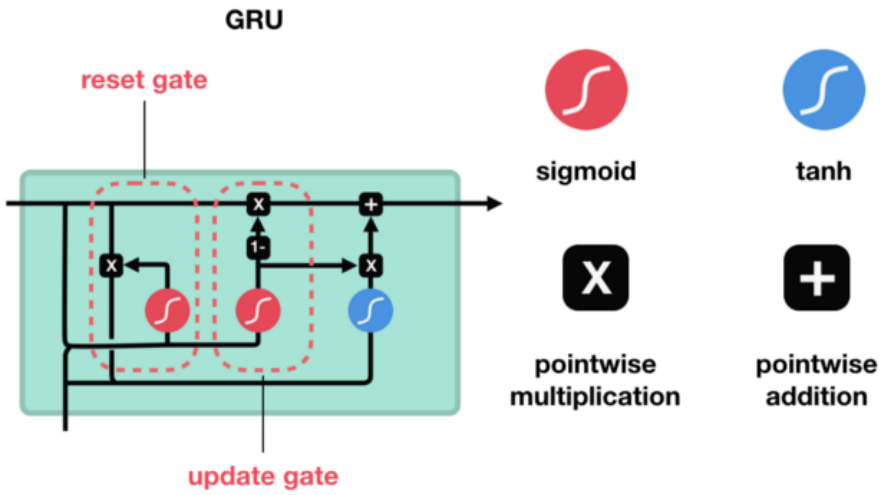


Figure 3.14: *Gated Recurrent Unit Architecture (GRU, 2018)*

Gated Recurrent Unit is another type of RNN which is very similar to an LSTM. It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes. GRUs adaptively capture dependencies of different time scales. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell. This is illustrated in Figure 3.14.

The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add. The reset gate is another gate used to decide how much past information to forget. GRUs have fewer tensor operations; therefore, they are a little faster to train than LSTMs. There isn't a clear winner which one is better. It later chapters, we choose LSTMs as it gives better results in our experiments.

3.3 Datasets

The availability of large and precise datasets is important to obtain good performance for all data-driven methods. In this section, we present an overview of the main datasets used for autonomous driving, considered in our work.

KITTI Vision Benchmark dataset (KITTI) (Geiger et al., 2013): KITTI is one of the most well-known datasets when it comes to autonomous vehicle research. The tasks focused by this dataset are: stereo, optical flow, visual odometry, 3D object detection, and 3D tracking. For this purpose, they equipped a standard station wagon with two high-resolution colour and grayscale video cameras. Accurate ground truth for the perceived objects is provided using a Velodyne 3D-LiDAR scanner and a GPS-based localisation system. The datasets are captured by driving around the mid-size city of Karlsruhe (Germany), in rural areas and on highways. Up to 15 cars and 30 pedestrians are visible per image. In total, it provides 6 hours of driving data collected in both rural and highway traffic scenarios around Karlsruhe. The dataset is provided under the Creative Commons Attribution-Non Commercial Share Alike License. Examples for annotated data in this dataset and the vehicle used to record it is shown in Figure 3.15.

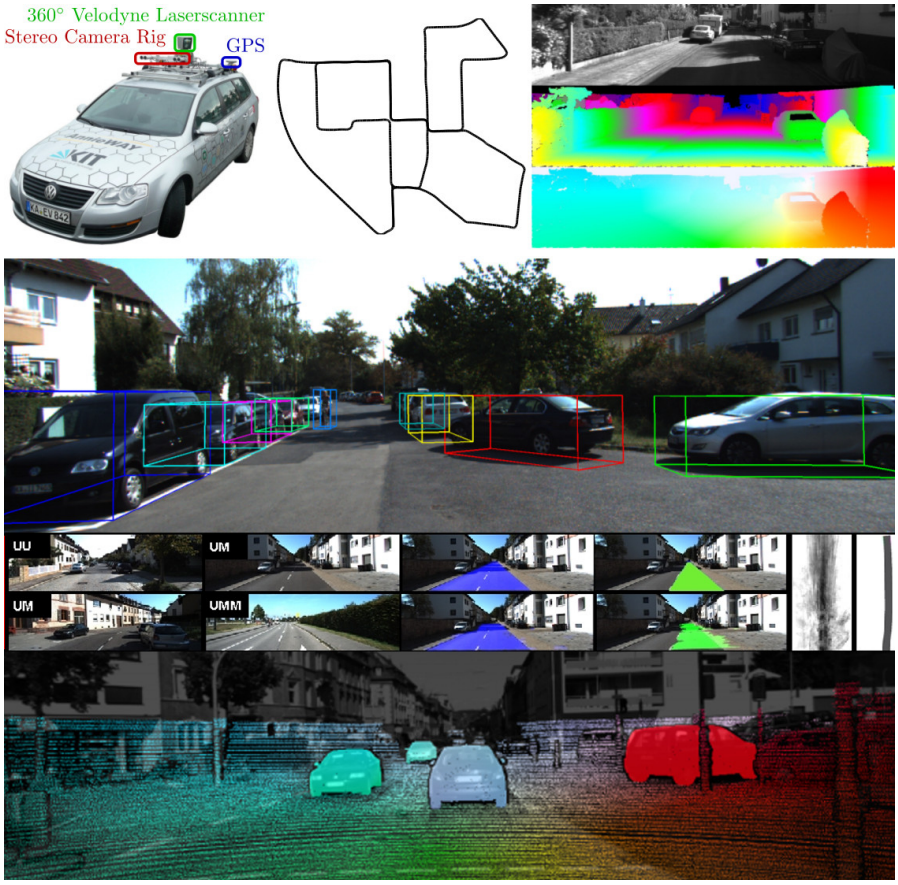


Figure 3.15: *Kitti Dataset: The vehicle used to record the dataset and some annotated camera and 3D LiDAR point-cloud data (Geiger et al., 2013).*

NuScenes dataset (Caesar et al., 2019): nuScenes claims to be the first large-scale dataset to provide data from the entire sensor suite of an autonomous vehicle (i.e., 6 cameras, 1 LiDAR, 5 RADAR, GPS, IMU). In 2019, the full dataset with 1,000 scenes was released. It includes approximately 1.4M camera images, 390k LiDAR sweeps, 1.4M RADAR sweeps and 1.4M object bounding boxes in 40,000 keyframes. Additional features with a map extension, more raw data is being released sequentially. They include 1000 driving scenes in Boston and Singapore, two cities that are known for their dense traffic and highly challenging driving situations. The period of each scene is 20 seconds. They are manually selected to show a diverse and interesting set of driving manoeuvres, traffic situations and unexpected behaviours. To facilitate common computer vision tasks, such as

object detection and tracking, they annotate 23 object classes with 3D bounding boxes at 2Hz over the entire dataset. An extension of the dataset, suited for trajectory prediction tasks was published in late 2020. Examples for annotated data in this dataset is shown in Figure 3.16.

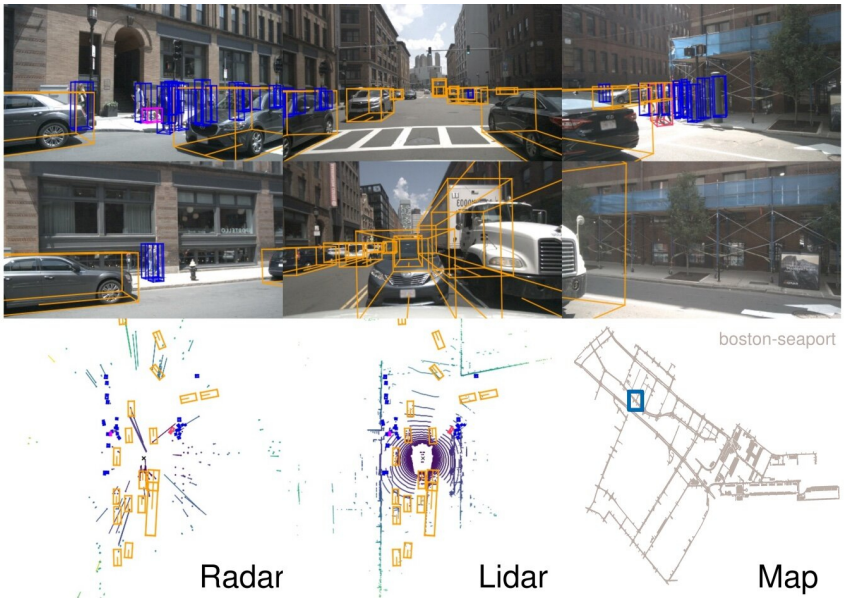


Figure 3.16: *Nuscenes Dataset: Annotated camera images, RADAR, LiDAR and Map data from the Nuscenes dataset (Caesar et al., 2019)*

ApolloScape Open Dataset (Wang et al., 2019): This is a large dataset that supports the training and evaluation of vision-based autonomous driving algorithms and systems. It includes various tasks, e.g. 3D reconstruction, self-localisation, semantic segmentation, instance segmentation, trajectory prediction, detection and tracking. Their acquisition system consists of two laser scanners, up to six video cameras, and a combined IMU/GNSS system. Organised into separate sets under the names Trajectory dataset, 3D Perception LiDAR Object Detection and Tracking dataset, it includes about 100K image frames, 80K LiDAR point cloud and 1000km trajectories in urban traffic. Apolloscape consists of varying conditions and traffic densities which include many challenging scenarios where vehicles, bicycles, and pedestrians move among one another.

Ford campus vision and LiDAR dataset (Ford) (Pandey et al.,

2011): This dataset is collected using an autonomous ground vehicle test-bed, a modified Ford F-250 pickup truck. The vehicle is outfitted with a high-end localisation system and inertial measurement unit (Applanix POS-LV and Xsens MTi-G), a Velodyne 3D-LiDAR scanner, two forward looking Riegl LiDARs, and a Point Grey omni-directional camera system. Approximately 100 gigabytes of data was recorded around the Ford Research campus and downtown Dearborn (Michigan) since 2009. The Ford campus vision and LiDAR dataset is well suited to test simultaneous localisation and mapping (SLAM) algorithms and different autonomous driving functions.

Udacity dataset (Higgins, 2017): It includes 223 gigabytes of image frames and log data from 70 minutes of driving in Mountain View (USA) on two separate days, with one day being sunny, and the other overcast. The Udacity dataset was made open source in 2016 for research purposes. The vehicle sensor setup consists of monocular colour cameras, GPS and IMU sensors, as well as a Velodyne 3D-LiDAR. Vehicle information like steering angle and other variables was recorded during the test runs by a human driver. The labelled data includes bounding boxes for cars, trucks and pedestrians and steering angle for the driving.

Cityscapes dataset (Cordts et al., 2016): This dataset was developed by Daimler AG R&D, Max Planck Institute for Informatics (MPI-IS), and TU Darmstadt Visual Inference Group, in Germany. The Cityscapes Dataset focuses on semantic understanding of urban street scenes, the reason for which it contains stereo vision colour images. The diversity of the images is very large: 50 cities, different seasons (spring, summer, fall), various weather conditions and different scene dynamics. There are 5000 images with fine annotations and 20,000 images with coarse annotations of the semantic labels. The Cityscapes dataset has been used for two important challenges that test and benchmark the development of semantic segmentation and instance segmentation algorithms (i.e., PASCAL VOC Challenge (Everingham et al., 2015) and Robust Vision Challenge (Rob, 2018)). An example for the annotation is shown in Figure 3.17

The Oxford RobotCar dataset (Maddern et al., 2017b): This dataset was developed by recording data on a route through central Oxford twice a week on average using the Oxford RobotCar platform, an autonomous Nissan LEAF.

Figure 3.17: *Semantic segmentation example from the Cityscapes dataset (Cordts et al., 2016)*

This resulted in over 1000km of recorded driving with almost 20 million images collected from 6 cameras mounted to the vehicle, along with LIDAR, GPS and IMU ground truth. Data was collected in all weather conditions, including heavy rain, night, direct sunlight and snow. The dataset provides Real-time Kinematic (RTK) ground truth for long-term localisation and mapping problems. This dataset focuses on long-term road vehicle autonomy by recording the same route in different conditions and long periods of time.

The Cambridge-driving Labeled Video Dataset (CamVid) (Brostow et al., 2008): This was the first collection of videos with object class semantic labels, complete with metadata. The database provides ground truth labels that associate each pixel with one of 32 semantic classes. Over ten minutes of high quality 30Hz footage is provided using a car-mounted camera, with corresponding semantically labelled images at 1Hz and in part, 15Hz. This dataset was used to test algorithms in domains like multi-class object recognition, pedestrian detection, and label propagation

The Daimler pedestrian benchmark dataset (Flohr and Gavrila, 2013): This dataset fits the topics of pedestrian detection, classification, segmentation, and path prediction. The dataset contains a collection of pedestrian sequences collected from a stationary and moving vehicle. Four different pedestrian motion types are considered: crossing, stopping, starting to walk, and bending-

in. Each sequence has no more than one pedestrian and the pedestrians are not occluded. They provide original stereo pairs (8 bit PGM, 1176x640), calibration data, ground truth (GT) annotations, pedestrian detector measurements and vehicle data (speed, yaw-rate), event tags and time-to-event labels (TTE in frames). Pedestrian data is observed from a traffic vehicle by using only on-board mono and stereo cameras. Recently, the dataset was extended with cyclist video samples captured with the same setup.

Caltech pedestrian detection dataset (Caltech) (Dollár et al., 2009): The Caltech Pedestrian Dataset consists of approximately 10 hours of 640x480 30Hz video taken from a vehicle driving through regular traffic in an urban environment. About 250,000 frames (in 137 approximately minute-long segments) with a total of 350,000 bounding boxes and 2,300 unique pedestrians were annotated. The annotation includes temporal correspondence between bounding boxes and detailed occlusion labels.

Table 3.1 shows a summary of the main features of these datasets as well as the sensors and the driving conditions. Due to the availability of the dataset, initial experiments were done on the Apolloscape dataset. However much information was still missing given the early stages of development at that time. Nevertheless this helped us to familiarise with the early models developed in this research.

Once the NuScenes dataset was available, and upon analysis with respect to all of the referred datasets at the moment of performing our research, the NuScenes dataset was chosen. The decision is based on the availability of synchronised data in the form of labelled images, 3D LiDAR point-clouds and map representations. This is important in the studies we conducted as it mainly depended on multi-sensor, multi-input trajectory prediction. Most of the other dataset that we tried had to be rejected because of lack of sensors and lack of map-information. The availability of these data in a synchronised format is still a big downside in autonomous driving research.

At the time of writing this thesis, richer datasets and updates are being made available to overcome this problem. The most important ones are Waymo

Dataset	Task	Sensor-Setup	Size	Traffic-condition
NuScenes (18)	3D tracking, 3D object detection	Radar, LiDAR, EgoData, GPS, IMU, Camera	345 GB (1000 scenes, clips of 20s)	Urban
KITTI (45)	3D tracking, 3D object detection, SLAM	Monocular cameras, IMU, LiDAR, GPS	180 GB	Urban, Rural
Apolloscape (131)	Self-localisation, semantic parsing, instance segmentation, trajectory prediction, detection and tracking	LiDAR, Cameras, IMU/GNSS		Urban
Udacity (55)	3D tracking, 3D object detection	Monocular cameras, IMU, LiDAR, GPS, EgoData	220 GB	Rural
Cityscapes (26)	Semantic understanding	Colour stereo cameras	63 GB (5 clips)	Urban
Oxford (90)	3D tracking, 3D object detection, SLAM	Stereo and monocular cameras, GPS, LiDAR, IMU	23 TB (133 clips)	Urban, Highway
CamVid (16)	Object detection, Segmentation	Monocular, colour camera	8 GB (4 clips)	Urban
Daimler Pedestrian Dataset (42)	Pedestrian detection, Classification, Segmentation, Path prediction	Stereo and monocular cameras	91 GB (8 clips)	Urban
Caltech (34)	Tracking, Segmentation, Object detection	Monocular camera	11 GB	Urban

Table 3.1: *Comparison of Autonomous Driving Datasets.*

(Sun et al., 2019), Argoverse (Chang et al., 2019) and Lyft Level 5 Dataset (Houston et al., 2020).

3.4 Conclusion

This chapter has introduced the different deep learning models, tools, and architectures that are relevant in trajectory prediction and classification. As a result of our analysis, the ones more suitable to the tracking problem are RNNs and

LSTMs. However, these neural networks by themselves have their own performance limitations. Now, for this purpose, in the next chapters, our own network architectures will be proposed based on the selected networks.

To train the models in the proposed architecture, data is required, which includes annotated 3D LiDAR point-clouds and camera images as well as map information. For this purpose, a dataset that is most appropriate to train this machine learning problem is sought. Different relevant open-source datasets were examined, the most appropriate for our application purposes is the NuScenes. This is the product of an analysis of the dataset and some experimental work.

In Chapter 4, 5 and 6, the details and design of the proposed architectures are described as applied to trajectory prediction and classification of different traffic agents (e.g., Cars, Bicycles and Pedestrians in several scenarios).

Chapter 4

Classification of Traffic-Agents by their Trajectory

Contents

4.1 Introduction

Agents that move around a car can be tracked, and hence their trajectory analysed. The trajectory of a moving object can carry a lot of useful information depending on what is sought. Each class of traffic-agents has its characteristic dynamic behaviour in a traffic scenario. Cars, bicycles and pedestrians move with different patterns when compared to each other. Therefore, their motion trajectories could be used as a feature to distinguish them from one another. As a first familiarisation with the LSTM neural networks, we examine whether or not it is possible to use this idea to classify the observed trajectories of traffic-agents.

The driving scenario present in the nuScenes dataset is used in this study. From this, a set of trajectory classes have been defined, based on the traffic-agent they correspond to. This is done by extracting sequential object positions perceived in the scenario. We limit our study to the three categories of cars, bicycles and pedestrians as the data at hand is limited. There is a lack of long sequences of trajectory data for different classes of traffic-agents. We choose the three aforementioned classes as they are the ones with adequate sequential data. The problem we address is to evaluate whether it is possible to classify these trajectories into their corresponding classes.

