Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 872

Hao Cheng

Deep Learning of User Behavior in Shared Spaces

München 2021

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325 | ISBN 978-3-7696-5284-0

Diese Arbeit ist gleichzeitig veröffentlicht in:

Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

ISSN 0174-1454, Nr. 371, Hannover 2021

Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 872

Deep Learning of User Behavior in Shared Spaces

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

Vorgelegt von

M. Sc. Hao Cheng

Geboren am 11.10.1987 in Hubei, China

München 2021

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5284-0

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 371, Hannover 2021

Adresse der DGK:

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 331 – 288 1685 ● Telefax +49 – 331 – 288 1759

E-Mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:

Vorsitzender: Prof. Dr.-Ing. Jakob Flury

Referentin: Prof. Dr.-Ing. habil. Monika Sester

Korreferenten: Prof. Dr.-Ing. habil. Christian Heipke
Prof. Dr.-Ing. Bodo Rosenhahn

Tag der mündlichen Prüfung: 18.03.2021

ISSN 0065-5325 ISBN 978-3-7696-5284-0

Kurzfassung

Das Verhalten von Verkehrsteilnehmern zu erlernen ist für zahlreiche intelligente Systeme von großer Bedeutung, sei es in der Steuerung der Verkehrssicherheit, in der Gestaltung intelligenter Transportsysteme oder in der Entwicklung autonomer Fahrzeuge. Allerdings stellt für die Realisierung solcher Systeme im von Dynamik und Unwägbarkeiten geprägten Stadtverkehr die automatisierte und genaue Erkennung des Verhaltens von Verkehrsteilnehmern noch immer eine Herausforderung dar. Es gibt urbane Umgebungen, z. B. vorübergehende Shared Spaces auf Kreuzungen bei Abbiegevorgängen oder Shared-Space-Verkehrsdesigns, in denen sich die Erkennung und die Vorhersage des Verhaltens der Verkehrsteilnehmer besonders schwierig gestalten. Shared Spaces auf Kreuzungen ermöglichen das Abbiegen von Fahrzeugen, während diese mit anderen Verkehrsteilnehmern, die die Straße überqueren, interagieren. Shared-Space-Verkehrsdesigns kommen zum Einsatz, wenn es darum geht, den Verkehrsraum so zu strukturieren, dass die Dominanz der Fahrzeuge verringert sowie die Fußgängerbewegung und der Komfort für Fußgänger verbessert werden. Aufgrund direkter Interaktionen zwischen Fahrzeugen und ungeschützten Verkehrsteilnehmern (vulnerable road users (VRUs), z. B. Fußgänger oder Fahrradfahrer) sowie mehrdeutiger Verkehrssituationen (Verkehrsteilnehmer müssen die Straßennutzung untereinander aushandeln) ist das Verhalten der Verkehrsteilnehmer stochastisch und schwer vorhersagbar.

Mit Bezug auf aktuelle Entwicklungen im Bereich Deep Learning sowie auf die Verfügbarkeit von umfangreichen, realen Verkehrsdaten verfolgt diese Arbeit den Ansatz, bedingte generative Modelle für die automatische Detektion von Interaktionen in vorübergehenden Shared Spaces auf Kreuzungen und die Vorhersage von Trajektorien in Shared Space-Verkehrsdesigns zu verwenden. Auf dem Conditional Variational Auto-Encoder (CVAE) basierende Modelle werden darauf trainiert, deterministische Eingaben (z. B. eine Sequenz von Videodaten oder eine beobachtete Trajektorie) auf viele mögliche Varianten des Verhaltens von Verkehrsteilnehmern abzubilden, welches durch die jeweilige Interaktion oder Bewegung der betreffenden Verkehrsteilnehmer in den nächsten Sekunden gekennzeichnet ist.

Diese Arbeit leistet zwei wesentliche Beiträge zur Modellierung des Verhaltens von Verkehrsteilnehmern in Shared Spaces mithilfe von Deep-Learning-Ansätzen:

(1) Das Interaktionserkennungsmodell verarbeitet Informationen über Art, Position und Bewegung der Verkehrsteilnehmer, die automatisch durch Objektdetektoren – umgesetzt mittels Deep Learning – sowie optischen Fluss aus Videodaten extrahiert werden, und generiert bildweise eine Wahrscheinlichkeit, die die Interaktionsdynamik zwischen einem abbiegenden Fahrzeug und allen beteiligten VRUs repräsentiert. Die Wirksamkeit des Modells wurde durch Tests an realen Datensätzen von zwei verschiedenen Kreuzungen mit starkem Verkehrsfluss nachgewiesen. Es erreichte einen F1-Wert von über 0,96 an einer Rechtsabbieger-Kreuzung in Deutschland und 0,89 an einer Linksabbieger-Kreuzung in Japan.

(2) Verschiedene Faktoren und Deep-Learning-Architekturen auf dem aktuellen Stand der Technik werden hinsichtlich der Vorhersage von Trajektorien untersucht. In dieser Arbeit werden drei auf CVAE basierende Frameworks zur genauen Multi-Pfad-Trajektorienvorhersage von heterogenen Verkehrsteilnehmern in Shared Spaces vorgeschlagen. Der Latenzraum des CVAE wird für die Kodierung stochastischer Verhaltensmuster trainiert. Der Multi-Sampling-Prozess aus dem trainierten Latenzraum ermöglicht es den Frameworks dabei, nicht nur eine deterministische zukünftige Trajektorie, sondern mehrere mögliche zukünftige Trajektorien für jeden Verkehrsteilnehmer zu generieren. Das erste Framework konzentriert sich auf die Untersuchung mehrerer Kontexte, nämlich Bewegung, Interaktion und Fußgängergruppierung sowie auf verschiedene Umgebungsbedingungen für die Trajektorienvorhersage. Das zweite und dritte Framework konzentrieren sich indes einereits auf die Untersuchung dynamischer Kontexte (d. h. Bewegung und Interaktion) unter Verwendung

von Aufmerksamkeitsmechanismen und andererseits auf die Verbesserung der Generalisierbarkeit der Modelle - d. h. auf die Vorhersage der Trajektorien heterogener Verkehrsteilnehmer in verschiedenen Shared Spaces, auf die das Modell zuvor nicht trainiert wurde. Die Leistung aller drei Frameworks, insbesondere des zweiten und dritten, erwies sich bei diversen gängigen Open-Source-Datensätzen und -Benchmarks als überlegen und erreichte den ersten Platz (in verschiedenen Abgabezeiten) bei einer bekannten offenen Challenge (TrajNet online test), bei der die in dieser Arbeit vorgestellten Modelle den gesamten durchschnittlichen bzw. finalen Displacement Error der für die nächsten 4.8 Sekunden vorhergesagten Trajektorien um 0.353 Meter bzw. 1.179 Meter reduzieren konnten.

Schlagworte: Gemischter Verkehr, Shared Space, ungeschützte Verkehrsteilnehmer, Nutzerverhalten, Trajektorienvorhersage, Interaktionserkennung, Deep Learning, neuronale Netze, bedingtes generatives Modell, Sequenz-zu-Sequenz

Abstract

Learning how road users behave is essential for the development of many intelligent systems, such as traffic safety control, intelligent transportation systems, and self-driving cars. However, automated and accurate recognition of road users' behavior is still one of the bottlenecks in realizing such systems in city traffic that is—compared to other types of traffic—especially dynamic and full of uncertainties. There are some urban environments which make detection and prediction of road users' behavior particularly challenging, e. g., temporarily shared spaces of intersections for vehicle turning or shared spaces as a traffic design. The former allow vehicles to turn and interact with other crossing road users, the latter intended to make different types of road users share the space, therefore reducing the dominance of vehicles and improving pedestrian movement and comfort. Direct interactions between vehicles and vulnerable road users (VRUs, e. g., pedestrians and cyclists) and ambiguous traffic situations (e. g., road users negotiating usage of the road) make road users' behavior stochastic and difficult to predict.

With the development of deep learning techniques and the availability of large-scale real traffic data, this thesis proposes deep conditional generative models for automated interaction detection in the temporarily shared spaces of intersections, as well as for trajectory prediction in shared spaces as a traffic design. Models based on Conditional Variational Auto-Encoder (CVAE) are trained to map deterministic input (e. g., a sequence of video data or a segment of an observed trajectory) to many possible outputs of road users' behavior, characterized by their interaction or their movement in the next seconds.

This thesis makes two main contributions to the research on modeling road users' behavior in shared spaces using deep learning approaches:

(1) The interaction detection model takes the information of road users' type, position, and motion—all of which have been automatically extracted by deep learning object detectors and optical flow from video data—as input, and generates a frame-wise probability that represents the dynamics of interaction between a turning vehicle and any VRUs involved. The model's efficacy was proven by testing on real-world datasets acquired from two different intersections. It achieved an F1-score above 0.96 at a right-turn intersection in Germany and 0.89 at a left-turn intersection in Japan, both with very busy traffic flows.

(2) Various factors and state-of-the-art deep learning architectures are investigated for trajectory prediction. In this thesis, three frameworks based on CVAE are proposed for accurate multi-path trajectory prediction of heterogeneous road users in shared spaces as a traffic design. The latent space of the CVAE is trained for encoding stochastic behavior patterns and the multi-sampling process from the trained latent space enables the frameworks to generate not only one deterministic future trajectory, but multiple possible future trajectories for each road user. The first framework focuses on studying multiple contexts, namely motion, interaction, pedestrian grouping, and different types of environmental scene context for trajectory prediction. The second and third frameworks focus on exploring dynamic context (i. e., motion and interaction) using attention mechanisms, and improving the models' generalizability—predicting trajectories of heterogeneous road users in various shared spaces that have not been used to train the models. All of the frameworks, but the second and third in particular, showed superior performance on various popular open-source datasets and benchmarks. The last two frameworks even took first place (in different submission times) in one of the most widely recognized open challenges (TrajNet online test) by reducing the overall average and final displacement errors of the predicted trajectories in the next 4.8 seconds to 0.353 meters and 1.179 meters, respectively.

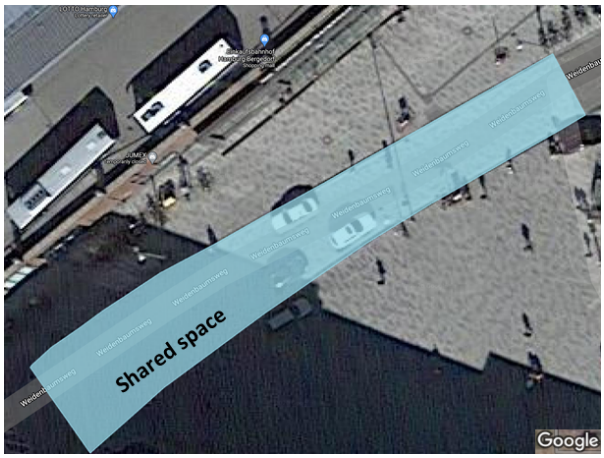
Keywords: Mixed traffic, shared space, vulnerable road user, road user behavior, trajectory prediction, interaction detection, deep learning, neural networks, conditional generative model, sequence-to-sequence

Contents

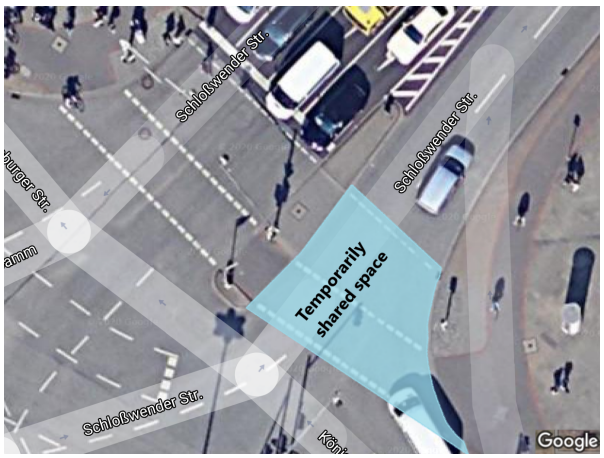
1 Introduction

1.1 Shared Space

In real-world traffic situations, it is not uncommon that heterogeneous road users such as vehicles and Vulnerable Road Users (VRUs) like pedestrians or cyclists have to directly interact with each other in a particular location. Apart from being assigned to dedicated lanes or time slots by traffic lights for traffic segregation, different types of road users are usually allowed to self-organize when they are using the road simultaneously. There are traffic locations particularly constructed to encourage mixed road use. One such location that has been gaining popularity in recent years is *shared space as a traffic design* (Reid, 2009). The other type commonly seen in city traffic is *temporarily shared spaces* at some intersections, including the turning area of the so-called Turn-on-Red (TOR) intersections or, more generally, intersections that allow vehicles to turn while other road users are crossing. Examples of both types of shared spaces are given in Fig. 1.1.