Trajectory analysis has been treated in multiple ways and in different contexts throughout literature. In most cases, the goal of trajectory analysis was to detect outliers or to predict future states using data-driven machine learning approaches. There are only a few approaches related to classifying traffic agents by their trajectories that we describe below.

Li et al. (2007) used a rule-based classifier to do hierarchical feature classification. In their work, they presented a technique for anomaly detection of moving objects called ROAM (Rule-and mOtif-based Anomaly detection in Moving objects). In their research, trajectories are expressed as a set of discrete fragments called motifs, which form a multidimensional feature space for every sample. A rule-based classifier is then responsible for a hierarchical exploration of the feature

space to find the effective regions which define an anomaly. Likewise, Lee et al. (2008) presented an outlier detection algorithm based on the partitioning of the trajectory. The concept is similar to the previously cited work, with the difference that in this case, the aim is to detect also the outlier sub-trajectories. Here the detection is done with a hybrid of a distance and a density-based approach. Khosroshahi et al. (2016) presented a work related to vehicle trajectory analysis. In particular, they performed behaviour analysis of surrounding vehicles for autonomous driving using Recurrent Neural Networks. Their aim was to classify the type of manoeuvres of other vehicles with a Long Short Term Memory model using their trajectories. Kumaran et al. (2018) presented work on anomaly detection and trajectory classification on traffic surveillance videos. By using a hybrid Convolutional Neural Network and Variational Autoencoder they were able to detect outliers and classify trajectories with impressive accuracy.

Inspired by the approaches exploiting recurrent neural networks, we propose to classify the type of traffic-agent from their trajectory behaviour using an LSTM model. The motivation is that we focus on the manner in which the object moves rather than on the use of their appearance. While in the autonomous vehicle scenario such classification is of limited interest, because perception of the agents usually gives good cues about its category, it is for us a first approach to study the LSTM model and analyse the effect of some hyper-parameters. The potential future use for such an approach could however be found in different fields. In sports, for example, it could be interesting to recognise a particular style of movement by looking at the trajectory of a sportsman, e.g. in winter sports. Or the observation of movement patterns of wild animals are very interesting tasks that could use the information provided by such an analysis.

In this chapter, Section 4.2 elaborates the problem formulation, describes the dataset and the generation of data for training the models. Section 4.3 explains the methodology involved in the classification of traffic-agent trajectories and and the details about the implementation and training of the models. Finally, Section 4.4.2 discusses the results and concludes the chapter.

4.2 Problem Formulation

4.2.1 Overview

The problem is formulated so as to provide the class of a traffic-agent according to the trajectory in a probabilistic manner. This is applied to three classes, namely - passenger cars, pedestrians and bicycles. The input is described as a sequence of points in Cartesian coordinates, that represents the trajectory of the traffic agents within the driving scenario. These points were obtained through detection and tracking systems applied to data obtained from a camera or LiDAR sensor.

4.2.2 Data Formalisation

The historical trajectory of a traffic-agent k, from time-step 1 to time-step T_{obs} is given by,

$$P^k = [p_1, ..., p_{T_{obs}}], \tag{4.1}$$

where,

$$p_t = [x_t, y_t] \tag{4.2}$$

represents the 2D position of the agent for each time-step t and x_t and y_t are the spatial coordinates.

The input to the LSTM network will be a part of this trajectory sequence, defined by a sliding observation window w which accounts for a sequence from time (T_{Obs-w}) to the current time instance (T_{Obs}). Hence, we have as input:

$$P^k_{T_{obs}} = [p_{T_{Obs-w}}, ..., p_{T_{obs}}], \tag{4.3}$$

where $p_{T_{Obs-w}}$ is the position at time T_{Obs-w} and $p_{T_{Obs}}$ is the position at time T_{Obs} or the current time. The choice of the size of the sliding window is explained in Section 4.4.1.

We focus on three classes of traffic-agents - cars, pedestrians and bicycles represented by class scores *c, p* and *b*. The goal is to find an LSTM architecture plus a classification layer that converts the input trajectory data into class scores of the predefined classes.

4.2.3 Dataset

The data used to train the LSTM model originates from the NuScenes dataset (Caesar et al., 2019). It includes ground truth for traffic-agents annotated from LiDAR and camera data. The nuScenes dataset offers approximately 15h of driving data recorded in Boston and Singapore. Trajectory data is available with annotations at 2 Hz for 20 second sequences, examples are shown in Figures 4.1 and 4.2. However, this data is insufficient to train the LSTM model. Some data manipulation is necessary. Data augmentation techniques like mirroring and flipping are used to generate more data. Further, trajectories which had less than 30 time-instances in the set were removed from the dataset because they were too short to get any meaningful pattern. After this data manipulation, a total of 3120 car trajectories, 2580 bicycle trajectories and 2092 pedestrian trajectories were generated from the nuScenes dataset for training purposes and 780, 645 and 523 trajectories respectively for test purposes.

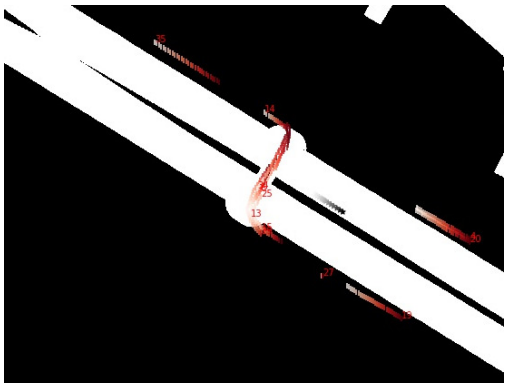


Figure 4.1: *Example trajectories for the pedestrian class (red lines). The black trace in the middle of the road represents the ego-vehicle trajectory. Caesar et al. (2019)*

4.3 Methodology

Sequence prediction problems are generally solved using LSTM encoder-decoder architectures as shown in Figure 4.3. We decided to use their encoder part as a basis for the classification task addressed in this chapter. Indeed, the hidden-state vector of the LSTM models will include information about the trajectory, a feature of the tracked object, and will contain information about the past trajectory and

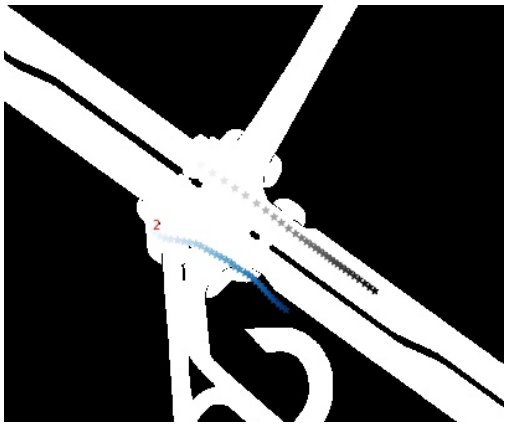


Figure 4.2: *An example trajectory for the bicycle class (blue traces). The black trace represents the ego-vehicle trajectory.*

the dynamics of each observed traffic-agent. As each agent has its own motion characteristics, the LSTM hidden-state could be used to classify them. To find the hidden-state size that stores the most information about a trajectory, a basic LSTM prediction network is trained. The value giving the best prediction performance is then chosen for the classification network. This optimisation is detailed in Section 4.4.1.

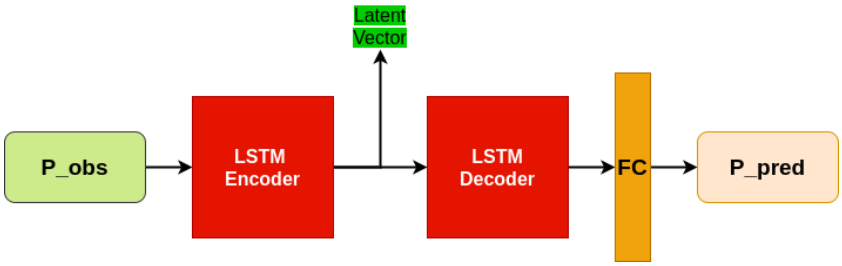


Figure 4.3: *A basic LSTM encoder-decoder trajectory prediction model.*

Following this approach, two different networks will be involved. First a prediction network which is used to find the best hidden state size of the LSTM model. Then a second Classification Network that enables us to classify the trajectories of the observed traffic-agents reusing part of the architecture from the first network.

For both network, the input is taken from the sequential data describing the traffic-agent trajectory as represented by Equation 4.1. This is used as input

to train the LSTM model. For this purpose, a window of time representing several time steps of the observed traffic agent is used. The trajectory sequence is represented by the spatial coordinates of the samples within the window given by Equation 4.3, i.e. a trajectory sequence $P^k_{T_{Obs}}$ from time (T_{Obs-w}) to the current time instance (T_{Obs}).

4.3.1 Prediction Network

For the prediction network, the sequential trajectories $P^k_{T_{Obs}}$ are passed through an LSTM encoder to generate its embedding (hidden-state) as follows:

$$h^k_{T_{Obs}} = LSTM(P^k_{T_{Obs}}, h^k_{T_{Obs-1}}) \tag{4.4}$$

where $h^k_{T_{Obs}}$ is the spatial embedding (i.e. embedding of the position) of traffic-agent k at time T_{Obs}, $P^k_{T_{Obs}}$ is the trajectory of the agent k in the observation window and $h^k_{T_{Obs-1}}$ is the hidden-state vector from the previous step.

Thereafter, the resulting hidden-state latent vector $h^k_{T_{Obs}}$ is passed through an LSTM decoder and a fully connected layer to obtain the predicted trajectory following the equation below:

$$Ppred^k_t = FC(LSTM(h^k_{t-1}, Ppred^k_{t-1})) \tag{4.5}$$

where $Ppred^k_t$ is the predicted trajectory at time t, h^k_{t-1} is the previous hidden-state vector from the previous time-step and $Ppred^k_{t-1}$ is the prediction in the previous time-step.

The predicted trajectory $Ppred^k_t$ can be expressed by:

$$Ppred^k_t = [\hat{x}^1_k, \hat{y}^1_k, \hat{x}^2_k, \hat{y}^2_k, ...\hat{x}^w_k, \hat{y}^w_k] \tag{4.6}$$

The Root-Mean-Square-Error Loss is used to train the Prediction Network:

$$L = \sqrt{\frac{1}{w} \sum_{j=1,2...w} (\hat{x}_k^j - x_k^j)^2 - (\hat{y}_k^j - y_k^j)^2} \tag{4.7}$$

where w is the observation window and x_k^j and y_k^j denote the position coordinates of the k^{th} agent in the predicted time-step j.

An Adam optimiser and a learning rate of $1 \times e^{-4}$ for 100 epochs is used for training the Prediction Network.

4.3.2 Classification Network

This network, shown in Figure 4.4, also starts by passing the sequential trajectories $P_{T_{Obs}}^k$ through an LSTM encoder to generate its spatial embedding (hidden-state). This feature vector $h_{T_{Obs}}^k$, as given by Equation 4.4, is then passed through a linear function representing a fully connected layer in a Neural Network to obtain the classification scores. The output of the network is then a vector giving the class scores for each sequence. Hence, in our case, the vector is of size 3 as there are three classes. The scores c, p and b represent the class of traffic-agent to which the trajectory belongs. The class scores are generated each time-step based on the observation window as in Equation 4.8. The implementation details are elaborated in the next section.

$$c, p, b = FC(LSTM(P_{T_{Obs}}^k, h_{T_{Obs-1}}^k)) \tag{4.8}$$

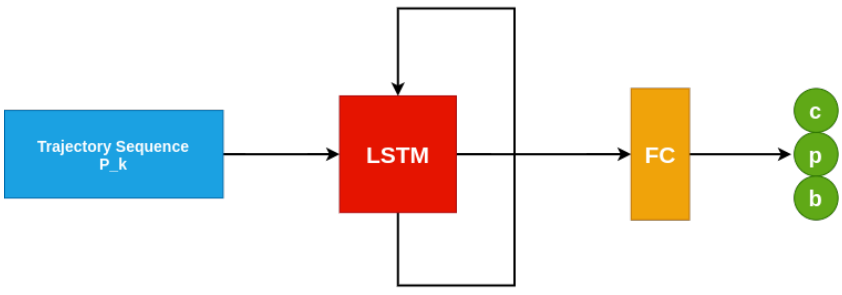


Figure 4.4: *System Architecture for the LSTM based Trajectory Classification. (Class of interest are cars, pedestrians and bicycles, represented by scores c, p and b)*

The loss used to train the classification model is a Cross-Entropy Loss:

$$L = -\Sigma_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{4.9}$$

where M is the number of classes (M=3), y is a binary indicator (0 or 1) if class label c is the correct classification for observation o and $p_{o,c}$ is the predicted probability observation that o is of class c. The final vector of size 3 obtained from here provides the c, p and b scores.

For training the classification network, an Adam optimiser (Kingma and Ba, 2017) with a learning rate of 0.001 is applied for 100 epochs.

4.4 Results

The prediction and classification networks described previously are applied to achieve specific purposes. The former helps in deciding the hidden-state size of the LSTM whilst the latter is used to classify observed trajectories into their specific traffic-agent class. Section 4.4.1 describes the parameter selection and Section 4.4.2 shows the classification accuracy reached using our network.

The experiments are based on the use of the NuScenes dataset and the overall purpose is to classify the trajectories of three traffic-agents - Cars, Bicycles and Pedestrians - as stated before.

4.4.1 Parameter Selection

The rationale behind the selection of the various parameters in the Classification Network is presented in this section. Initially, to select the observation window which also decides the sequence length, results from Fernando et al. (2018b) were studied. They consider 10 time-instances as a long-term prediction horizon. With the dataset we are using - NuScenes, the sampling frequency is 2Hz. Therefore, the 10 time-instances correspond to 5 seconds. This is used as a cue to choose the sequence length for the classification problem. Hence, the observation window w (c.f. Section 4.3) was initially chosen as 10 for the trajectory prediction network. Different windows lengths, from 8 to 20, for the observation window are tested in

the classification evaluation. However, increasing the length of this window implies that it would be less appropriate for real-time trajectory classification.

The hidden-state size with the best performance for prediction is chosen for training the classification network. For this purpose, the LSTM encoder-decoder network (Figure 4.3) is trained with varying hidden-state sizes. The metrics chosen to evaluate the performance of prediction networks are the Average Displacement Error (ADE) and the Final Displacement Error (FDE) as presented in Section 2.3.

To select these parameters, the prediction network is used to predict the trajectories of traffic-agents obtained from the NuScenes dataset, described in Section 4.2.3. Different hidden-state sizes are used to train the network:

$$h = [50, 100, 150, 250, 300, 350] \tag{4.10}$$

The results are presented in Table 4.1. These are used to determine the optimal LSTM hidden-state size. This value is 300 as it produced the least ADE and FDE, performance metrics. Therefore, the FC layer of the classification network will have an input size of $1 * 300$ and an output size of $1 * 3$.

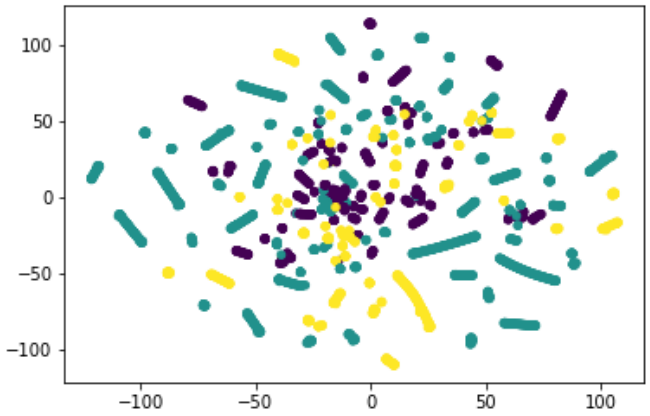
Hidden-State Size	ADE	FDE
50	4.13	8.12
100	3.99	7.86
150	3.56	7.56
250	3.32	5.45
300	**2.23**	**4.54**
350	2.38	4.96

Table 4.1: *Performance of LSTM model with changing Hidden-State Size*

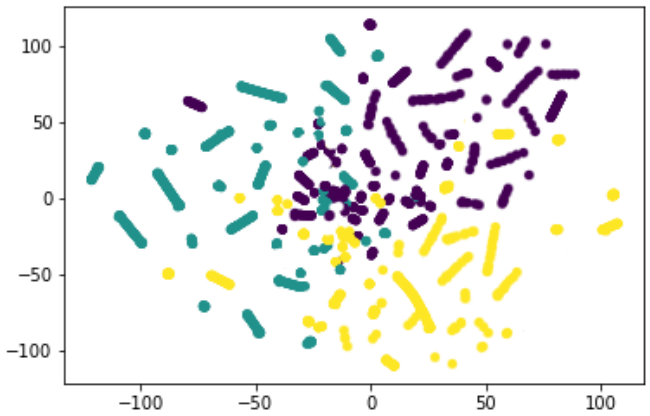
4.4.2 Classification Accuracy

Before training a network to classify trajectories of different traffic-agents, we visually verified the difficulty of the task using the *t-distributed stochastic neighbour*

embedding (t-SNE) approach. This is a statistical method used for visualising high-dimensional data by assigning each data-point a location in a two or three-dimensional map (van der Maaten and Hinton, 2008). This technique is applied to assess if the trajectories of different classes of traffic-agents can be separated from each other. The results are shown in Figure 4.5. The three colours represent the trajectories of cars, pedestrians and bicycles, which are our concern. In Figure 4.5a, it can be observed that the data-points are hard to be separated from each other while in Figure 4.5b they are easily separable. This difference is linked to the short sequence length (8) in the former case and long sequence length (20) in the latter case, making them easier to be classified in the second situation.



(a) *t-SNE plot for trajectories with sequence length 8.*



(b) *t-SNE plot for trajectories with sequence length 20.*

Figure 4.5: *Effect of changing sequence length of the classifiability of trajectories*

To see the effect of changing the observation window length w on the classification accuracy, the classification network is trained with different window sizes. The results shown in Table 4.2, indicate that a window length of 10 offers limited performance, while a length of 20 provides an improvement in performance of 15.7 percent. However, this requires a trajectory observation of 10 seconds which is very long time for vehicle navigation applications. We can see an improvement in classification accuracy of 22.9 percent for bicycles and 20.8 percent for pedestrians while only 7.1 percent for cars. This is because, with a longer observed trajectory of bicycles and pedestrians, there is an improvement in classification.

Observation Window (w)	Cars	Bicycle	Pedestrians	Overall Accuracy
6	52.2	25.3	26.3	34.6
8	57.4	37.4	39.5	44.77
10	83.9	54.8	64.8	67.83
12	84.7	58.7	68.4	70.6
14	86.5	62.7	69.9	73.03
16	87.7	63.4	70.1	73.73
18	87.9	64.3	75.3	75.83
20	**89.9**	**67.4**	**78.3**	**78.53**

Table 4.2: *Effect of varying the observation window length on Trajectory Classification Accuracy*

The trajectory classification with an observation window of 10 frames, resulting from the NuScenes dataset are presented in Table 4.3. The classification accuracy is 89.9% for cars, 67.4% for bicycles and 78.3% for pedestrians. These experiments demonstrate that the classification of traffic-agents by their trajectory is insufficient for safety critical applications, that need very high levels of classification accuracy. The experiments using a sequence length of 10 time-steps or less result in poor accuracy. For instance, when the traffic-agents move at low speeds, it is difficult to differentiate between a car and a bicycle as their trajectories would look similar. This can be observed from the confusion matrix shown in Figure 4.6. When bicycle is the true class, a majority of the samples are classified as cars. Also there are cases where it is hard to separate the trajectories of bicycles and pedestrians. This is also noticeable from the confusion matrix. When 339

samples are correctly predicted to be pedestrians, 168 of it is falsely classified to be bicycles.

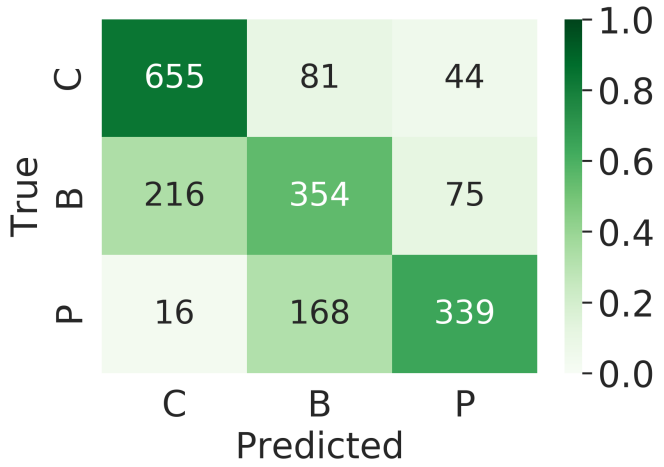


Figure 4.6: *A confusion matrix showing the classification of the three trajectory classes: C - Cars, B - Bicycles, P - Pedestrians.*

Class	Samples	True Classification	False Classification	Precision
Car	780	655	125	83.9
Bicycle	645	354	291	54.8
Pedestrian	523	339	184	64.8
			Overall Precision	67.8

Table 4.3: *Precision of Trajectory Classification using LSTMs on the test set*

The results show that the overall performance of our model based on trajectory alone is insufficient to classify the observed traffic-agents. Using the same sensors, appearance based models provide better results than trajectory-only classification as demonstrated in literature for similar instances (Krizhevsky et al., 2012; Qi et al., 2017).

4.5 Conclusion

In this chapter, a prediction LSTM encoder-decoder pipeline has been developed, and the effect of changing hidden-state sizes on the prediction accuracy has been

analysed. This has established that the hidden-state size of an LSTM is an important hyper-parameter defining the performance of prediction applications. The encoder was then used as the basis of a classification network applied to traffic-agents by observing only their trajectories. While the approach succeeded in classifying the agents with an accuracy of 78%, this performance is low for safety critical reasons. Further, it requires a long observation window (i.e. 10 seconds) which cannot be applicable for autonomous vehicle navigation.

The proposed approach has enabled us to classify traffic-agent trajectories with limited accuracy. For autonomous vehicle navigation, this is not very relevant because agent classification could be better performed using sensor data and applying established methods (Krizhevsky et al., 2012; Qi et al., 2017; Zhang and Rabbat, 2018). Nevertheless, this has resulted in the development of an LSTM encoder-decoder architecture which will be extended in the remainder of the thesis through a mechanism that allows for the incorporation of real-time contextual information. This should allow for the prediction of the trajectories of the observed traffic-agents which is an important parameter that facilitates the situation awareness of the vehicle prior to decision making.

The preliminary results in section 4.4.1 have demonstrated that the accuracy level of the trajectory prediction of traffic-agents is limited. Nevertheless, this can be improved using additional information that is available as the vehicle navigates. The machine learning architecture developed in this chapter will be used as a basis for the proposed trajectory prediction network using contextual information, in Chapter 5. Its application to pedestrian crossing prediction in urban settings is presented in Chapter 6. The focus will be to study how additional information can be fed to the developed model to improve the trajectory prediction. As most intelligent vehicles use HD-maps as part of their functional architectures, we seek to use context cues like driveable areas or road markings stored in HD-maps or observed by intelligent cameras. The incorporation of these cues in the prediction network is presented in the next chapters.

Chapter 5

Context Aided Trajectory Prediction using Map-Masks

Contents

5.1 Introduction

Predicting the trajectory of traffic-agents with which an autonomous vehicle interacts allows it to prepare for future decisions and manoeuvres. It can assess the

risk involved with respect to the decisions taken. The purpose is to enable the vehicle to gain situation understanding so as to allow autonomous navigation in complex conditions. The high level of interactions with other traffic-agents sharing the same work-space and the randomness of the different situations encountered make it difficult to solve the prediction problem with high accuracy. Multiple data inputs, sensors and observations of the situation must be studied to attain the necessary trajectory prediction accuracy of the relevant traffic-agents.

Single-agent sequence prediction problems have been solved using classical model-based techniques as discussed in Chapter 2. However, the use of data-driven methods has been demonstrated to attain better results (Ma et al., 2018; Altché and de La Fortelle, 2018; Messaoud et al., 2021). The behaviour of different types of traffic-agents changes with their class - whether it is a car, a pedestrian or a bicycle. Therefore, the approach taken in this thesis will exploit the specific behaviour of the different traffic-agent classes and predict their individual trajectories. We must recall that the interactions between different traffic-agents play an important role as vehicles navigate. The behaviour of a particular agent will depend heavily on the interactions with the surrounding agents and their positions within the road network. Hence, an interaction-aware model is included into the proposed approach. The behaviour of all traffic-agents is nominally governed by its context. For example, motor vehicles will travel only along paths where they are allowed (i.e., road). In the case of pedestrians, they are expected to walk on curbs, move from one street to the next while walking on road crossings, etc. The areas where traffic-agents can move are strongly associated with the navigation maps. In this thesis, our approach incorporates the use of map information to enhance the prediction of the identified traffic-agents. We introduce the use of map-masks, which are defined as binary maps that provide information on the driveable and non-driveable areas for the different traffic-agents. They provide spatial context for all the observed traffic-agents. A similar work is published by Messaoud et al. (2020) where they propose an approach applying multi-head attention by considering a joint representation of the static scene and surrounding agents.

Trajectory predictions can be regarded as a sequence prediction problem.

This has been solved successfully using machine learning networks like RNNs and LSTMs (c.f. Chapter 2). In this chapter, these are applied as part of our technique. Our approach introduces a machine learning architecture which takes as input - historical trajectories obtained from 3D LiDAR point-cloud data and map information in the form of map-patches as detailed in Section 5.2.2. This approach is trained and tested on the NuScenes dataset (Caesar et al., 2019) which provides all the necessary information which includes 3D bounding-boxes in LiDAR point-clouds for different traffic agents, object/agent IDs and maps of the environment from which we can extract our map-masks.

Using map-patches as input exploits the idea that the manner in which traffic-agents behave is governed by their spatial context. When projected into public road networks, it can be partitioned by areas into which they are usable and non-usable. A contribution of this thesis reside on the use of such spatial information in the form of map segments (patches). It can be regarded as the incorporation of spatial context information to classical machine learning sequence prediction methods.

To summarise, the focus is on the use of map-mask patches to improve the prediction of trajectories for different classes of interacting traffic-agents. The contributions of this chapter are:

- A new LSTM encoder-decoder architecture that uses these masks to make trajectory predictions for different classes of traffic agents in drivable and non-drivable areas
- An evaluation of the proposed model in comparison with classic and LSTM baselines. The performance is superior when applied to single and multi traffic-agent interaction scenarios.

Section 5.2 formulates the trajectory prediction problem with respect to the proposed approach. The formalisation of the inputs is included as these are important for training any machine learning model. Section 5.3 describes the adopted methods and the developed machine learning architecture for the

trajectory prediction of traffic agents. Section 5.4 tabulates the results and finally Section 5.5 concludes the chapter.

5.2 Problem Formulation

5.2.1 Overview

The prediction problem can be expressed as a sequence prediction task. For this purpose, the behaviour of the relevant traffic-agents is observed within the interval $[T_{obs} - w : T_{obs}]$ and the future trajectories in the interval $[T_{Obs+1} : T_{Pred}]$ are predicted. w is the observation window, or the number of time-instances for which the traffic-agent is observed. The process followed for the implementation of this system is shown in Figure 5.1. The inputs to the prediction module are 3D LiDAR point-cloud data and binary map-patches around each observed traffic-agent. From the 3D LiDAR point-cloud data, the historical trajectory of each traffic-agent is observed with the map-patches providing information on the drivable and non-drivable areas in the work-space. These inputs are embedded into a mathematical model so they can be fed to a machine learning model. The encoded sequence is used as input to train the LSTM Encoder which produces a feature vector. From this feature vector the trajectories are predicted using an LSTM decoder. The formalisation of these data inputs to the machine learning model are discussed in the next section.

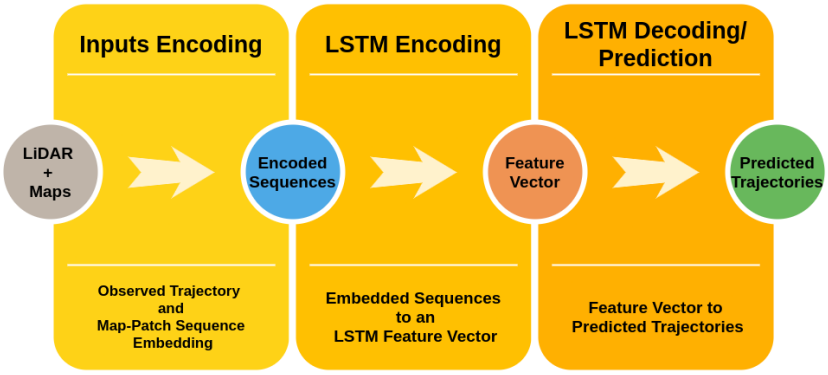


Figure 5.1: *The process involved in the Trajectory prediction module.*

5.2.2 Data Formalisation

In a driving scenario, several objects might be present, sharing the same work-space with the ego-vehicle. These include pedestrians, other vehicles or cyclists. State-of-the-art perception algorithms based on camera images or 3D LiDAR point-cloud data provide bounding boxes/segmentation for each object class. In our approach, we use 3D LiDAR point-cloud data and off-the-shelf detection algorithms to extract the features of all the traffic-agents within the field of view of the sensor used.

For experimental purposes, the PointPillars object-detection network by Lang et al. (2018) is adopted. This network is chosen as it shows superior performances over other frameworks with their code available in the GitHub[1] repository. To predict trajectories during testing phase of the machine learning model, the class and observed trajectory of the perceived agent is extracted from the output of the object detection framework. This sequence is directly obtained from the point-pillars network. However, for training the trajectory prediction network, the ground truth trajectory available in the NuScenes dataset is used.

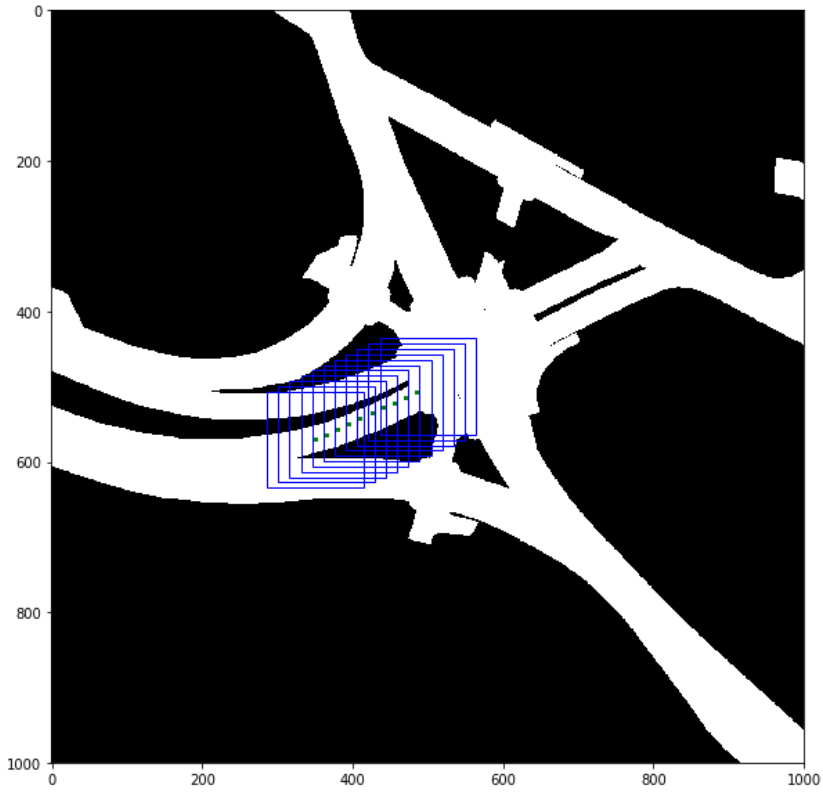
The NuScenes dataset includes HD-maps as used for autonomous driving. This HD maps are converted into a binary map-mask. The drivable area/roads are given an overall value (i.e., white) while the non-drivable areas are given another value (i.e., black). To understand the behaviour of different classes of traffic-agents in the drivable and non-drivable spaces, we use as input the transformed map information. From this we extract square patches of the size 128×128 pixels, which represents approximately 10 meters $\times$ 10 meters, as shown in Figure 5.2b. These square patches are extracted around each detected traffic-agent in the perception space, as shown in Figure 5.2a, of the ego-motion vehicle and for each time-step in the observed time interval. From this data, we define the feature set of each traffic agent k at time t as

$$f_t^k = (p_t^k, c_t^k, M_t^k) \tag{5.1}$$

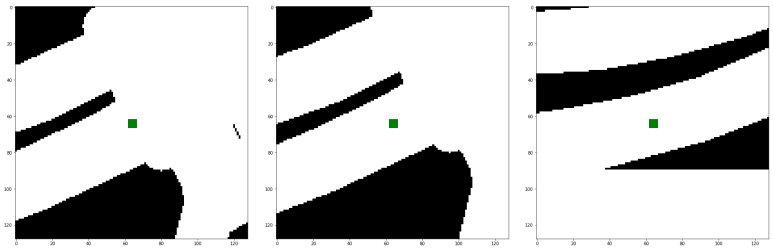
where

$$p_t^k = [x_t^k, y_t^k] \tag{5.2}$$

[1] https://github.com/nutonomy/second.pytorch



(a) *Binary map-mask of a scene*



(b) *Examples of patches centred around the agent*

Figure 5.2: *Binary map-mask and map patches: A trajectory and the map-patches surrounding the agent are marked by blue squares, centred around the traffic agent marked in green*

In these data, x_t^k and y_t^k are the spatial coordinates of the detected traffic-agent obtained from the top-down view of the LiDAR point-cloud detections. c is the class of the traffic agent, $c_i \in (1, 2, 3)$ where 1 is for pedestrians, 2 is for cars and 3 is for cyclists. M_t^k is the 128×128 pixels binary map patch of the area

surrounding each detected object.

The task is to observe the features of all traffic-agents in the interval $[T_{obs} - w : T_{obs}]$ and predict their positions in $[T_{obs+1} : T_{pred}]$.

5.3 Methodology

The system architecture developed for the trajectory prediction of traffic agents takes two inputs - 3D LiDAR point-cloud data and map masks. From the 3D LiDAR point-cloud data, the historical trajectories of observed traffic-agents are extracted using the VoxelNet Object Detection Network. Using this trajectory information, the binary map-patches around each observed traffic agents are extracted for each time-step (Scene Mask Patches Retrieval). These map patch sequences are then encoded using a CNN Encoder. The observed trajectories generated by the VoxelNet are then encoded together with the encoded map patches using the LSTM Trajectory encoder. The resulting feature map is then fed into an LSTM Trajectory Decoder to obtain the Predicted Trajectories. The resulting architecture of this process is shown in Figure 5.3. The details of the different components of this architecture are presented in the next section.