(a) Example of shared space as a traffic design



(b) Example of a temporarily shared space

Figure 1.1: Examples of shared spaces in Germany, (a) a street with pedestrians crossing from both sides without traffic signals, and (b) a permissive right-turn intersection in right-hand traffic with pedestrians and cyclists crossing the street (background image: Imagery ©2020 Google, Map data ©2020 GeoBasis–DE/BKG(©2009), Google).

Shared space as a traffic design

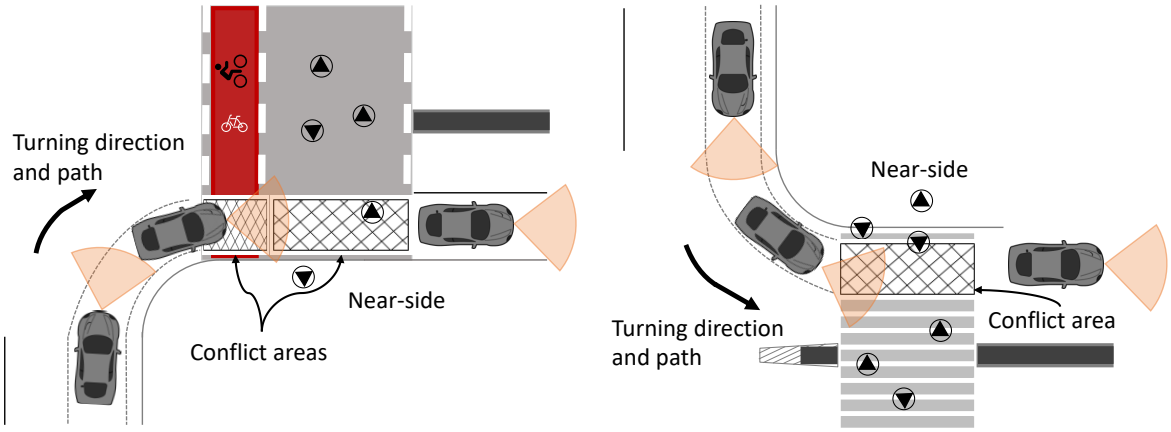
Unlike classic traffic designs that normally dedicate road resources to road users by time or space division, an alternative traffic design—shared space—has been proposed by traffic engineers. This concept was first introduced by the Dutch traffic engineer Hans Monderman in the 1970s (Clarke, 2006). It was later formally defined by Reid as "a street or place designed to improve pedestrian movement and comfort by reducing the dominance of motor vehicles and enabling all users to share the space rather than follow the clearly defined rules implied by more conventional designs" (Reid, 2009). This design largely removes road signs, markings, and traffic lights, allowing vehicles moving at a limited speed (commonly under 30 km/h) to directly interact with pedestrians and cyclists. Mixed road users negotiate to take or give their right-of-way based on social and physical context (Hamilton-Baillie and Jones, 2005). Shared spaces as a traffic design nowadays can be found in

urban areas of many European cities, such as the Laweiplein intersection in the Dutch town of Drachten, Skvallertorget in Norrköping, and Kensington High Street in London (Hamilton-Baillie, 2008), as well as in many other countries.

The uncertainties brought on by the lack of explicit rules in shared spaces have triggered contradictory discussions. On the one hand, the removal of traffic signals and rules may lead to more vehicle–pedestrian conflicts than the conventional design with unambiguous clarity (Kaparias et al., 2013). This especially concerns particular groups of road users, such as disabled and elderly people who are not good at judging ambiguous situations and are not as acute as others when confronted with a vehicle (Methorst et al., 2007). On the other hand, shared space purposely introduces ambiguity with the hope that road users behave more cautiously when they cannot rely on explicit rules, thereby increasing the level of safety (Hamilton-Baillie, 2004; Hamilton-Baillie and Jones, 2005). For example, in shared spaces, vehicle drivers need to conduct their driving with courtesy and VRUs need to carefully evaluate the traffic situation before crossing.

Temporarily shared space

At some intersections, vehicles are permitted to turn while other road users are crossing. One special type among these intersections is the so-called Turn-on-Red (TOR) intersection. The legislation of the concept of TOR goes back to 1937 in the United States. Californian law—Section 21453 (b) of the California Vehicle Code (CVC)—then permitted vehicles to turn right at traffic lights showing a red signal (McGee and Warren, 1976; Fleck and Yee, 2002; Yi et al., 2012). Later on, the idea of TOR intersections was adopted by many other countries. Besides TOR intersections, it is more common to see permissive right-turn intersections in right-hand traffic countries and permissive left-turn intersections in left-hand traffic countries. Vehicles are allowed to temporarily share the turning area with other road users currently crossing same road.



(a) A right-turn intersection in Germany

(b) A left-turn intersection in Japan

Figure 1.2: Examples of temporarily shared spaces. (a) In Germany, a dedicated lane for cyclists is typically parallel to the crossing zone. (b) In Japan, left-turning vehicles interact with pedestrians in the crossing zone. Figure (b) is partially adapted from (Alhajyaseen et al., 2012).

In comparison to most conventional intersections strictly governed by traffic signals for segregating different types of road users, a TOR or permissive right-turn/left-turn intersection involves interactions between vehicles and VRUs. In Germany, as shown in Fig. 1.2a, a turning vehicle at a permissive right-turn intersection often encounters cyclists that are passing by and pedestrians that are cross-walking in the conflict areas. In Japan, as shown in Fig. 1.2b, a similar situation can be found at a permissive left-turn intersection where vehicles and pedestrians interact (Alha-

jyaseen et al., 2012). During the time window necessary for vehicles to turn, these vehicles need to directly interact with VRUs. Their behavior is largely guided by social protocols (right-of-way or courtesy). In this manner, these intersections with their conflict areas are treated as temporarily shared spaces.

The temporarily shared spaces of intersections have some unique impacts on traffic. Multiple studies (Tarko, 2001; Fleck and Yee, 2002; Wong et al., 2004; Massaad and Massaad, 2020) have proven that the operation of such intersections provides additional capacity for the turning lane, reduces unnecessary delays for the turning vehicles and even lessens frustration during the wait. Furthermore, it reduces energy consumption and pollution while slightly increasing the overall capacity of the intersection. Counterintuitively, the operation of a TOR intersection does not show a statistical connection with increased traffic accidents (Compton and Milton, 1994; Fleck and Yee, 2002; Yi et al., 2012). In some cases, it may even reduce vehicle–pedestrian collisions (Wong et al., 2004).