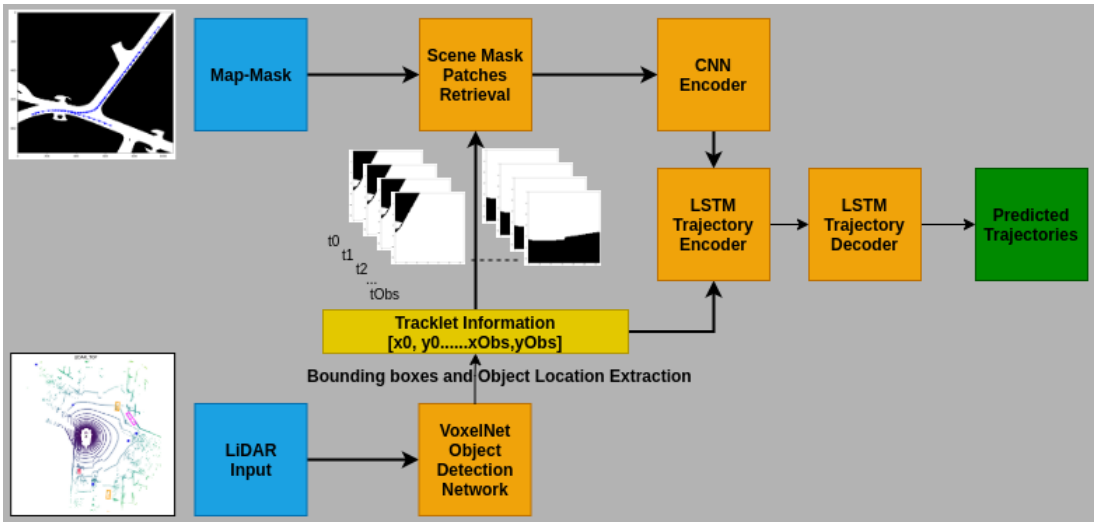


Figure 5.3: *System Architecture: The prediction module has two inputs: sequential historical trajectories of all traffic-agents and the associated map-mask patches of each agent for the observed time instances*

5.3.1 Information encoding using LSTMs

The prediction framework developed by Fernando et al. (2018a) which applied LSTMs for predicting pedestrian trajectory from images, is adapted to use the trajectories obtained from 3D LIDAR point-cloud and to which we provide the additional map context.

Let the historical trajectory of a traffic-agent k, from time-step 1 to time-step T_{obs} be given by,

$$p^k = [p_1^k, ..., p_{T_{obs}}^k], \tag{5.3}$$

where,

$$p_i^k = [x_i^k, y_i^k] \tag{5.4}$$

for each time-step i. These historical trajectories are passed through the LSTM encoder of each respective traffic-agent to generate its historical embedding as follows,

$$h_t^k = LSTM(p_t^k, h_{t-1}^k), \tag{5.5}$$

generating a sequence of historical embedding. Each class of traffic-agent has its own LSTM encoder. For our experiments, the LSTM encoders have a hidden state size of 300 chosen by experimentation and following the recommendations found in Fernando et al. (2018b).

A historical context vector $C_t^{H,k}$ is defined to encode the trajectory information from the traffic-agent of interest (k), which can be computed as a weighted sum of hidden states from $t = [1 : T_{obs}]$:

$$C_t^{H,k} = \sum_{j=1}^{T_{obs}} \alpha_{tj} h_j^k \tag{5.6}$$

and the weight α_{tj} as shown in Bahdanau et al. (2014) can be computed by:

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{l=1}^{T} exp(e_{tl})} \tag{5.7}$$

$$e_{tj} = a(h_{t-1}^k, h_j^k) \tag{5.8}$$

where the function a is a feed forward neural network which is trained jointly with the whole network.

A spatial context vector $C_t^{S,k}$ is used for embedding the neighbouring trajectories. The spatial weights, denoted by w_j^n are computed as:

$$w_j^n = \frac{1}{dist(n,j)} \tag{5.9}$$

where $dist(n,j)$ is the Euclidean distance between the n^{th} neighbour and the traffic-agent of interest at the j^{th} time-step, and w_j^n is the generated spatial attention weight. When there are N neighbouring trajectories in the local neighbourhood, and h_j^n is the encoded hidden state of the n^{th} neighbour at the j^{th} time-step, then the spatial context vector is defined as:

$$C_t^{S,k} = \sum_{n=1}^{N} \sum_{j=1}^{T_{obs}} w_j^n h_j^n \tag{5.10}$$

From these two context vectors, a merged context vector, $C_t^{*,k}$ is computed by concatenating them and applying a non linear function:

$$C_t^{*,k} = tanh([C_t^{H,k}; C_t^{S,k}]) \tag{5.11}$$

For our experiments $C_t^{S,k}$ and $C_t^{H,k}$ has a size of (1,300) and the merged vector at T_{Obs} will have a length of $w \times (1,300)$.

For the baseline LSTM model used to compare with our final model, the merged context is then passed through the LSTM decoder to predict the future trajectory of the traffic-agent of interest (c.f. Section 5.3.3). Alternatively, as presented in the next section, in the proposed method, this context vector is modified using the information from the binary map-masks before it is used as input to the prediction model. Through this approach, contextual information is fed into the model.

5.3.2 Map Input encoding

In this section, the encoding of the map-patches, to obtain context vectors for the LSTM, is explained. Each of the 128×128 pixels binary map patch M_t^k is passed

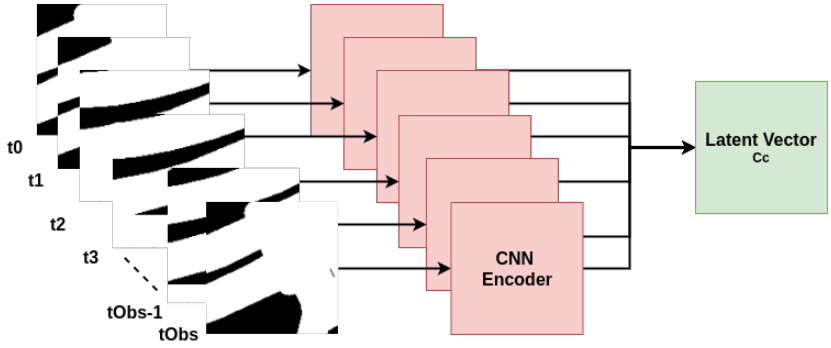


Figure 5.4: *Passing the map-patches through a CNN Encoder generates the latent vector that stores information about the drivable area and this vector is passed on to the LSTM encoder architecture*

through a Convolutional Encoder. The resulting latent vector $C_t^{C,k}$ holds the information about the drivable space associated with the respective traffic-agent k at time-instance t. This concept is illustrated in Figure 5.4. These sequential latent vectors are concatenated to form the Map Context vector corresponding to the observed traffic-agents.

This Map Context vector is then concatenated with the merged context vector $C_t^{*,k}$ from the LSTM Model to obtain a new latent vector $C_t^{M,k}$:

$$C_t^{M,k} = [C_t^{*,k}; C_t^{C,k}] \tag{5.12}$$

Layer Name	Layer Type	Input Size	Output Size
Conv1	ReLu(MaxPool2d(Conv2d(input)))	$1 \times 128 \times 128$	$16 \times 21 \times 21$
Conv2	ReLu(MaxPool2d(Conv2d(input)))	$16 \times 21 \times 21$	$8 \times 10 \times 10$
FC	ReLu(Linear(input))	1×800	1×300

Table 5.1: *Layers of the CNN Encoder - Convolutions have a kernel size of* 3×3*, maxpool has a kernel size of* 2×2*. Parameters not mentioned are default parameters in Pytorch library (in torch.nn).*

The CNN encoder has two convolutional layers and one fully connected (FC) layer as shown in Table 5.1. The stride for Conv1 and Conv2 layers are empirically set to 3 and 2 respectively and a padding of 1 is set for both the layers. The layers are initialised with Xavier init method from the PyTorch library

(Glorot and Bengio, 2010). The output size of the FC layer is set to (1×300), to be compatible with the size of the LSTM hidden cell, as this will be concatenated with it. Therefore, at time T_{Obs}, the length of the merged vector will be equal to observation window $w \times (1, 300)$. Conv1 converts the map-mask to a volume of 16 in thickness. And Conv2 reduces it further to 8. From this we obtain the final output of (1,300) through the FC layer.

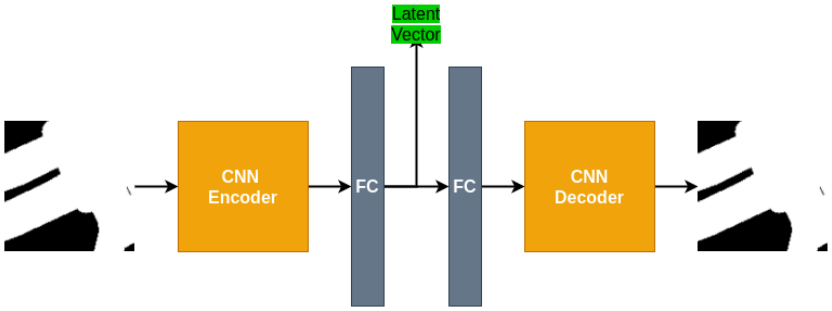


Figure 5.5: *The CNN Autoencoder architecture. The latent vector is extracted from the FC later at the end of the encoder side.*

In order to facilitate the LSTM training, we tested the pre-training of this CNN encoder using an Autoencoder. This is illustrated in Figure 5.5. For this, we append a CNN decoder to the CNN encoder and train it to reconstruct the map patches used as input. The CNN decoder layer has an architecture mirroring the one of the encoder as outlined in Table 5.2. It also utilises two layers, Conv1 with a stride of 3 and Conv2 with a stride of 2, both having a padding of 1. These layers are also initialised with Xavier init method. The CNN Autoencoder is pre-trained and then the CNN Encoder is inserted into the prediction network.

It was found experimentally that training this architecture end-to-end with the prediction module works faster than using the pre-trained CNN encoder (c.f. Section 5.3.4).

5.3.3 LSTM Decoder for Trajectory Prediction

The next step is to obtain the final predicted trajectory q_t^k. This is done by passing the context vector $C_t^{M,k}$ in the case of map input through the LSTM decoder and

Layer Name	Layer Type	Input Size	Output Size
FC	ReLu(Linear(input))	800×300	800×800
Conv1	ReLu(ConvTranspose2d(input))	1×800	$16 \times 21 \times 21$
Conv2	ReLu(ConvTranspose2d(input))	$16 \times 21 \times 21$	$1 \times 128 \times 128$

Table 5.2: *Layers of the CNN Decoder - Convolutions have a kernel size of* 3×3*, maxpool has a kernel size of* 2×2*. Parameters not mentioned are default parameters in Pytorch library (in torch.nn).*

two fully connected layers as shown in Equation 5.13. If only historical trajectories are used, this context vector $C_t^{M,k}$ is replaced by $C_t^{*,k}$.

$$q_t^k = FC(LSTM_{c^k}(h_{t-1}^k, q_{t-1}^k, C_t^{M/*,k})) \tag{5.13}$$

The decoder used for each of the traffic-agent class c^k of interest (i.e., pedestrian/car/cyclist) is different. Therefore, based on c^k, the respective LSTM model is chosen to decode. The resulting output q_t^k is made of a sequence of points in a Cartesian grid:

$$q_t^k = [x_{Obs+1}, y_{Obs+1}, ..., x_{Obs+N}, y_{Obs+N}] \tag{5.14}$$

where N is the prediction horizon. This could range between 3 and 10 frames as per the experiments discussed in Section 5.4. The number of observed frames and the prediction horizon N are equal as per the system architecture design. For example, when 3 frames are observed, the prediction horizon will consist of 3 future frames.

The LSTM decoder also has a hidden state size of 300 similar to the size of the LSTM encoder. This size has been chosen by experimentation and following the recommendations made by Fernando et al. (2018b). This hidden state is mapped to a vector of size $1 \times (2 \times N)$ through two FC layers. Therefore, if the prediction horizon is equal to 10 frames, the output will be 1×20 which includes the value of the x,y coordinates of the predictions of the 10 frames.

5.3.4 Training

The architecture described in the previous section is trained using the NuScenes dataset. We extracted 3200 different trajectories with at least 30 time frames for the experiments which represent 15 seconds. These are of sufficient length for our experiments, they were selected from 850 scenes recorded for periods of 20s in the NuScenes data setup. The selected trajectories were split into the following sets: 2100 for training, 550 for validation and 550 for testing. For single-agent trajectory prediction, these sequences are considered separately, whilst for the multi-agent trajectory predictions, the information of all interacting agents are taken into account. Each trajectory sequence is linked with the relevant interacting agent information.

The training methods adopted by Bahdanau *et. al.* in their Neural Machine Translation work (Bahdanau et al., 2014) was used because they have a similar sequence prediction problem. For each time-step, an object-pool is created which is associated with a trajectory prediction network. The resulting model predicts short-term and long-term trajectories for each traffic-agent in the object-pool. This is based on the historic trajectory of the observed agent and the trajectories of other neighbouring agents. Therefore, each traffic-agent that resides in the pool is associated with a predicted short-term and long-term trajectory. The approach results in a short-term prediction for 3 frames (1.5 seconds) and a long-term for 10 frames (5 seconds). Studies show that occlusions last around 3-10 frames (Sadeghian et al., 2017). Having a long-term prediction enables the architecture to take into account the noise in the raw outputs from the detection network such as occlusions, false alarms, inaccurate bounding boxes, and missing detections. Hence, the choice of short-term and the long-term horizons for the prediction, they will help to compensate for occlusions by increasing the number of predicted trajectories that can be used for training.

The CNN encoder applied on the map-masks was tested with and without pre-training also using the NuScenes dataset. Experiments have shown us that the method without pre-training is faster, leading to the same final performance. The resulting model is attached to the LSTM encoder-decoder and then trained end-to-end.

The Adam optimiser was used as it is the most advantageous for this application (Kingma and Ba, 2017), with a learning rate of $1 \times e^{-4}$ for 100 epochs and fine-tuned with a learning rate of $1 \times e{-}5$ for 20 epochs.

5.4 Results

To demonstrate the feasibility and performance of the previously developed system, the trajectory prediction network was implemented and tested using part of the NuScenes dataset. The performance is compared with other existing methods. For evaluation purposes, the prediction accuracy is measured based on two metrics as described in Section 2.3: Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE provides the average error between the ground truth and predicted trajectories. This is measured in the map space (meters) with respect to each time step t (seconds) within the prediction horizon. The FDE provides the error between the final positions in the predicted trajectory and the ground truth.

To illustrate the advantage of using map information through the exploitation of the drivable/non-drivable areas definition in both single and multi-agent scenario, two machine learning models are compared. This can be summarised as follows:

- **Single Agent with LSTM Model and Map-Mask input:** Map-mask is taken as input to predict the trajectories of single agents, not considering interactions.

- **Multi-Agent interactions and Model with LSTM and Map-Mask input:** Multi-agent interactions are taken into account in the LSTM encoder-decoder architecture and map-masks are used to improve the predictions of traffic-agents involved in such interactions.

The developed models are compared, first with the model-driven approaches represented by:

- **Constant Velocity Model:** Trajectory is predicted by assuming constant velocity to that of the observed trajectory (Schöller et al., 2019).
- **Constant Curvature Model:** Trajectory is predicted by assuming constant curvature to that of the observed trajectory (Horst and Barbera, 2006).

and then with data-driven models, described below:

- **Single Agent LSTM Model for all traffic agents:** Separate LSTM models are used to predict the motion of single traffic agents. Interactions are not taken into account in this model.
- **Multi-Agent interactions and Model with LSTM only:** Interactions are taken into consideration and only the LSTM encoder-decoder architecture is used to predict trajectories. This is similar to the method in Fernando et al. (2018b).

Method	ADE(Metres)	FDE(Metres)
1. Constant Velocity Model	7.65	8.2
2. Constant Curvature Model	5.47	6.78
3. Single Agent LSTM only	5.31	6.2
4. Single Agent LSTM + Map-Input	**1.67**	**1.95**
5. Multi-Agent LSTM only	4.34	5.21
6. Multi-Agent LSTM + Map-Input	**1.45**	**2.2**

Table 5.3: *Comparison of Prediction Accuracy for different models*

The comparison of the ADE/FDE values obtained for the model-based methods, the LSTM baselines and our models with map-input is presented in Table 5.3. It shows that the machine learning models have higher performance than the classical methods. However, it can be observed that the LSTM approach predicting each single agent individually without map information presents performances close to the constant curvature model. The interest of the deep-learning approaches is only apparent when taking the agent interactions or the map context into account.

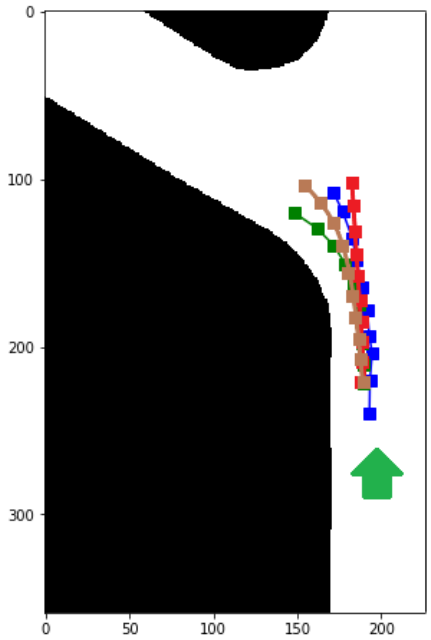


Figure 5.6: Comparison of trajectory prediction with classical methods: *Green is ground truth, red is constant velocity, brown is constant curvature and blue is map-mask based prediction. The green arrows in the images show the direction of motion of the ego-vehicle*

It is observed from Table 5.3 that in both single-agent and multi-agent predictions, the addition of map-mask patches help the system make better predictions than the models without the map-input. For single agent predictions, the ADE and FDE are reduced by 68.5%, when the map input is introduced. While, for multi-agent predictions, these are reduced by 66.6% and 57.8% respectively. It can be observed that there is a noticeable performance jump when contextual information is fed into the trajectory prediction system.

The performance of the model in different scenarios is illustrated in the following figures. Figure 5.6 shows the comparison of our model with model-driven approaches like the constant velocity model (red colour) and the constant curvature model (brown colour) and ground truth (green colour). Although the constant curvature model (brown colour) shows a better performance in the particular situation depicted in this figure, the overall performance of our proposed model (blue colour) exceeds that of the constant curvature model.