1.2 Motivation and Research Objectives

Since their emergence, shared spaces have been studied and analyzed by different disciplines. One important topic is recognition and prediction of road users' behavior and analyzing critical situations. Statistics show that accidents between vehicles and VRUs often occur in places where they confront each other (Choi, 2010; Habibovic and Davidsson, 2011; Shirazi and Morris, 2016). The encouragement of direct interactions between vehicles and VRUs has caused a lot of concern about traffic safety in shared spaces (Preusser et al., 1982; Alhajyaseen et al., 2012; Hamilton-Baillie, 2008; Gerlach et al., 2009). Nowadays, with the ubiquity of traffic data and the development of computer vision techniques, there is a high chance of automatically recognizing individual road users' behavior from massive video data, especially detecting interactions between vehicles and VRUs from various traffic scenes and then differentiating dangerous behavior (e. g., being unaware of the other approaching road users or violating traffic rules) from normal behavior (e. g., acting courteously and following traffic rules). Meanwhile, the foreseeable advent of autonomous driving in urban areas (Franke et al., 1998), particularly in shared spaces with weakened traffic rules, highly depends on automated systems that can correctly predict other road users' behavior, i. e., their intended trajectories in the next seconds. Accurate trajectory prediction of road users is a crucial task in many other academic fields as well, not only for autonomous driving and Intelligent Transportation Systems (ITS) (Morris and Trivedi, 2008; Cheng and Sester, 2018b), but also for photogrammetry (Schindler et al., 2010; Klinger et al., 2017; Cheng and Sester, 2018a), computer vision (Alahi et al., 2016; Mohajerin and Rohani, 2019) and mobile robot applications (Mohanan and Salgoankar, 2018). Such systems are required to automatically process the input data extracted from the involved road users and environment in order to generate desirable output that represents how road users interact and move, so as to support the decisions to be made in the following steps.

However, road users' behavior in shared spaces is dynamic and complex. Automatically learning their behavior, therefore, is very challenging. In various situations of mixed traffic[1], a human road user subconsciously anticipates others' possible behavior (Gindele et al., 2010) based on the information of road geometry, transport mode, distance, travel speed, orientation, audio signals, body posture, gestures, and even small hints such as facial expressions, to adjust her or his own movement accordingly. On the other hand, the dynamics of movement and mutual influence

[1] In this thesis, mixed traffic refers to the traffic that involves different types of human road users, which distinguishes it from the mixed traffic of human road users and automated vehicles.

between road users make their behavior stochastic. For instance, one road user may have to change direction or travel speed to cope with a sudden change of motion from others. Moreover, mixed types and varying numbers of roads users, as well as direct confrontations between vehicles and VRUs greatly complicate foreseeing road users' behavior in shared spaces. Unlike human road users who gain and refine the ability of understanding and predicting other people's behavior by learning from experience over time (Gallagher and Frith, 2003), it is a big challenge for systems like ITS or self-driving cars to automatically learn and predict road users' behavior in such kinds of environment. Given the ambiguities and uncertainties of traffic rules that are intentionally introduced to shared spaces as well as the stochastic behavior of human road users, there is still a long way to go until automated systems achieve human-level performance.

There are different methods for modeling road users' behavior. The conventional methods use hand-crafted rules and features to mimic road users' behavior, such as force-based rules (Helbing and Molnar, 1995) with attractive and repelling forces for influencing road user agents' movement, Cellular Automata (CA) (Bandini et al., 2017b) using a set of predefined rules for simulating the movement of agents, and distance-based energy functions for modeling interactions between agents (Pellegrini et al., 2009). Their performance is affected by the quality of manually designed features and their lack of generalizability. Recently, with the advent of large real-world datasets and the development of deep learning technologies (LeCun et al., 2015), deep learning methods keep reporting state-of-the-art performance on benchmarks for behavior prediction (Alahi et al., 2016; Gupta et al., 2018; Cheng et al., 2021c). Instead of using manually designed features and rules, deep learning is a class of methods that automatically discover intricate structures of large datasets by using representations of different levels of abstraction (LeCun et al., 2015).

This thesis is targeted at using deep learning methods that are trained using various sources of real-world data for learning road users' behavior in shared spaces. Due to the distinct constructions of the two types of shared spaces described above, different aspects of road users' behavior are investigated. In the temporarily shared spaces of intersections, the trajectories of agents are, to some extent, predefined by road geometry and markings, e. g., vehicle turning and VRUs crosswalking (as shown in Fig. 1.1b). It is important to learn how vehicles and VRUs interact with each other, i. e., whether the continuity of their behavior is interrupted by each other's presence and actions. However, if road users are not explicitly constrained by the space layout, they (especially VRUs) can move freely in different directions within the shared spaces (as shown in Fig. 1.1a). Predicting the agents' trajectory, e. g., their continuous orientation and speed, is the major task. Therefore, two respective aspects of road users' behavior are investigated: (I) *interaction detection* of vehicle turning sequences in the temporarily shared spaces of intersections and (II) *trajectory prediction* in shared spaces as a traffic design.

With regard to traffic and the agents involved, there are different definitions of *interaction*. The concept of interaction with VRUs is closely related to the concept of *conflict* (Perkins and Harris, 1968). Interaction can range from collision to negligible conflict risk (Sayed and Zein, 1999; Svensson and Hydén, 2006). Saunier and Sayed (2008) defined an interaction as "a situation in which two or more road users are close enough in space and time and their distance is decreasing". Similarly, Svensson and Hydén (2006) described an interaction between road users as "a continuum of safety related events". Both concepts emphasize that an interaction is not static but rather dynamic and evolves over time, which represents a changing level of reaction between road users. In this thesis, interaction detection is defined as follows:

> Interaction is needed if the turning vehicle drives into an intersection while any VRUs are approaching or moving within the intersection space, in order to avoid any conflicts that might happen at any time during the vehicle's turning, they adapt their movement, i. e., velocity and orientation, accordingly. In contrast to this, no interaction is needed

> if the target vehicle drives in an undisturbed manner with VRUs—if there are any—in its neighborhood. The task of interaction detection is to differentiate interaction and non-interaction levels through the dynamics of the vehicle turning sequence.

Moreover, this thesis follows previous works (Helbing and Molnar, 1995; Alahi et al., 2016; Rudenko et al., 2020) in its definition of trajectory prediction:

> Trajectory prediction means predicting in 2D or 3D the plausible (e. g., collision free) and socially-acceptable (e. g., considering social rules and relations between agents) positions of non-erratic target agents at each discrete time step within a predefined future time interval relying on observed partial trajectories over a certain period of time. A target agent is defined as the dynamic object for which the actual prediction is made. In this thesis, an agent refers to a road user that could be a pedestrian, a cyclist, a vehicle, or any other type of road user.

1.3 Proposals and Contributions

The tasks of interaction detection and trajectory prediction in shared spaces require learning the stochastic behavior of heterogeneous road users. As discussed above, both types of shared spaces described in Sec. 1.1 are dynamic and complex. The behavior of a road user can present multimodalities in interactions with others. Here, multimodality means that the road user in a traffic situation can act differently, e. g., move at a different speed in different directions and choose one path out of multiple possible paths. Hence, it is essential for a model to map one deterministic input to many possible outputs. In the context of interaction detection and trajectory prediction, the deterministic input is the observation of the road user's past motion, such as a clip of a video recording at an intersection or the road user's past trajectory, and the output is one of the many possible patterns of motion to be recognized in the next seconds.

This thesis proposes to use deep generative models to address the above one-to-many mapping problem. Deep generative models are the class of deep learning methods that learn the distributions of the input data and generate new data instances based on the learned distributions. In this, they are different from the other class of discriminative deep learning methods. For example, discriminative models commonly resort to objective functions that minimize the distance between the pair of ground truth and predicted output. These objective functions often lead to the models predicting the "average" results. The commonly used generative models are Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and Variational Auto-Encoder (VAE) (Kingma and Welling, 2014; Kingma et al., 2014), as well as the extension of VAE—Conditional Variational Auto-Encoder (CVAE) (Rezende et al., 2014; Sohn et al., 2015). Both GANs and VAE are able to generate diverse outputs by sampling from some prior distribution with uncertainty, such as Gaussian distribution. Interaction detection and trajectory prediction are conditioned on the observations of road users' past motion. Hence, more specifically, this thesis proposes to use conditional generative models based on CVAE. The proposed models incorporate a generative module that is trained to encode the motion and interaction patterns with a set of random variables, the so-called Gaussian latent variables. In this way, the models are able to perform probabilistic inference and make diverse predictions conditioned on the observation of past motion.

In addition, attention mechanisms are incorporated into the generative models to automatically extract spatio–temporal relationships for predicting road users' behavior. Attention mechanism is a technique used in deep neural networks for mapping the salient features of the input data to the output data. It was initially invented for machine translation but quickly applied to image processing as well (Xu et al., 2015). Nowadays, the Transformer network with a self-attention

mechanism (Vaswani et al., 2017) is the most widely used attention network for processing sequential data.

Large-scale realistic data is leveraged for training and testing conditional generative models for interaction detection and trajectory prediction. Table 1.1 lists all the datasets that are used in this thesis, which cover a wide variety of real-world traffic activities in different places and countries. Interaction detection is carried out by directly using video frame sequences that record varying traffic situations between vehicles and VRUs in the temporarily shared spaces of two intersections. Trajectory prediction is carried out by using the trajectories provided by other datasets, in which all the trajectories have been projected on a 2D plane with a bird's-eye view for trajectory prediction. All the datasets were acquired from a third-person perspective by a static camera for both the interaction detection and trajectory prediction tasks. Unlike approaches using the ego-perspective from a target agent (Geiger et al., 2013), the focus of this thesis is the behavior of any dynamic object (agent) on the surface of the areas of interest. A third-person perspective facilitates the observation of all agents, rather than focusing on a single one.

Table 1.1: List of datasets used in this thesis.

Dataset	Reference	Country	Public	Type	View	Application
KoW	Koetsier et al. (2019)	Germany	no	intersection	oblique	interaction detection
NGY	Cheng et al. (2020c)	Japan	no	intersection	oblique	interaction detection
HC	Cheng et al. (2019)	Germany	yes	shared space	bird's-eye	trajectory prediction
HBS	Pascucci et al. (2017)	Germany	no	shared space	bird's-eye	trajectory prediction
Gates3	Robicquet et al. (2016)	US	yes	shared space	bird's-eye	trajectory prediction
TrajNet*	Sadeghian et al. (2018a)	mainly US	yes	shared spaces	bird's-eye	trajectory prediction
inD	Bock et al. (2019)	Germany	yes	intersections	bird's-eye	trajectory prediction

*TrajNet is a super-set that contains several other public datasets; more details are given in Sec. 6.2.2.1

As the specific challenges may differ for the tasks of interaction detection and trajectory prediction, corresponding strategies are used to tackle the respective challenges. Therefore, these two tasks are further broken down and addressed accordingly.

1.3.1 Interaction Detection

To achieve automated detection of interactions between vehicles and VRUs, the following challenges have to be tackled: (I) How to efficiently acquire, process, and label a large amount of video data for training a deep learning model for interaction detection considering all relevant road users? (II) How to automatically detect the location and motion of the involved road users? (III) How to represent the dynamics of interactions in vehicle turning sequences of varying duration?

Considering the above challenges, a generative model is proposed for detecting interactions between turning vehicles and VRUs directly from traffic video data. The contributions of this thesis, therefore, are as follows:

1) *Processing large-scale real-world datasets in both right- and left-hand traffic.* Various activities among all road user types were recorded using a camera at a right-turn intersection in Germany and a left-turn intersection in Japan for very busy traffic flows. The video data was manually annotated and divided into vehicle-turning sequences with interaction class labels *interaction* and *non-interaction* by multiple annotators.

2) *Automatically extracting dynamic information.* Deep learning object detectors were leveraged to automatically detect all related road users. At the same time, optical flow was used to extract the motion of road users. The combination of object detection and motion extraction enables the model to capture the dynamics of all the road users and circumvent the tremendous work of manual tracking.

3) An *end-to-end conditional generative model with a self-attention mechanism for interaction detection.* After the data acquisition and pre-processing, an end-to-end conditional generative model is proposed for interaction detection. The proposed model uses the extracted sequence of both object and motion information simultaneously and generates a probability of interaction at each short interval (< 0.1 seconds). The probability changes accordingly when the intensity of interaction changes between a turning vehicle and VRUs over time.