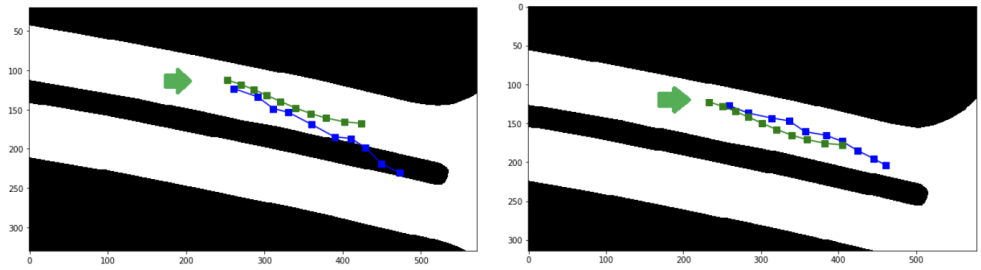


Figure 5.7: Straight path trajectory predictions: *Behaviour of Trajectory Prediction Models without Map-mask (Left) and with Map-mask (Right) input. The green arrows in the images show the direction of motion of the ego-vehicle, ground truth (in green) and predicted trajectory (in blue).*

Figure 5.7 shows an example trajectory of a car in a straight line. The prediction shows a trajectory in the non-drivable area when only an LSTM based model is used. The predicted trajectories are shown in blue and the ground truth in green. Once map-masks are included as inputs, the results show that the prediction (blue trajectory in the right image) is limited to the drivable area. In this example, we can clearly observe that contextual information aids in making better predictions.

Figure 5.8 shows that the fitting of the trajectory prediction to road curves when using map-masks is better predicted. When map-masks are not used in the system, the predicted trajectory diverges from the ground truth. By contrast, when map-masks are included, the prediction accuracy is better, following the curvature of the trajectory very closely. The divergence of the FDE for this scenario is attributed to weaker long term predictions. Figure 5.9 shows another example of the LSTM-only model predicting trajectories in non-drivable areas. It can be seen that adding map-mask solves this error and makes a better prediction compared to the LSTM baseline.

Figure 5.10 shows a scene with two interacting cars and a pedestrian. The observed car trajectories are shown in black and the pedestrian trajectories are shown in red, the ground truth trajectories for both agents are plotted in green. The predictions on the left are by a model without knowledge about interaction or maps while the one on the right is our interaction-aware model with map-

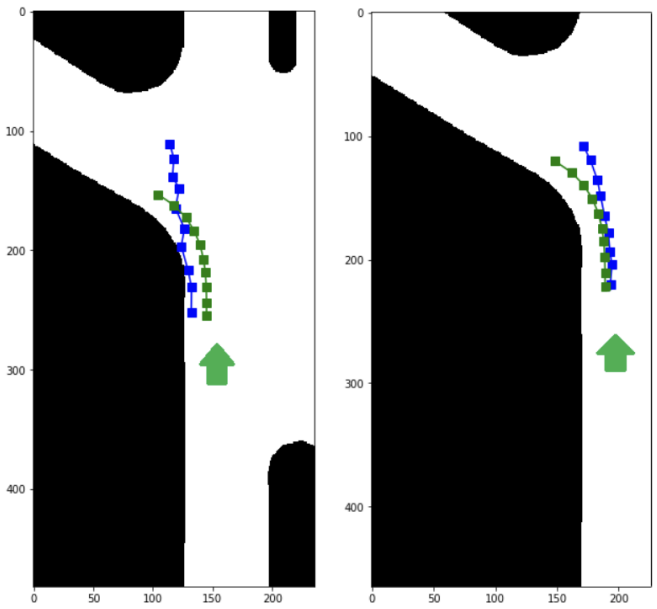


Figure 5.8: Trajectory prediction at turnings: *Behaviour of Trajectory Prediction Models without (Left) and with Map-mask (Right) input. The green arrows in the images show the direction of motion of the ego-vehicle. The green trajectories are the ground truth and the blue one is the predicted trajectory*

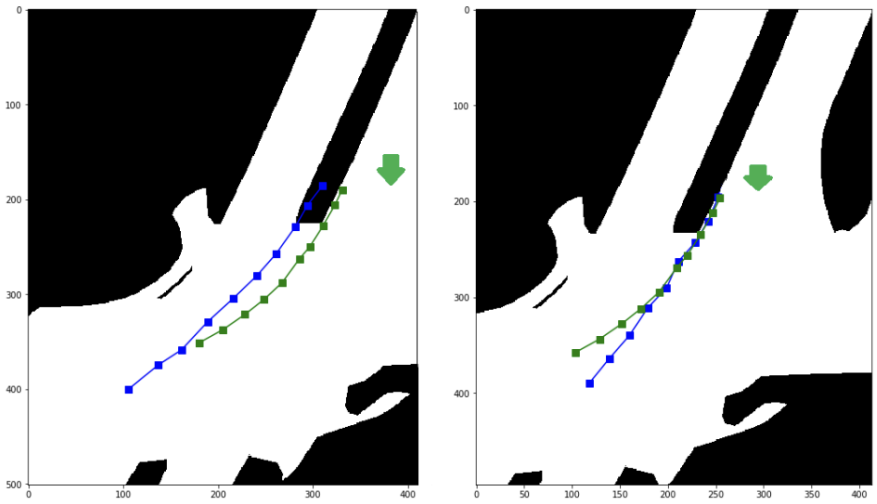


Figure 5.9: Better behaviour in non-drivable areas: *Behaviour of Trajectory Prediction Models without (Left) and with Map-mask (Right) input. The green arrows in the images show the direction of motion of the ego-vehicle. The green trajectories are the ground truth and the blue one is the predicted trajectory*

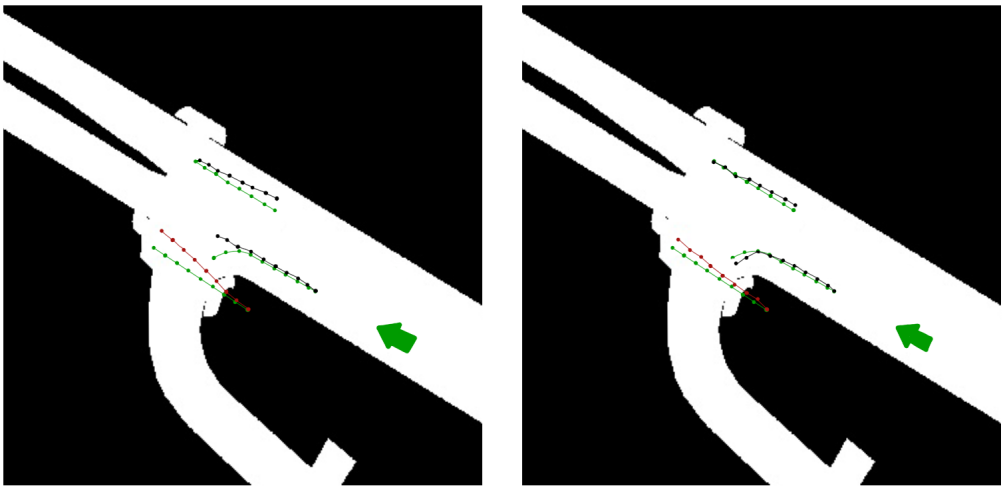


Figure 5.10: Multi-agent Trajectory prediction: *Predictions from a model without interaction information and map-input (Left) and one with interaction information and map-input (Right). The green arrow in the image shows the direction of motion of the ego-vehicle. The green trajectories are the ground truth and the predicted trajectories are plotted in red for pedestrians and black for cars.*

input. It can be observed that the predictions are more accurate in the proposed model. The curvature of the car trajectory is better predicted with the map-input. Pedestrian trajectory is also better estimated by the proposed model. This illustrates its capability to predict the trajectories of multiple traffic-agents with better accuracy than vanilla models.

5.5 Conclusion

In this chapter, a method was successfully proposed to predict the trajectory of interacting traffic-agents applicable to vehicle navigation. It uses the knowledge of the road network structure in the form of binary masks in conjunction with 3D LiDAR points to provide information about the driving scenario. A comparison of single-agent and multi-agent trajectory prediction models is analysed and the overall improvement of the prediction when map-masks are given as input is validated.

These results serve as a preliminary step in the direction of using environmental context information for improving prediction accuracy. Our approach

addresses the problem by using binary map-masks to facilitate the understanding of the scene environment but more semantic information could be used. For example, it would be interesting to exploit the Map Extension tool of the NuScenes dataset and use information such as crosswalks, sidewalks, traffic lights, stop lines and lanes that can be obtained from their network of mapped areas in order to improve the quality of trajectory prediction.

Most autonomous vehicles benefit from multi-sensor systems, where perception is enhanced by the use of multi-sensor fusion techniques. In the next chapter, 3D point-clouds from LiDAR is enriched by data from the images from on-board video cameras. These images include important situational information. For example, pedestrian awareness about the presence of a neighbouring vehicle (i.e., our ego-vehicle). The exploitation of this additional context cue to predict the pedestrian trajectories next to pedestrian crossings is developed in the next chapter to demonstrate that the perceived contextual information can enhance the predictions.

Chapter 6

Pedestrian Trajectory Prediction using Multimodal Context

Contents

6.1 Introduction

In the previous chapter the traffic-agents of interest namely - cars, pedestrians and bicycles are treated in a similar manner. A specific model for each was learnt, using their previous trajectory. It included the drivable areas available for each, extracted from information stored in HD maps (i.e, map-masks). However, the pedestrian motion is unpredictable compared to cars and bicycles which have tighter holonomic constraints. Additional information on their intentions is needed to make better predictions.

Interacting with pedestrians is an integral part of autonomous vehicle navigation for these to be accepted by society. In an urban scenario, an autonomous car will come across several pedestrians. It has to make decisions on whether to slow down, stop, keep moving or make an evasive manoeuvre. This implies observing the scenario, identifying the relevant factors that affect the motion of pedestrians and predict their future motion. Pedestrians might not be aware of their surroundings, could be distracted or in the best case scenario, interact with gestures or glances towards the vehicles. This is a very difficult task as it involves social interaction. The contribution of this work is to provide situational awareness so as to resolve this interaction task. Pedestrian interaction implies several challenges to a driver and even more to an autonomous driving system as it requires high levels of abstraction (beyond the classification only systems). Recognising different behavioural cases and making the decision whether to cede, stop or slow down is a problem that has to be solved.

In Chapters 4 and 5, trajectory prediction was achieved only by using a LiDAR sensor and masks issued from HD-maps. However, some limitations were found, there were some difficulties on the classification and prediction of trajectories of the traffic-agents. The accuracy of predicting pedestrian trajectories is much lower compared to other classes of traffic-agents. Hence, in this chapter the focus on the pedestrian class. For this purpose, we introduce a second sensor in the form of a video camera to exploit the detailed visual context that it provides. Hence, a multi-sensor perception system is proposed. A camera exploits the presence of visual cues in the scene, for example, determines whether the observed pedestrian has been looking in the direction of the autonomous vehicle and

acknowledges its presence. In this research, the focus is on using enriched information, like for example, pedestrian attention inferred from other systems to make predictions in more complex scenarios, as used in road intersection crossings. It is assumed that the detection of intention and attention has been solved (Lefévre et al., 2011; Schneemann and Heinemann, 2016).

The review paper (Rudenko et al., 2020) presents an extensive survey of pedestrian trajectory prediction. Among all the reviewed approaches, there are several examples of research on predicting pedestrian behaviours at intersection crossings using vision as input. For example, Kooij et al. (2014) propose a Dynamic Bayesian Network for pedestrian path prediction that incorporates environment parameters such as pedestrian situational awareness and head orientation with a Switching Linear Dynamical System to predict changes in the dynamics of pedestrians. These motion models, to be efficient, require accurate and precise segmentation and tracking of pedestrians. Such assumption can be challenging due to the difficulty of extracting reliable image features for segmentation and tracking. Hasan et al. (2016) treat the prediction of adverse pedestrian actions as an anomaly detection problem. They built a fully convolutional auto-encoder to learn the local features and use the reconstruction error to analyse the behaviour regularities, followed by a classifier to detect anomalies. Rasouli et al. (2017) extract context features from input frames using an AlexNet network (Krizhevsky et al., 2012) and train a linear SVM model to predict future crossing action of pedestrians. This is applied on the JAAD dataset (Rasouli et al., 2017). These approaches are limited in the fact that they focus only on spatial appearances in small temporal windows, ignoring the temporal coherence in long-term motions. To solve this issue, Gujjar and Vaughan (2019) perform the crossing actions classification by feeding the predicted frames of a frame prediction network to a 3D convolution based network (Tran et al., 2014). This takes into account the temporal dynamics in addition to the spatial appearances. Using different techniques, Keller and Gavrila (2014) propose two novel approaches, based on Gaussian process dynamical models and probabilistic hierarchical trajectory matching. They use augmented features derived from dense optical flow for pedestrian path prediction and action classification, at short sub-second time intervals as shown in Figure 6.1.

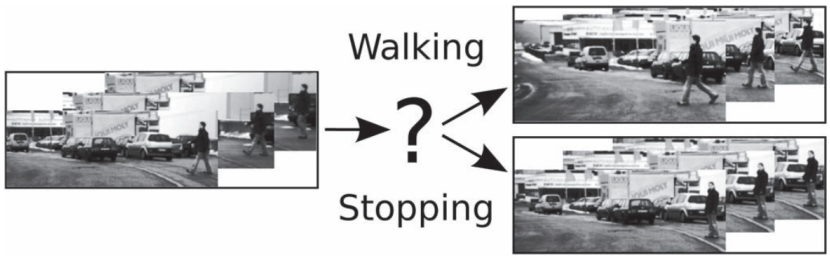


Figure 6.1: *Pedestrian trajectory and behaviour prediction while crossing a road. Image from (Keller and Gavrila, 2014)*

Kotseruba et al. (2016) present a novel dataset for a critical aspect of autonomous driving, the joint attention that must occur between drivers and of pedestrians, cyclists or other drivers. They also show how visual complexity of the behaviours and scene understanding is affected by various factors such as different weather conditions, geographical locations, traffic and demographics of the people involved. Malla et al. (2020) introduce TITAN (Trajectory Inference using Targeted Action priors Network), a new model that incorporates prior positions, actions, and context to forecast future trajectory of agents and future ego-motion.

This review shows that using contextual information from machine vision, it is possible to estimate pedestrian intention at intersection crossings. This information is used as input to our trajectory prediction network. In this chapter, the focus is on the motion of pedestrians at an intersection crossing. This data is obtained from the NuScenes dataset. To predict the future pedestrian positions within the field of view of the ego-motion vehicle, map-context, historical trajectory of the observed pedestrians and a confidence parameter on whether the pedestrian has seen the autonomous agent are used. The historical trajectory is obtained from LiDAR data and the vision data provides the confidence parameter.

Section 6.2 formulates the trajectory prediction problem that is to be solved. Section 6.3 describes the dataset and the data generated to train the machine learning model. The approach proposed to solve the prediction problem is developed in Section 6.4. The results are discussed in Section 6.5 to conclude the chapter.

6.2 Problem Formulation

6.2.1 Overview

The problem is formulated to find the manner in which the future trajectory of the observed pedestrians can be estimated. For this purpose, using contextual information, the model must predict what the trajectory of the pedestrian will be as it intends to cross a road. Formally, the task is to observe the features of pedestrians within the interval $[T_{obs} - w : T_{obs}]$ and to predict their positions in the interval $[T_{obs+1} : T_{pred}]$. w is the observation window for each detected pedestrian.

The process followed for the implementation of this system is shown in Figure 6.2. The inputs to the prediction module are 3D LiDAR point-cloud data, binary map-patches around each observed pedestrian and images. From the 3D LiDAR point-cloud data, the historical trajectory of each traffic-agent is observed with the map-patches providing information on the drivable and non-drivable areas and road crossings in the work-space. From images, the pedestrian attention to the oncoming vehicle at each time step is obtained. These inputs are embedded into a mathematical model so they can be fed to a machine learning model. The encoded sequence is used as input to train the LSTM Encoder which produces a feature vector. From this feature vector the trajectories are predicted using an LSTM decoder.

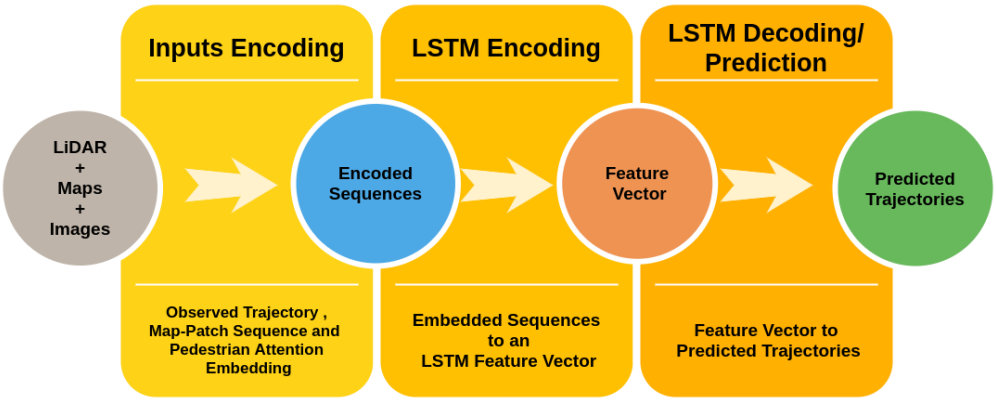


Figure 6.2: *The process involved in Pedestrian Trajectory Prediction*

6.2.2 Pedestrian Feature Set

A sequence of inputs that include LiDAR point-clouds and camera images together with map-masks from HD maps defines the feature set for each pedestrian k at time t. This can be represented as

$$f_t^k = (p_t^k, M_t^k, A_t^k) \tag{6.1}$$

where

$$p_t^k = [x_t^k, y_t^k] \tag{6.2}$$

Here, $[x_t^k, y_t^k]$ represents the position coordinates of the detected pedestrian in the ego-motion vehicle reference frame, obtained from the detections of LiDAR point-cloud. M_t^k represents the map mask, defined by a 128×128 pixels binary map patch, which is the area surrounding each observed pedestrian. A_t^k is a scalar confidence parameter representing the attention of the pedestrian k to the ego-vehicle. This is assumed to be obtained from vision data, where A_t^k is equal to 1 when the pedestrian is aware of the ego-vehicle and equal to 0 if it is unaware.