1.3.2 Trajectory Prediction

To achieve effective and accurate trajectory prediction of heterogeneous agents, the following questions have to be addressed: (I) How to model the complex behavior and uncertain intention of each agent? (II) How to map the interactions between dynamic objects (i. e., interactions between the target agent and its neighboring agents) and static environment (i. e., the constraints by physical contexts such as buildings, vegetation and obstacles)? (III) As there is usually more than one socially acceptable path that an agent could take in the future, how to predict multiple possible paths?

To address this second set of challenges, conditional generative models are proposed to generate diverse patterns of future trajectories of heterogeneous agents in various shared spaces. The contributions are summarized as follows:

1) *Modeling agent-to-agent interactions by mapping.* In order to explore how to accurately capture the interactions between agents, different mapping mechanisms are investigated, such as occupancy grid, dynamic maps with a self-attention mechanism, and differentiating group and non-group behavior using density-based clustering.

2) *Modeling agent-to-environment interactions.* To explore the effect of the environment, three types of scene context are studied: *motion heat maps* that describe the prior of how different agents move; an *aerial photograph image* that provides global visual information of the scene; and *accessibility maps* that define the accessible areas with respect to the road agents' transport mode, such as a pedestrian, cyclist, or vehicle.

3) *Multi-path trajectory prediction.* The generative models predict multiple socially acceptable and plausible trajectories conditioned on the given past trajectory for all the heterogeneous agents in shared spaces.

1.4 Structure of the Thesis

This thesis is structured as follows:

Chapter 2 gives a brief introduction to deep learning (Sec. 2.1) and provides the fundamental methods adopted in this thesis for learning road users' behavior. The fundamental methods include two state-of-the-art deep learning approaches for object detection and classification (Sec. 2.2), the dense optical flow algorithm for motion extraction (sec. 2.3), the spatial clustering algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Sec. 2.4), the Transformer encoder with a self-attention mechanism (Sec. 2.5), and the theories of conditional generative models VAE and CVAE (Sec. 2.6).

Chapter 3 discusses the related works and identifies the research gaps for interaction detection (Sec. 3.1) and trajectory prediction (Sec. 3.2).

Chapter 4 details the methodological contributions made by the author for automated interaction detection using video data (Sec. 4.1) and multi-path trajectory prediction of heterogeneous road users in shared spaces (Sec. 4.2).

Chapter 5 first introduces the two datasets acquired from Germany and Japan (Sec. 5.1), and then demonstrates the detailed experiments (Sec. 5.2), results (Sec. 5.3) and discussions (Sec. 5.4) for interaction detection.

Chapter 6 investigates three different frameworks tested on various realistic open-source datasets for trajectory prediction in shared spaces, namely, Multi-Context Encoder Network (Sec. 6.1), Attentive Maps Encoder Network (Sec. 6.2), and Dynamic Context Encoder Network (Sec. 6.3).

Chapter 7 contains conclusions (Sec. 7.1) and provides an outlook to possible future research (Sec. 7.2).

2 Concepts of Deep Learning and Fundamental Methods for Behavior Modeling

This chapter provides a brief introduction to deep learning, the basic models, algorithms and networks, which have been adopted in this thesis for interaction detection and trajectory prediction. The theories and mathematics are adapted from the referenced papers.

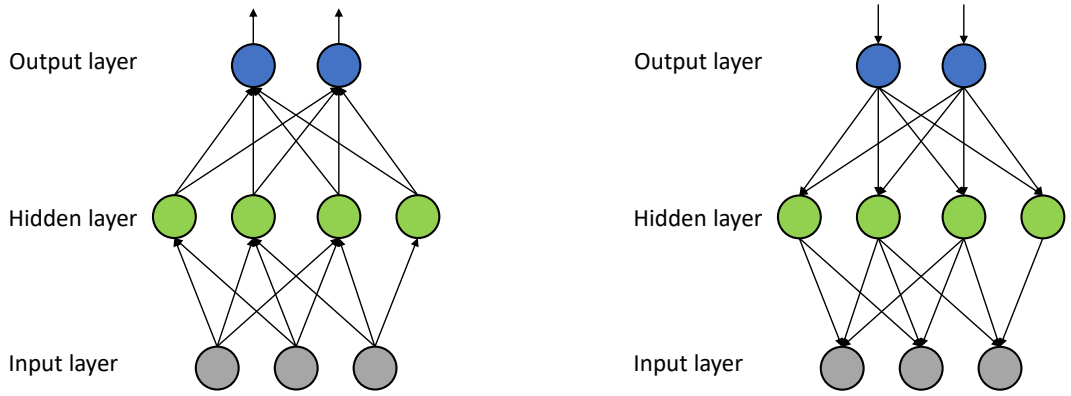
2.1 Introduction to Deep Learning

Deep learning is a type of representation learning. According to (LeCun et al., 2015), "deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representations at one level (starting with the raw input) into a representation at a higher, slightly more abstract level." These methods use the backpropagation (Rumelhart et al., 1986) algorithm to discover intricate structures in large datasets, indicating how a machine should adjust its internal parameters for computing the representation at each level (layer) from the representation at the previous levels. In the paradigm of deep learning, Convolutional Neural Networks (CNNs) have achieved remarkable successes in image processing, such as object detection and classification (Redmon et al., 2016; Redmon and Farhadi, 2017, 2018; Zhao et al., 2019a), whereas Recurrent Neural Networks (RNNs) have made considerable progress in sequential data modeling, such as trajectory prediction (Alahi et al., 2016; Lee et al., 2017; Gupta et al., 2018; Zhang et al., 2019).

This section aims to give a very brief introduction to the basic concepts of the networks that are used in this thesis. A more thorough introduction to deep learning than this thesis can provide can be found in (Goodfellow et al., 2016).

2.1.1 Feed-Forward Network with Backpropagation

The feed-forward network is the simplest neural network of deep learning, in which nodes are connected in a forward way with no cycle or loop. Fig. 2.1a shows a feed-forward network that consists of an input layer (gray), a hidden layer (green), and an output layer (blue). The nodes represent the neurons in each layer and the edges represent the connection with weights of neurons between layers. The arrows of the edges indicate the flow from the input layer to the output layer. The input after some non-linear transformation is transferred from one neuron to another. Neurons in deep learning are named after biological neurons, and non-linear transformation, also called activation, mimics the firing of biological neurons.