The task is to observe the features of all traffic-agents (pedestrians in this case) within the interval $[T_{obs} - w : T_{obs}]$ and to predict their positions in the interval $[T_{obs+1} : T_{pred}]$. The later will reflect if the pedestrian will cross the road or not, where w is the sliding observation window, defined in Chapters 4 and 5. So the problem resides in determining whether or not the pedestrian will cross the road. Our approach relies on the use of an LSTM network that should incorporate the contextual information.

In the next section, we describe the dataset used to train this LSTM Network.

6.3 Dataset

The NuScenes dataset is used in this chapter also [c.f. Section 3.3]. From this dataset, sequential point-cloud data, images and map-patches around the ego-motion vehicle are extracted. LiDAR data and map information is used as it is

from the NuScenes dataset. The pedestrian attention information is generated manually. To describe this, we split the dataset into two - Data from NuScenes and Generated Data. This is explained as follows:

6.3.1 Data from NuScenes

LiDAR data Observed Trajectory: The point-cloud data from the LiDAR is used to generate the observed pedestrian trajectory. This is obtained by using the Point Pillars approach applied previously in Chapter 5. The result will be a vector of (x,y) - coordinates representing the position of the observed pedestrians. This is illustrated in Figure 6.3. Out of 393 scenes identified from the NuScenes involving pedestrians, 198 separate trajectories involving a pedestrian crossing the road in front of the ego-vehicle are extracted. These involve scenarios where the pedestrian crosses the road with or without pedestrian-crossings and also scenarios where they stop and do not cross.

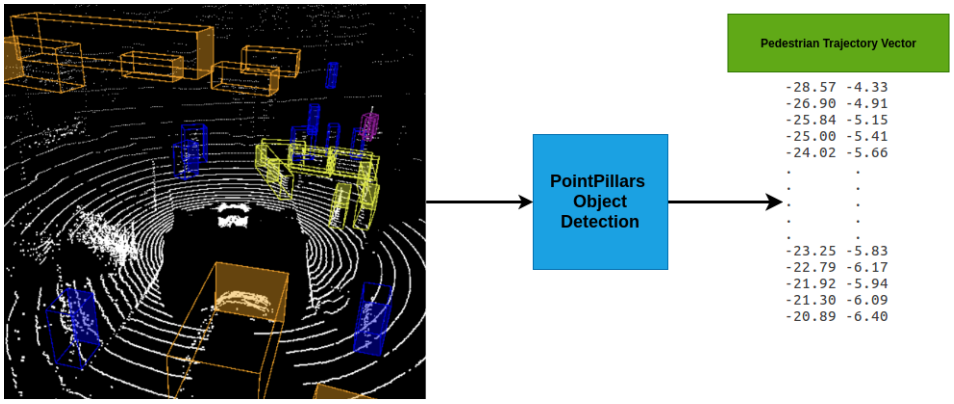


Figure 6.3: *Observed Pedestrian Trajectory generation applying the PointPillars Algorithm (Lang et al., 2018)*

Map Information Input (Context): Whilst previously, we only used binary information on drivable and non-drivable areas for this map-input, an addition description is incorporated regarding the presence of pedestrian crossings. The pedestrian crossings information is incorporated because pedestrians are likely to cross the road in those situations [1]. The pedestrian crossings are added to the

[1] Accident studies have shown that, despite different assumptions, there are a significant number of pedestrian accidents occurring when they cross the road at the pedestrian crossings. [Accidentology Laboratory Renault, 2021]

black areas in the map. Figure 6.4 shows this representation. An empirical study showed that the size of the map-patch - 128x128 pixels - is practical as in Chapter 5.

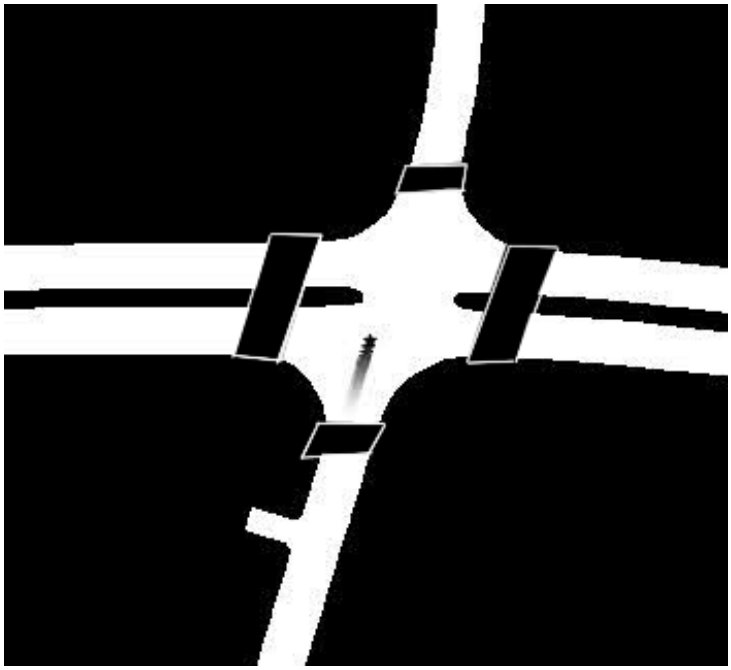


Figure 6.4: *Map masks including pedestrian crossings marked with white rectangles*

6.3.2 Generated Data: Pedestrian Attention from Images

The third input used for the pedestrian crossing prediction network are obtained from images, i.e, we have manually annotated a score that represents the level of attention of the pedestrian crossing to the presence of the ego-vehicle. This is shown in Figure 6.5. The annotation has been done by a single annotator. This could have led to inaccurate annotations and should be taken into account while analysing the results. This can be replaced by a pedestrian attention estimation network in the future. This is represented by a scalar labelled as the Pedestrian Attention A_t^k. For each time instance, the pedestrian attention is 1 if the pedestrian is looking at the incoming ego-vehicle and 0 if it has never looked at it. This value is approximately decided based on the angle of orientation of the pedestrian gaze. The pedestrian attention for all the time-instances observed are joined together to form the Pedestrian Attention Vector $A_{T_{obs}}^k$. This is shown in Figure 6.5.

Figure 6.5: *Pedestrian images used as visual input to annotate the Pedestrian Attention Vector*

Therefore, the training dataset will consist of Observed Pedestrian Trajectories, the corresponding map-patches and the pedestrian attention vectors.

6.4 Methodology

The architecture developed in this chapter is an extension of the works done in Chapter 5. The system exploits three input streams - LiDAR point-cloud data, Map-masks and images. From the 3D LiDAR point-cloud data, the historical trajectories of observed traffic-agents are extracted using the VoxelNet Object Detection Network. Using this trajectory information, the binary map-patches around each observed traffic agents are extracted for each time-step (Scene Mask Patches Retrieval). These map patch sequences are then encoded using a CNN Encoder. From images, the pedestrian attention is extracted for each time step. This is done through manual annotation. The observed trajectories generated by

the VoxelNet are then encoded together with the encoded map patches and the pedestrian attention vector using the LSTM Trajectory encoder. The resulting feature map is then fed into an LSTM Trajectory Decoder to obtain the Predicted Trajectories. This system architecture with these inputs is illustrated in Figure 6.6. The details of the different components of this architecture are presented in the next section.

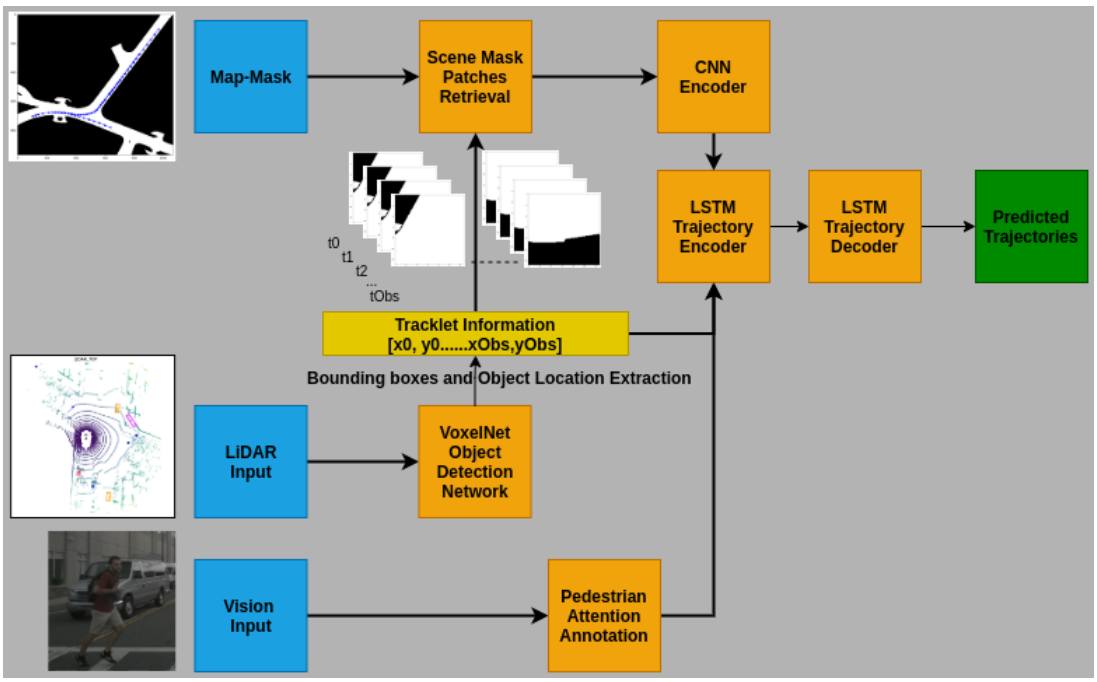


Figure 6.6: *System Architecture: The Prediction Network has three inputs: sequential historical trajectories of all traffic-agents, the associated map-mask patches around each agent for the observed time instances and the confidence of pedestrian attention towards the autonomous agent.*

6.4.1 Historical Trajectory Embedding

Firstly, LiDAR point-cloud data is passed through the VoxelNet Object detection network to extract the location of each pedestrian. The pedestrians are observed for T_{Obs} and the sequence of positions extracted are passed to the next steps in the prediction pipeline. These sequences are used to extract map-patches and also passed on to the LSTM encoder to generate the historical trajectory embedding.

Following the same steps in Chapter 5, let the historical trajectory of a pedestrian k, from time-step 1 to time-step T_{obs} be given by,

$$p^k = [p_1^k, ..., p_{T_{obs}}^k], \tag{6.3}$$

where,

$$p_i^k = [x_i^k, y_i^k] \tag{6.4}$$

for each time-step i. These historical trajectories are passed through the LSTM encoder of each respective pedestrian to generate its historical embedding as follows,

$$h_t^k = LSTM(p_t^k, h_{t-1}^k), \tag{6.5}$$

generating a sequence of historical embedding. With these historical embedding, a merged context vector, $C_t^{*,k}$ is computed by concatenating the historical context vector $C_t^{H,k}$ and spatial context vector $C_t^{S,k}$ (as defined in Chapter 5) and applying a non linear function:

$$C_t^{*,k} = tanh([C_t^{H,k}; C_t^{S,k}]) \tag{6.6}$$

6.4.2 Context from Map input

Secondly, Map-masks associated with the driving location is the input that serves for extracting map-patch sequences. These binary image sequences provide static-context information to the prediction module, i.e., non-drivable areas including pedestrian crossings on the map. Square patches of 128×128 pixels are extracted, centred around each location of the detected pedestrian from the LiDAR input. These map-patch sequences are encoded using a CNN encoder (elaborated in Chapter 5). The resulting feature vector is also passed to the LSTM encoder. To elaborate, each of the 128×128 pixels binary map patch M_t^k is passed through a Convolutional Encoder producing a latent vector $C_t^{C,k}$ which holds the information about the non-drivable areas and pedestrian crossings associated with the respective pedestrian k at time-instance t. These sequential latent vectors for each observed time instant in the observation window are concatenated to form the Map Context vector $C_{T_{Obs}}^{C,k}$ at time T_{Obs}.

6.4.3 Pedestrian Attention Embedding

The third input that feeds the prediction module are images. Images are used to extract Pedestrian Attention Vector as described in Section 6.3.

Let the historical attention of a traffic-agent k, from time-step T_{obs-w} to time-step T_{obs} be given by,

$$A^k = [A^k_{T_{Obs-w}}, ..., A^k_{T_{obs}}], \tag{6.7}$$

where, $A^k_i = [0, 1]$ for each time-step i.

An embedding of the attention vectors is generated by passing it through an LSTM as follows:

$$Ah^k_t = LSTM(A^k_t, Ah^k_{t-1}), \tag{6.8}$$

where Ah^k_t are the hidden states for pedestrian k at time instance t. Now, the Pedestrian Attention Embedding $C^{A,k}_t$ is computed as a weighted sum of hidden states from $t = [1 : T_{obs}]$:

$$C^{A,k}_t = \sum_{j=1}^{T_{obs}} \alpha_{tj} Ah^k_j \tag{6.9}$$

and the weight α_{tj} as shown in Bahdanau et al. (2014) can be computed by:

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{l=1}^{T} exp(e_{tl})} \tag{6.10}$$

$$e_{tj} = a(Ah^k_{t-1}, Ah^k_j) \tag{6.11}$$

where the function a is a feed forward neural network which is trained jointly with the whole network.

6.4.4 Pedestrian Trajectory Prediction

We have three vector sequences which provide static-context (maps), historical trajectory information (LiDAR) and Attention Context (Pedestrian Attention vector extracted from images). The Map Context vector $C^{C,k}_t$ is therefore concatenated with the merged context vector $C^{*,k}_t$ generated from the historical trajectory and

the Attention Embedding $C_t^{A,k}$ generated from the Pedestrian Attention sequences to obtain the final Pedestrian context vector $C_t^{P,k}$:

$$C_t^{P,k} = [C_t^{*,k}; C_t^{C,k}; C_t^{A,k}] \tag{6.12}$$

Now, the final predicted trajectory q_t^k is obtained by passing the final Pedestrian context vector $C_t^{P,k}$ through the LSTM decoder and two fully connected layers:

$$q_t^k = FC(LSTM(h_{t-1}^k, q_{t-1}^k, C_t^{P,k})) \tag{6.13}$$

The output q_t^k is composed of points in a Cartesian grid:

$$q_t^k = [x_{Obs+1}, y_{Obs+1}, ..., x_{Obs+N}, y_{Obs+N}] \tag{6.14}$$

where N is the prediction horizon which could range from 3-10 frames in our experiments.

6.5 Results

The final model described in the previous section is implemented to predict the trajectories of pedestrians at road crossings. Based on the prediction we also check if the model is able to differentiate the cases where the pedestrian crosses the road or not. To demonstrate the feasibility and performance of this system, the network was implemented and tested using part of the NuScenes dataset.

The prediction accuracy is measured based on two criteria as described in Section 2.3 : Average Displacement Error (ADE) and Final Displacement Error (FDE) in the map space/pixel space with respect to each time step t within the prediction horizon. ADE provides the average error between the ground truth and predicted trajectories. The FDE provides the error between the final positions in the predicted trajectory and the ground truth.

To illustrate the interest of using map information and pedestrian attention from images, three variants of our approach are implemented:

- **LSTM-only Baseline:** This method uses observed trajectory obtained from LiDAR data and a basic LSTM encoder-decoder architecture.
- **LiDAR + Maps (LM):** This method exploits LiDAR data and map-masks providing information about pedestrian crossings.
- **LiDAR + Maps + Attention (LMA):** This is the proposed approach combining information from all three sources - LiDAR, maps and images.

Method	ADE(Metres)	FDE(Metres)
1. Baseline - LSTM only	3.31	5.54
2. LiDAR + Map-mask (LM)	2.23	3.32
3. LiDAR + Maps + Attention (LMA)	**1.2**	**1.7**

Table 6.1: *Comparison of Prediction Accuracy for different models*

Table 6.1 clearly shows that the full model LMA, with all three inputs, outperforms the baseline and LM model. By using LMA, the prediction error ADE is reduced by 46.18% compared to LM and 63.75% when compared to the LSTM baseline. Furthermore, the FDE is reduced by 48.8% and 69.3% with respect to LM and the baseline respectively. However the performance of this model in specific situations is not apparent from this table alone. For this, we analyse its performance to predict whether the pedestrian crosses the road (C) or do not cross the road (NC) in predefined conditions. The scenarios that are analysed include - road crossing present (RC - Present), road crossing absent (RC - Absent), pedestrian has seen or is aware of the ego-vehicle (Seen) and situation where the pedestrian is not aware of the ego-vehicle (Not seen). For each model listed above, the C or NC prediction in these four cases are tabulated and presented in Figure 6.7. For each pedestrian trajectory, a threshold distance is calculated to decide whether he/she has crossed the road or not. Figure 6.7 shows the confusion matrices for each scenario. The colour of each cell represents the probability of predicting the label in the horizontal axis when the true label is the one in the vertical axis. Light colour signifies low probability while dark shades of green signifies higher probability. A model that makes perfect predictions will show a matrix with positions [0,0] and [1,1] in dark green, and positions [0,1] and [1,0] as

white. Table 6.2 shows the precision of the prediction in these scenarios. "RC-Present" or "RC-Absent" refers to the presence or absence of road crossings and "Seen" or "Not Seen" refers to the pedestrians' awareness of the ego-vehicle.

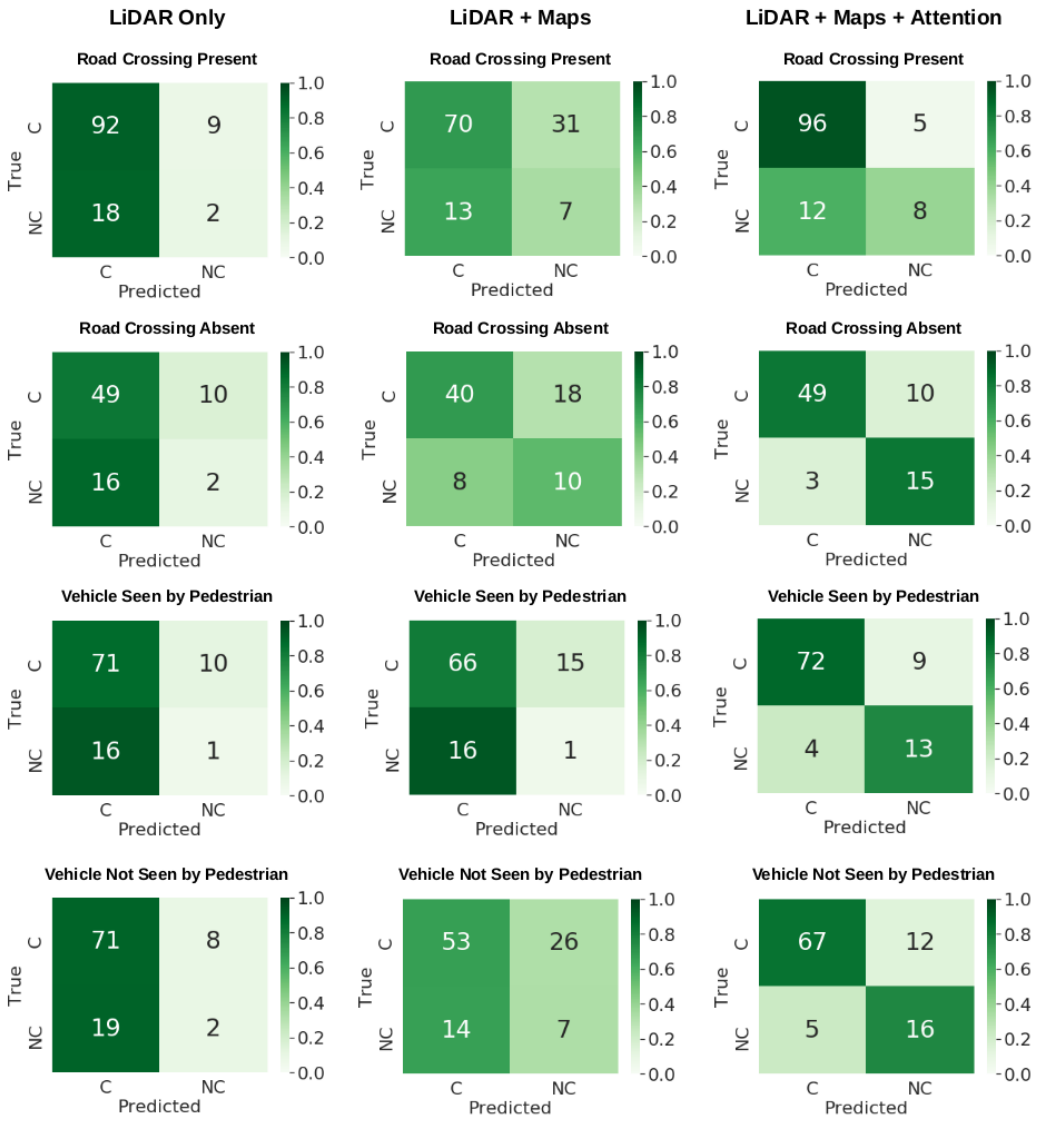


Figure 6.7: *Prediction of Crossing or Not Crossing of a pedestrian in different scenarios. The heatmap value shows the probability of predicting the label in the horizontal axis when the true label is the one in the vertical axis.*

Examining the performance of the LSTM only model, shown in the first column of Figure 6.7, it is observed that the model predicts the crossing action

	Precision for Crossing				Overall Precision
	RC – Present	RC – Absent	Seen	Not Seen	
LiDAR only	0.84	0.75	0.82	0.79	0.80
LiDAR + Maps	0.84	0.83	0.8	0.79	0.82
LiDAR + Maps + Attention	0.89	0.94	0.95	0.93	0.93

	Precision for Not Crossing				Overall Precision
	RC – Present	RC – Absent	Seen	Not Seen	
LiDAR only	0.19	0.17	0.1	0.2	0.17
LiDAR + Maps	0.18	0.34	0.06	0.21	0.20
LiDAR + Maps + Attention	0.62	0.6	0.59	0.57	0.60

Table 6.2: *Precision of pedestrian crossing prediction in different scenarios*

of pedestrians with a precision of 80%. But when it comes to predicting the non crossing of pedestrians, the model fails. It only has a precision of 10-20% in different scenarios. This shows that without contextual information, the LSTM-only model mostly predicts a sequence that fits a pedestrian crossing the road. It is unable identify cases where the pedestrian would stop and not cross.

LM model shows a 100% improvement in the crossing prediction compared to the LiDAR only model, when road crossings are absent. This can be noted from Table 6.2. However, the precision is still only 34%. This model however, shows similar performances to the LSTM only model while considering the attention cue - Seen/Not Seen. This is evident from the fact that the model only has information about past trajectories and maps. It is unable to differentiate cases where the pedestrian behave differently because of the perceived ego-vehicle.

Finally, if we consider the performance of the complete model with LiDAR, Maps and Attention (LMA), an overall improvement can be observed in all scenarios. For predicting the crossing of pedestrians, it shows a precision of around 90% in all contexts. Also, for predicting that the pedestrian stops and do not cross, the precision has hugely improved to 60% in the different scenarios. This is an improvement of 200% compared to LM and the baseline LSTM. The overall precision for predicting the crossing of pedestrians has also improved by

11.8% compared to the LM model.

6.6 Conclusion

The question of whether a pedestrian will cross the road or not, is critical in the context of autonomous vehicle navigation. This chapter addresses this problem from a trajectory prediction point of view. A lot of factors are involved in the decision that leads to a pedestrian crossing a road. Presence of road crossing, traffic-lights, awareness of oncoming vehicles, distraction etc., are a few examples of these factors. Without having a global idea of all such factors, it is hard to estimate the behaviour of pedestrians. In this chapter, the use of such context cues are studied to predict the trajectories of pedestrians at road crossings.

The models developed in this chapter show promising results in the context of pedestrian trajectory prediction. The final model which incorporates pedestrian awareness to oncoming traffic when compared to the baselines with only observed trajectory as input, and the model with LiDAR and Maps as input show superior performances in all situations. The precision of crossing prediction under four predefined scenarios are compared to test the model performances. These scenarios are - Road crossing present, Road crossing absent, Pedestrian has seen oncoming vehicle, Pedestrian has not seen oncoming vehicle. Our model shows a precision of 93% to predict cases where the pedestrian crosses and 60% to predict cases where the pedestrian do not cross. Even though this is not the best performance one can expect, the improvement over the other models to predict specific situations has to be noted.

As future work, adding more context cues and optimising the performance of the developed model has to be studied. Training the model on a bigger and better dataset will also aid in achieving superior results. Several assumptions have been made to make the development and testing of the proposed methods easier. Pedestrian attention is assumed to be obtained from an existing solution. The dataset also provides synchronised data. This is not the case in real-life. This assumption of having synchronised data will also have to be put to test and verified with real driving data. Future works would involve getting rid of these

suppositions and improving the prediction performance in all scenarios.

Chapter 7

Conclusion

Contents

The Association Nationale de la Recherche et de la Technologie (ANRT) under the Conventions Industrielles de Formation par la REcherche (CIFRE) Project between Groupe Renault and ENSTA Paris sponsored this thesis. The research objective was to develop multi-sensor data fusion perception algorithms applicable to autonomous vehicles. The initial high level goal was set to improve existing approaches and extend current algorithms to understand the physics of the perception problem . After this exploration phase, the trajectory prediction task was chosen to be the focus of this thesis. The rationale resides on the need of trajectory estimates, to facilitate the understanding of the behaviour of the observed traffic- agents, prior to decision making utilising the data acquired by the on-board exteroceptive sensors. The approach taken on this research is to gradually build a machine learning architecture that ultimately is capable to incorporate contextual information as part of inputs into the prediction process.

7.1 Summary of the Work

Chapter 2 undertakes a state-of-the-art review of different trajectory prediction methods and presents them in a structured manner. They are classified into: model-based, data-based and models that are aware of interaction and context cues. The methods are also classified according to the type of perception sensors namely, active and passive. The notions of multi-sensor fusion are also introduced as this approach is preferred to enhance results but also provide redundancy in safety critical applications. The evaluation metrics used in this thesis are also included. The chapter introduces the framework that is used as the basis for the approach proposed in this thesis.

Chapter 3 introduces the reader to various machine learning networks, their structures, guidelines to build them and their application areas. It enabled us to gain an understanding of the most suitable for our objectives, at different stages some of these networks were tested. Datasets play a crucial role in the development of machine learning algorithms. Without them, most of the myriad tasks solved with machine learning could not be accomplished with the accuracy levels attained in the past few years. The different datasets which could be used in our research were analysed and the most relevant selected.

Chapter 4 details the initial research direction in this thesis. It evolves around the idea that each class of traffic-agents has its own dynamic behaviour in a traffic scenario. Therefore, their motion trajectories could be used as a feature to distinguish them from one another. As a first familiarisation with the LSTM neural networks, we examined whether it is possible to use this idea to classify the observed trajectories of traffic-agents. Although, in the domain of autonomous vehicle navigation, such classification is of limited interest, because perception of the agents usually gives good cues about their class, it served us to use and understand the LSTM model in particular the effect of some the hyper-parameters used to define it. The results in this chapter pointed us to the direction of trajectory prediction as well as to understand the different challenges.

In Chapter 5, the main contribution of this thesis is presented. Experimental results showed that perception alone has its limitations when predicting

the future trajectories of the observed traffic agents. The hypothesis related to the introduction of context cues is formulated. Different studies on human understanding have also shown that drivers use multiple cues to understand what other traffic agents might do, reacting accordingly. There are inherent rules that decide the path taken by different classes of traffic-agents. In this chapter, we state that maps can provide contextual information. For this purpose, we use map-masks available in the NuScenes dataset. The challenge is on the manner that they can be introduced into a machine learning model to improve the prediction of the trajectories of the traffic-agents. These map-masks provide crucial information about the driveable and non-drivable areas within the vehicle workspace, thus determining indirectly where the traffic-agents can evolve. A comparison between single-agent and multi-agent trajectory prediction models is included. The results shown an overall improvement of the prediction when map-masks are given as input. It is possible to associate contextual cues into machine learning models.

Chapter 5 addresses the application of the models developed in Chapter 5 to the task of pedestrian crossing prediction. The approach is to seek an extension of our previous machine learning architecture so it can incorporate additional contextual cues. For this purpose, the attention of the pedestrian to the oncoming ego-vehicle is incorporated into the ML architecture. The assumption is made that this information is available from images acquired by the on-board sensors. This additional context cue plays a significant role as most humans interact with the oncoming vehicles with their gaze. In the proposed architecture, manually annotated information from camera images are included. The results provide sufficient insight to show that adding other context cues help to improve the prediction of pedestrian trajectories and crossing behaviours.

7.2 Summary of Findings

The main findings in this thesis are:

- **Interaction Aware systems are crucial in autonomous navigation:** To predict the motion of relevant traffic-agents in a traffic scenario, many elements are critical. The interaction between them is very significant. The

proposed approach investigated the trajectory prediction of the traffic-agents when we consider or ignore the awareness of other agents within the same workspace. Within the context of this thesis, by interaction, we mean the interaction of trajectories. Test results showed that when the system had information about other road-users, the predicted trajectories were better aligned with respect to the ground truth.

- **Maps provide important semantic information about the traffic-agents:** General rules constrain the motion of traffic-agents in the environment. There are spaces where cars can navigate and those where they cannot. On the contrary, pedestrians can be present in both drivable and non-drivable areas. The use of simple binary map-masks (drivable and non-driveable areas) in the prediction model helped intrinsically when they are added into the knowledge pool of the system. Results prove that the use of such map-masks indeed improve the prediction performance in specific scenarios. It was demonstrated that this information could be included also into ML architectures. The resulting model is able to predict cases where the pedestrian crosses because of the presence of a road-crossing.

- **Social Cues improve prediction tasks:** Driving is a social task. There is a lot of subtle interaction that happen between different traffic-agents. Cars beeping, pedestrian gaze, use of vehicle indicators etc. are some examples of such interaction. In the proposed method, we focus on one such cue, i.e., pedestrian attention to the oncoming vehicle. It can be inferred that such awareness of pedestrian intention provides substantial information to the prediction module in deciding whether the road-user will cross or not for example. This cue is particularly useful when people deviate from normal behaviour, for example - crossing the road in the absence of a road crossing. Having information about the attention of the pedestrian helps predict such situations correctly.

7.3 Perspectives

This thesis proposes an interaction-aware trajectory prediction system that depends on context cues. The results of this research is of relevant industrial interest

and they point towards the need for further work in this area. For safety critical functions, the classification of traffic agents might be insufficient to provide full situational awareness, by adding contextual information this classification can be extended. This section presents some general perspectives and discusses them in the context of the conclusions mentioned above.

- **Building a complete framework from ground up:** Though initial works undertaken in this thesis were to replicate results from other researchers, finding a tangible research direction took time. Work on machine perception applicable to vehicle navigation has been exploited as a research domain in recent years. The system architecture built in this thesis is an original work and was developed from ground up. Consequently, there is a good scope for improvement. Preliminary results are promising enough to continue the works in this direction by using more cues and optimising the developed machine learning models. Hyper-parameter tuning can produce better results. Further, contextual information could be linked in a spatial manner to provide a graphical relationship between them that bound likely errors.

- **The Dataset Limitation:** Datasets are the most important aspect of solving a problem using machine learning. Having a well-rounded dataset, that comprises of all or most of the relevant cases in the problem, makes a big difference in the accuracy in which the problem can be solved. This thesis also deals with such a constraint. Finding a dataset with the required data proved to be difficult. Even with the final choice of the dataset, a big part of it had to be manually generated through different techniques. This may cause constraints in replicating results. In the future, a dataset where all the scenarios are recorded from real-life examples can make the model consistent. For the exploration of machine learning models, synthetic models could be used that represent the situations that pose safety concerns when driving autonomously, generating out-of-the-norm situations for example, for model improvements and testing.

- **Localisation problem with maps:** The use of maps in our system also poses the question of localisation. The availability of maps has been proven to be useful to make accurate predictions. But in real life, these maps could

have errors because of road-works or outdated maps or even localisation errors. Our system does not have a method to account for such errors. It assumes a perfectly synchronised world-model. Having the luxury of the NuScenes dataset aids in this context. Hence, in the future, an extension that deals with such errors and variances in the map-masks has to be developed. Further, the use of semantic maps where the spatial relationships are described in a graphical model should be explored on the manner this can be exploited as part of the machine learning architecture.

- **Improving pedestrian intention recognition:** The proposed framework takes only the position, heading and attention of pedestrians into account during the inference phase to predict their positions. Prediction of pedestrian behaviour can be greatly improved by taking into consideration some of the other cues used by human drivers to decide whether a pedestrian will cross or not. This extension could incorporate cues such as shoulder and body poses, or current behaviour such as looking at a smartphone (Rasouli et al., 2017; Hoy et al., 2018). Further, the notion of hazard could be incorporated by including a clue on the likelihood of safety issues with respect to the situations in which pedestrian accidents occur from accidentology data.

- **Utilising more context cues:** We have studied the benefit of adding context cues to the prediction module. However, in real-life, context cues are more than pedestrian attention or drivable or non-drivable areas. They can be traffic-signs, vehicle speeds, more semantic information from HD-maps, pedestrian gestures, etc. The incorporation of these cues into a prediction system shall improve situation understanding and lead to better autonomous vehicle navigation (Oh and Peng, 2020).

- **Multi-Modal Prediction:** The work presented in this thesis addressed solutions with a single possible predicted trajectory for traffic-agents. Multi-modal predictions are important when it comes to situations where there are multiple paths or options that a traffic-agent can take. Works similar to Mercat et al. (2020) point to a direction that can be adopted as future developments to our model. They use a multi-head attention mechanism to achieve multi-modal trajectory predictions.

- **System Validation:** The validation of the type of systems as the one proposed is complex, it is difficult to build statistical evidence. Validation of the models developed in this thesis is a task that must be done in real-life driving scenario. Although the models have been tested and verified to work in specific scenarios, this is yet to be validated when it comes integrated onto an autonomous vehicle. More complex scenarios that pose themselves in real-life have more variables, thus the scenario exploration complexity increases the number of scenarios to be tested drastically. A first step would be to validate these in the predefined use-cases and then extend from there or to explore the use of simulation technologies.

  The validation problem remains a core challenge for the deployment of complex autonomous systems. This is not only a crucial step in the industry to test a system but also a research problem to find the correct methods. These are likely to be different given the nature of the tested systems.

List of Figures

List of Tables

Bibliography

[1] (2018). *Gated Recurrent Units.*

[2] (2018). *Robust Vision Challenge 2018.*

[3] (2019). *Self-Driving Made Real.*

[4] (2020). *Changing how the world moves.*

[5] (2020). *Driving cities forward.*

[6] (2020). *IBM Machine Learning.*

[7] (2020). *We're building the World's Most Experienced Driver.*

[8] 2020, A. E. (2020). Avenue europe 2020.

[9] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Li, F. F., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. pages 961–971.

[10] Altché, F. and de La Fortelle, A. (2018). An LSTM network for highway trajectory prediction. *CoRR*, abs/1801.07962.

[11] Althoff, M., Heß, D., and Gambert, F. (2013). Road occupancy prediction of traffic participants. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 99–105.

[12] Ammoun, S. and Nashashibi, F. (2009). Real time trajectory prediction for collision risk estimation between vehicles. In *ICCP 2009*, CLUJ NAPOCA, Romania.

[13] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

[14] Bartoli, F., Lisanti, G., Ballan, L., and Del Bimbo, A. (2018). Context-aware trajectory prediction. *2018 24th International Conference on Pattern Recognition (ICPR)*.

[15] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166.

[16] Brostow, G. J., Fauqueur, J., and Cipolla, R. (2008). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x):xx–xx.

[17] Brännström, M., Coelingh, E., and Sjöberg, J. (2010). Model-based threat assessment for avoiding arbitrary vehicle collisions. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):658–669.