(a) Forward propagation *(b)* Backward propagation

Figure 2.1: An example of a feed-forward network. The nodes represent neurons, the edges represent connections between neurons, and the arrows indicate the flow direction.

Mathematically, a feed-forward network can be expressed by learning a representation from the input $\boldsymbol{x}$ to the output $\boldsymbol{y}$ through stacked non-linear transformations. A simple input–hidden–output feed-forward network is formulated as follows:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\boldsymbol{x} + b^{(1)}, \tag{2.1}$$

$$\boldsymbol{a}^{(1)} = \sigma(\mathbf{z}^{(1)}), \tag{2.2}$$

$$\mathbf{z}^{(2)} = \mathbf{W}^{2}\boldsymbol{a}^{(1)} + b^{(2)}, \tag{2.3}$$

$$\boldsymbol{y} = \sigma(\mathbf{z}^{(2)}), \tag{2.4}$$

where $\mathbf{z}$ denotes the intermediate output of a linear transformation, $\mathbf{W}$ stands for weights and b for bias. The superscripts denote the index of the layers that start from 0, i. e., the input layer. σ is a non-linear activation function.

Activation functions are commonly used in neural networks, mainly for the purposes of (1) constraining the values of $\mathbf{z}$ to a certain range and (2) introducing non-linear transformations to the neural networks. If the intermediate values are not constrained, they may expand to a very high magnitude due to, e. g., the large number of stacked layers of a neural network. This can lead to undesirable performance and computational issues. More importantly, without non-linear transformation, the stacked layers can only perform linear transformation, which keeps the network from solving more complicated and non-linear problems. In fact, many tasks such as image classification problems are non-linear problems and therefore require non-linear mapping from the input data to the output data.

The most basic non-linear activation functions are denoted by the following equations:

$$\text{Sigmoid:} \qquad \sigma(\mathbf{z}) = \frac{1}{1 + e^{-\mathbf{z}}}, \tag{2.5}$$

$$\text{tanh:} \qquad \sigma(\mathbf{z}) = \frac{e^{\mathbf{z}} - e^{-\mathbf{z}}}{e^{\mathbf{z}} + e^{-\mathbf{z}}}, \tag{2.6}$$

$$\text{ReLU:} \qquad \sigma(\mathbf{z}) = \max(0, \mathbf{z}), \tag{2.7}$$

$$\text{Softmax:} \qquad \sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}, \tag{2.8}$$

where $i = 1, \cdots N$, denoting the index of the single value of $\mathbf{z} = \{z_1, \cdots z_N\}$. The Sigmoid function returns the real values in the range of $(0, 1)$ with a monotonically increasing trend. The Hyperbolic tangent (tanh) function also has a monotonically increasing trend, but the returned real values are in the range of $(-1, 1)$ with the central value being zero. The Rectified Linear Unit (ReLU) function returns the identical values of the positive input and resets the negative input to zero, which is much easier to compute than the other activation functions and is therefore widely used for non-linear transformation in neural networks. The Softmax function is very similar to the Sigmoid function, but the sum of the values given by the Softmax function is normalized to one. It is often used to calculate the probabilities for classification problems. There are, of course, many other non-linear activation functions used in neural networks. But they are not adapted in this thesis. Hence, they are not introduced here.

The objective is to learn a set of parameters (weights and bias) that optimize the error between the output of the network and the so-called *ground truth* summed over all the data points of a dataset. This particular error is also called *loss* or *cost*, as denoted by E. Without loss of generality, let $g(x)$ denote the mapping of the input data point x by the feed-forward network. The loss is defined as

$$E = \sum_n C(y_n - g(x_n)), \tag{2.9}$$

where C stands for a loss function and n for the index of a data point.

The backpropagation algorithm (Rumelhart et al., 1986) is employed to calculate the gradients with respect to weights and bias of the multi-layer stack of modules, as shown in Fig. 2.1b. In terms of the above feed-forward network, the gradients regarding the weights and bias in each layer are calculated using the chain rule as follows:

$$\frac{\partial E}{\partial \mathbf{W}^{(2)}} = \frac{\partial E}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}^{(2)}} \frac{\partial \boldsymbol{z}^{(2)}}{\partial \mathbf{W}^{(2)}}, \tag{2.10}$$

$$\frac{\partial E}{\partial b^{(2)}} = \frac{\partial E}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}^{(2)}} \frac{\partial \boldsymbol{z}^{(2)}}{\partial b^{(2)}}, \tag{2.11}$$

$$\frac{\partial E}{\partial \mathbf{W}^{(1)}} = \frac{\partial E}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}^{(2)}} \frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{a}^{(1)}} \frac{\partial \boldsymbol{a}^{(1)}}{\partial \boldsymbol{z}^{(1)}} \frac{\partial \boldsymbol{z}^{(1)}}{\partial \mathbf{W}^{(1)}}, \tag{2.12}$$

$$\frac{\partial E}{\partial b^{(1)}} = \frac{\partial E}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}^{(2)}} \frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{a}^{(1)}} \frac{\partial \boldsymbol{a}^{(1)}}{\partial \boldsymbol{z}^{(1)}} \frac{\partial \boldsymbol{z}^{(1)}}{\partial b^{(1)}}. \tag{2.13}$$

Afterwards, gradient descent is applied to adjust the weights and bias of the network in each layer. For example, the formulas (2.14) and (2.15) calculate weight and bias, respectively, for the l-th layer. The learning rate lr specifies how fast these adjustments should be made according to the gradients. In this context, it should be noted that the backpropagation algorithm can be easily generalized to a network with more hidden layers. For simplicity's sake, however, only one hidden layer is given in the exemplary network.

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \mathrm{lr} \times \frac{\partial E}{\partial \mathbf{W}^{(l)}}, \tag{2.14}$$

$$b^{(l)} \leftarrow b^{(l)} - \mathrm{lr} \times \frac{\partial E}{\partial b^{(l)}}. \tag{2.15}$$

The stochastic gradient descent (Bottou, 2010) method is widely applied for optimizing the loss of a network. Instead of computing the gradients exactly using a whole dataset, the network is shown a few examples of input data for computing the outputs and errors. The average gradient

for those examples is used to update the weights and biases accordingly. It turns out that after a certain number of training iterations, the network can quickly find a good set of parameters (LeCun et al., 2015). Thanks to the stochastic procedure, the gradients can be calculated on the fly in each iteration with a small number of examples, thereby significantly reducing the burden of computational cost.