[18] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2019). nuscenes: A multimodal dataset for autonomous driving. *CoRR*, abs/1903.11027.

[19] Casas, S., Luo, W., and Urtasun, R. (2018). Intentnet: Learning to predict intention from raw sensor data. In Billard, A., Dragan, A., Peters, J., and Morimoto, J., editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 947–956. PMLR.

[20] Chandra, R., Guan, T., Panuganti, S., Mittal, T., Bhattacharya, U., Bera, A., and Manocha, D. (2019). Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms.

[21] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps.

[22] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017a). Multi-view 3d object detection network for autonomous driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[23] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017b). Multi-view 3d object detection network for autonomous driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Chiu-Feng Lin, Ulsoy, A. G., and LeBlanc, D. J. (2000). Vehicle dynamics and external disturbance estimation for vehicle path prediction. *IEEE Transactions on Control Systems Technology*, 8(3):508–518.

[25] Colyar, J. and Halkias, J. (2007). Ngsim - us highway 101 dataset. In *Federal HighwayAdministration (FHWA) technical report, FHWA-HRT07-030*.

[26] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Coutaz, J., Crowley, J., Dobson, S., and Garlan, D. (2005). Context is key. *Commun. ACM*, 48:49–53.

[28] Danielsson, S., Petersson, L., and Eidehall, A. (2007). Monte carlo based threat assessment: Analysis and improvements. pages 233 – 238.

[29] Deng, J., Li, K., Do, M., Su, H., and Fei-Fei, L. (2009). Construction and Analysis of a Large Scale Image Ontology. Vision Sciences Society.

[30] Deo, N. and Trivedi, M. M. (2018). Convolutional social pooling for vehicle trajectory prediction. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[31] Dertat, A. (2017). Applied deep learning - part 4: Convolutional neural networks.

[32] Dewan, A., Caselitz, T., Tipaldi, G. D., and Burgard, W. (2016). Motion-based detection and tracking in 3d lidar scans. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4508–4513.

[33] Dimitrievski, M., Veelaert, P., and Philips, W. (2019). Behavioral pedestrian tracking using a camera and lidar sensors on a moving vehicle. *Sensors*, 19(2).

[34] Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2009). Pedestrian detection: A benchmark. In *CVPR*.

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

[36] Endsley, M. R. (2015). Situation awareness misconceptions and misunderstandings. *Journal of Cognitive Engineering and Decision Making*, 9(1):4–32.

[37] Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., and Posner, I. (2017). Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361.

[38] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

[39] Fernando, T., Denman, S., Sridharan, S., and Fookes, C. (2018a). Soft + hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *Neural Networks*, 108:466–478.

[40] Fernando, T., Denman, S., Sridharan, S., and Fookes, C. (2018b). Tracking by prediction: A deep generative model for mutli-person localisation and tracking. *CoRR*, abs/1803.03347.

[41] Firl, J., Stubing, H., Huss, S., and Stiller, C. (2012). Predictive maneuver evaluation for enhancement of car-to-x mobility data. pages 558–564.

[42] Flohr, F. and Gavrila, D. (2013). Pedcut: An iterative framework for pedestrian segmentation combining shape models and multiple data cues. pages 66.1–66.11.

[43] Fujiyoshi, H., Hirakawa, T., and Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4):244 – 252.

[44] Fung, M. L., Chen, M. Z. Q., and Chen, Y. H. (2017). Sensor fusion: A review of methods and applications. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 3853–3860.

[45] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

[46] Gindele, T., Brechtel, S., and Dillmann, R. (2013). Learning context sensitive behavior models from observations for predicting traffic situations. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1764–1771.

[47] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

[48] Graves, A., Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

[49] Guerrero-Higueras, A. M., Alvarez-Aparicio, C., Calvo Olivera, M. C., Rodriguez-Lera, F. J., Fernandez-Llamas, C., Rico, F. M., and Matellan, V. (2019). Tracking people in a mobile robot from 2d lidar scans using full convolutional neural networks for security in cluttered environments. *Frontiers in Neurorobotics*, 12:85.

[50] Guivant, J., Nebot, E., and Durrant-whyte, H. (2000). Simultaneous localization and map building using natural features in outdoor environments. *Intelligent Autonomous Systems*, 6.

[51] Gujjar, P. and Vaughan, R. (2019). Classifying pedestrian actions in advance using predicted video of urban driving scenes. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2097–2103.

[52] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[53] Habibi, G., Jaipuria, N., and How, J. P. (2018). Context-aware pedestrian motion prediction in urban intersections.

[54] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., and Davis, L. S. (2016). Learning temporal regularity in video sequences.

[55] Higgins, M. (2017). Udacity self-driving car driving data.

[56] Hillenbrand, J., Spieker, A. M., and Kroschel, K. (2006). A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):528–540.

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

[58] Horst, J. and Barbera, A. (2006). Trajectory generation for an on-road autonomous vehicle - art. no. 62302j. *Proc SPIE*, pages 82–.

[59] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. (2020). One thousand and one hours: Self-driving motion prediction dataset.

[60] Hoy, M., Tu, Z., Dang, K., and Dauwels, J. (2018). Learning to predict pedestrian intention via variational tracking networks. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3132–3137.

[61] Hu, Y., Zhan, W., and Tomizuka, M. (2019). Multi-modal probabilistic prediction of interactive behavior via an interpretable model. *CoRR*, abs/1903.09381.

[62] Jawed, S., Boumaiza, E., Grabocka, J., and Schmidt-Thieme, L. (2019). Data-driven vehicle trajectory forecasting.

[63] Jihua Huang and Han-Shue Tan (2006). Vehicle future trajectory prediction with a dgps/ins-based positioning system. In *2006 American Control Conference*, pages 6 pp.–.

[64] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*.

[65] Karpathy, A. and Li, F.-F. (2015). Cs231n: Convolutional neural networks for visual recognition.

[66] Keller, C. and Gavrila, D. (2014). Will the pedestrian cross? a study on pedestrian path prediction. *Intelligent Transportation Systems, IEEE Transactions on*, 15:494–506.

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

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

[69] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[70] Klingelschmitt, S. and Eggert, J. (2015). Using context information and probabilistic classification for making extended long-term trajectory predictions. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 705–711.

[71] Kocic, J., Jovičić, N., and Drndarevic, V. (2018). Sensors and sensor fusion in autonomous vehicles.

[72] Kooij, J. F. P., Schneider, N., Flohr, F., and Gavrila, D. M. (2014). Context-based pedestrian path prediction. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 618–633, Cham. Springer International Publishing.

[73] Kotseruba, I., Rasouli, A., and Tsotsos, J. K. (2016). Joint attention in autonomous driving (JAAD). *CoRR*, abs/1609.04741.

[74] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

[75] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc.

[76] Kumaran, S. K., Dogra, D. P., Roy, P. P., and Mitra, A. (2018). Video trajectory classification and anomaly detection using hybrid CNN-VAE. *CoRR*, abs/1812.07203.

[77] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2018). Pointpillars: Fast encoders for object detection from point clouds.

[78] Lee, J., Han, J., and Li, X. (2008). Trajectory outlier detection: A partition-and-detect framework. In *2008 IEEE 24th International Conference on Data Engineering*, pages 140–149.

[79] Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H. S., and Chandraker, M. (2017). Desire: Distant future prediction in dynamic scenes with interacting agents. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[80] Lefévre, S., Laugier, C., and nez guzmán, J. I. (2011). Exploiting map information for driver intention estimation at road intersections. In *in Proc. IEEE Intelligent Vehicles Symposium*.

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

[82] Lerner, A., Chrysanthou, Y., and Lischinski, D. (2007). Crowds by example. *Comput. Graph. Forum*, 26:655–664.

[83] Levinson, J. and Thrun, S. (2014). *Unsupervised Calibration for Multi-beam Lasers*, pages 179–193.

[84] Li, B. (2017). 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518.

[85] Li, J., Dai, B., Li, X., Xu, X., and Liu, D. (2019). A dynamic bayesian network for vehicle maneuver prediction in highway driving scenarios: Framework and verification. *Electronics*, 8(1).

[86] Li, X., Han, J., Kim, S., and Gonzalez, H. (2007). *ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets*, pages 273–284.

[87] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Lecture Notes in Computer Science*, page 740–755.

[88] Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W., and Manocha, D. (2018). Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. *CoRR*, abs/1811.02146.

[89] Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017a). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15.

[90] Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017b). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15.

[91] Maleki, M., Chan, Y., and Arani, M. (2021). Impact of autonomous vehicle technology on long distance travel behavior. *CoRR*, abs/2101.06097.

[92] Malla, S., Dariush, B., and Choi, C. (2020). TITAN: future forecast using action priors. *CoRR*, abs/2003.13886.

[93] Mercat, J., Gilles, T., Zoghby, N. E., Sandou, G., Beauvois, D., and Gil, G. P. (2020). Multi-Head Attention for Joint Multi-Modal Vehicle Motion Forecasting. In *IEEE International Conference on Robotics and Automation*, Paris, France. Virtual conference.

[94] Messaoud, K., Deo, N., Trivedi, M. M., and Nashashibi, F. (2020). Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation.

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

[96] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. (2019). Relational recurrent neural networks for vehicle trajectory prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1813–1818.

[97] Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., and Nashashibi, F. (2021). Attention based vehicle trajectory prediction. *IEEE Transactions on Intelligent Vehicles*, 6(1):175–185.

[98] Milan, A., Rezatofighi, S. H., Dick, A. R., Schindler, K., and Reid, I. D. (2016). Online multi-target tracking using recurrent neural networks. *CoRR*, abs/1604.03635.

[99] Miller, R. and Huang, Q. (2002). An adaptive peer-to-peer collision warning system. In *Vehicular Technology Conference. IEEE 55th Vehicular Technology Conference. VTC Spring 2002 (Cat. No. 02CH37367)*, volume 1, pages 317–321. IEEE.

[100] Nikhil, N. and Morris, B. T. (2019). Convolutional neural network for trajectory prediction. *Computer Vision - ECCV 2018 Workshops*, pages 186–196.

[101] Oh, G. and Peng, H. (2020). Impact of traffic lights on trajectory forecasting of human-driven vehicles near signalized intersections.

[102] Olah, C. (2015). Understanding lstm networks.

[103] Oñoro, D., Lensky, A., and Ryu, J.-H. (2013). Connected components for a fast and robust 2d lidar data segmentation. pages 160–165.

[104] Palli Thazha, V., Filliat, D., and Ibañez-Guzmán, J. (2019). Applying map-masks to Trajectory Prediction for Interacting Traffic-Agents. 3rd Edition Deep

Learning for Automated Driving (DLAD) workshop, IEEE International Conference on Intelligent Transportation Systems (ITSC'19).

[105] Palli-Thazha, V., Filliat, D., and Ibañez-Guzmán, J. (2020). Trajectory prediction of traffic agents: Incorporating context into machine learning approaches. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–6.

[106] Pandey, G., Mcbride, J. R., and Eustice, R. M. (2011). Ford campus vision and lidar data set. *Int. J. Rob. Res.*, 30(13):1543–1552.

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

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

[109] Pool, E., Kooij, J., and Gavrila, D. (2019). Context-based cyclist path prediction using recurrent neural networks. pages 824–830.

[110] Premebida, C., Ludwig, O., and Nunes, U. (2009). Lidar and vision-based pedestrian detection system. *Journal of Field Robotics*, 26(9):696–711.

[111] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation.

[112] Raack, T. (2018). Autonomous vehicle technology: System integration with ros.

[113] Rasouli, A., Kotseruba, I., and Tsotsos, J. K. (2017). Agreeing to cross: How drivers and pedestrians communicate.

[114] Rasouli, A., Kotseruba, I., and Tsotsos, J. K. (2017). Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 206–213.

[115] Ridel, D. A., Deo, N., Wolf, D., and Trivedi, M. M. (2019). Understanding pedestrian-vehicle interactions with vehicle mounted vision: An lstm model and empirical analysis.

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

[117] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

[118] Sadeghian, A., Alahi, A., and Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *2017 IEEE International Conference on Computer Vision (ICCV)*.

[119] SAE (2018). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.

[120] Said, M., Zajdela, E. R., and Stathopoulos, A. (2021). Accelerating the adoption of disruptive technologies: The impact of covid-19 on intention to use self-driving vehicles.

[121] Schneemann, F. and Heinemann, P. (2016). Context-based detection of pedestrian crossing intention for autonomous driving in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2243–2248.

[122] Schöller, C., Aravantinos, V., Lay, F., and Knoll, A. (2019). What the constant velocity model can teach us about pedestrian motion prediction.

[123] Smith, M., Baldwin, I., Churchill, W., Paul, R., and Newman, P. (2009). The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599.

[124] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. (2019). Scalability in perception for autonomous driving: Waymo open dataset.

[125] Teichman, A., Levinson, J., and Thrun, S. (2011). Towards 3d object recognition via classification of arbitrary object tracks. pages 4034–4041.

[126] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., and Mahoney, P. (2006). Stanley: The robot that won the darpa grand challenge. *J. Field Robotics*, 23:661–692.

[127] Tonoki, H., Yorozu, A., and Takahashi, M. (2017). Model-based pedestrian trajectory prediction using environmental sensor for mobile robots navigation. *International Journal of Advanced Computer Science and Applications*, 8(2).

[128] Tran, D., Bourdev, L. D., Fergus, R., Torresani, L., and Paluri, M. (2014). C3D: generic features for video analysis. *CoRR*, abs/1412.0767.

[129] Treml, M., Arjona-Medina, J. A., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., Nessler, B., and Hochreiter, S. (2016). Speeding up semantic segmentation for autonomous driving.

[130] van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

[131] Wang, P., Huang, X., Cheng, X., Zhou, D., Geng, Q., and Yang, R. (2019). The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*.

[132] William, M. M., Zaki, P. S., Soliman, B. K., Alexsan, K. G., Mansour, M., El-Moursy, M., and Khalil, K. (2019). Traffic signs detection and recognition system using deep learning. *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*.

[133] Wirthmüller, F., Schlechtriemen, J., Hipp, J., and Reichert, M. (2020). Towards incorporating contextual knowledge into the prediction of driving behavior.

[134] Zhang, Y. and Rabbat, M. (2018). A graph-cnn for 3d point cloud classification.

[135] Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., and Wu, Y. N. (2019). Multi-agent tensor fusion for contextual trajectory prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[136] Zhou, Y. and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Titre : Utilisation d'indices contextuels et d'interactions pour la prédiction de la trajectoire des usagers de la route

Mots clés : Prédiction de trajectoire, LSTM, apprentissage automatique, indices contextuels, cartography, fusion

Résumé : La navigation autonome des véhicules dans les zones urbaines implique des interactions avec les différents usagers de la route ou agents de la circulation partageant le même réseau routier comme les voitures, les vélos et les piétons. La capacité du véhicule autonome à observer, comprendre et prédire le comportement de ces agents est très importante pour acquérir une bonne compréhension de la situation avant de décider de la manœuvre à suivre.
La thèse construit progressivement une architecture d'apprentissage automatique basée sur une formulation théorique et des expérimentations. Notre approche est basée sur un modèle d'encodeur-décodeur LSTM qui accepte les données de différentes entrées. Des observations de trajectoire à partir de données de nuages de points LiDAR 3D et d'informations sémantiques à partir de masques de carte sont utilisées. Les masques de cartes représentent des zones où les agents peuvent opérer ou non, de manière binaire. Les informations sur l'attention des piétons aux véhicules venant en sens inverse obtenues à partir des images des caméras sont également exploitées pour enrichir le système de prédiction de séquence. De plus, les interactions du véhicule autonome avec les agents de la circulation régissent souvent son comportement lorsque le véhicule navigue. Un mécanisme pour incorporer ces informations au modèle d'apprentissage est également développé aboutissant à un système de prédiction de trajectoire intégrant les interactions et des indices contextuels. Nos expériences ont permis de valider nos modèles et de construire progressivement leur architecture. Leurs performances sont démontrées à l'aide du célèbre jeu de données NuScenes acquis en milieu urbain. Les performances de l'approche proposée comparées aux approches basées sur des modèles et des données démontrent que l'ajout de multiples informations contextuelles et des interactions d'agents permet une augmentation substantielle des performances.

Title : Using context-cues and interaction for traffic-agent trajectory prediction

Keywords : Trajectory Prediction, LSTM, Machine Learning, context-cues, cartography, sensor-fusion

Abstract : Autonomous vehicle navigation in urban areas involves interactions with the different road-users or traffic-agents like cars, bicycles, and pedestrians, sharing the same road network. The ability of autonomous vehicle to observe, understand and predict the behaviour of these traffic-agents is very important to gain a good situation understanding prior to deciding what manoeuvre to follow. While this is achieved to various degrees of success using model-based or data-driven methods, human drivers remain much more efficient at this task, instinctively inferring different agent motions even in previously unseen and challenging situations. Moreover, context plays a very important role that enables us humans to understand what is being perceived and make finer predictions. The need to increase situational awareness of autonomous vehicles, as well as for safety related driving assistance functions, stimulates our goal to exploit contextual information to predict the future trajectories of the observed traffic-agents in different conditions.
The thesis gradually builds a machine learning architecture based on a theoretical formulation and experimentation. Our approach is based on an LSTM encoder-decoder model that accepts data from different inputs. Map masks represent areas where the traffic-agents can operate or not, in a binary manner. The information on pedestrian attention to oncoming vehicles obtained from camera images is also exploited to enrich the sequence prediction system. Moreover, interactions of the autonomous vehicle with traffic-agents often govern its behaviour as the vehicle navigates. A mechanism to incorporate this information to the machine learning model is also developed as an interaction-aware trajectory prediction system enhanced by context-cues. Experiments were performed for our models to learn, and gradually build the resulting architecture. Their performance are demonstrated using the well-known NuScenes dataset acquired in urban settings. The performance of the proposed approach were compared with model and data-driven approaches, demonstrating that the incorporation of multiple contextual information and agent interactions provides a substantial performance increase.

Institut Polytechnique de Paris
91120 Palaiseau, France