2.1.2 Convolutional Neural Network

A convolutional neural network (CNN) is a type of neural network most commonly used for processing data in the form of multiple arrays with a grid pattern, particularly images (LeCun et al., 2015; Yamashita et al., 2018). The architecture of a CNN is designed in a way which enables the network to automatically learn representations from low levels (e. g., edges and shapes of objects) to high levels (e. g., object classes) of spatial hierarchies. A CNN typically consists of one or several blocks that contain convolutional and pooling layers, as well as a fully connected layer in the last part of the network. Fig. 2.2 shows a simple CNN.

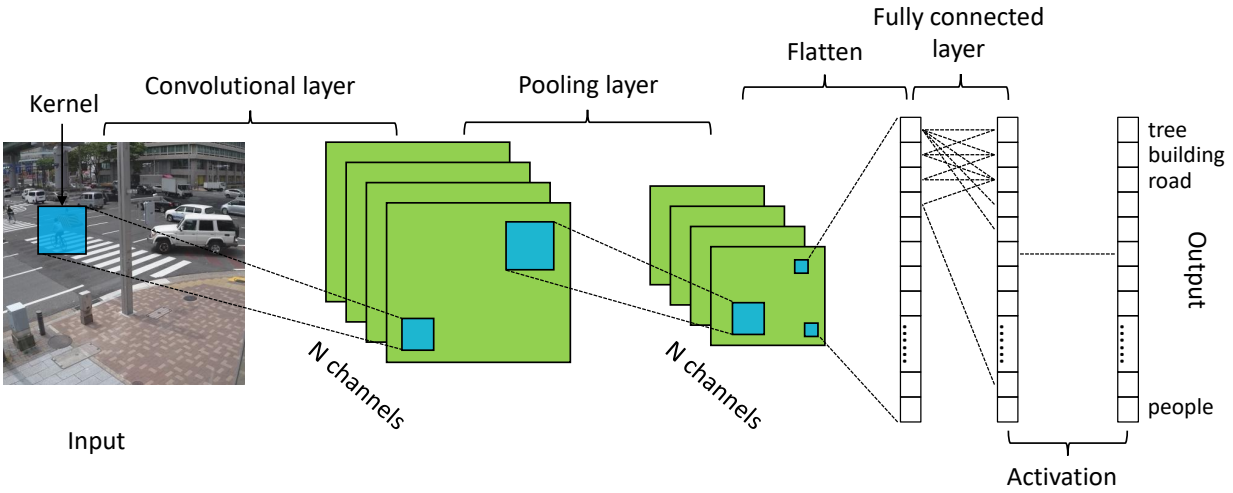


Figure 2.2: Architecture of a convolutional neural network. It mainly contains a block of a convolutional layer, a pooling layer, and a fully connected layer.

A convolutional layer uses the so-called *kernels* to extract local features. A kernel (also called *filter*) is a small array (width×height×channel) of real values (weights). In the forward pass, each kernel slides across the width and height of the input array and conducts the element-wise multiplication between the kernel and the input at any position, as denoted by the blue rectangles in Fig. 2.2. Then the sum of the products is passed through a non-linear activation function. The tensor that stores the output values in the corresponding positions is called a *feature map*. In the convolutional step, the shared weights of a small kernel that slides across the whole input turn the convolution into a fast and computationally efficient process.

A pooling layer adds a downsampling operation after a convolutional layer. It reduces the spatial size of the representation in each layer so as to reduce the number of parameters, variance of small shifts and distortions of the local features. The most commonly used pooling functions are the max pooling, average pooling, and global average pooling functions. The max pooling function outputs the maximum value of each patch of a feature map and discards the other values. In a similar way, the average pooling function calculates the average value of each patch of a feature map. The global average pooling function, however, is different from the max and average pooling functions that compute the values of patches of a feature map locally. Instead, the global average pooling

function outputs the average value of each feature map, which is used to partially or completely replace the fully connected layer when the number of feature maps is equal to the dimensionality of the output vector.

In the last part of a CNN, e. g., the output of the last convolutional and pooling layers (as shown in Fig. 2.2), is reshaped into a flattened feature vector. Then a fully connected layer, also called *dense layer*, is added to connect every entry of the flattened feature vector. In the end, an activation function is used to convert the output into a desirable range, e. g., the probabilities for a classification task. Similar to a feed-forward network, the backpropagation algorithm is employed to compute the gradients for adjusting the weights[1] of the network in order to optimize the loss between ground truth and prediction.

2.1.3 Recurrent Neural Network

A recurrent neural network (RNN) is another type of neural network. In contrast to CNN, however, it is most useful when working with time-dependent data, such as natural languages and sound. Feed-forward networks and CNNs assume that values of inputs (or outputs) are independent from each other and there is no time dependency from one value after another. RNNs can be distinguished from these networks by taking a closer look at the way that the "memory" of RNNs propagates the information step by step from the input sequence to the output sequence, thereby enabling the networks to learn a representation of time dependency. Fig. 2.3 shows a simple RNN architecture with one hidden layer. The output h_t of the hidden layer not only depends on the current input x_t, but also on the information from the previous time step.

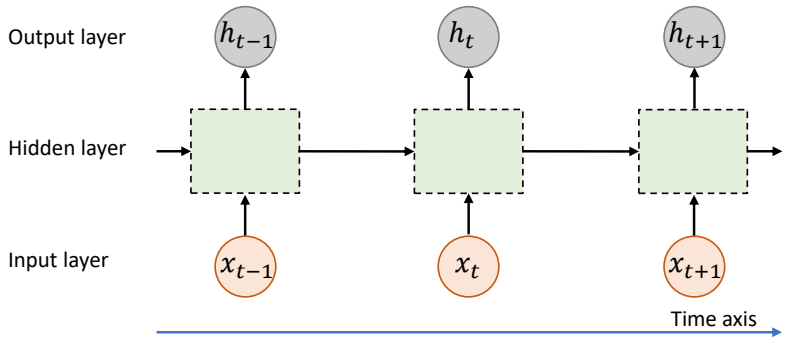


Figure 2.3: Architecture of a recurrent neural network.

The backpropagation algorithm is employed to train RNNs by adjusting the weights of the networks as well. Due to the time dependency, in an RNN, weights at each time step are summed in the backpropagation from the output layer to the input layer. This is a bottle neck for training RNNs with long time intervals using the gradient descent method (Bengio et al., 1994). If the gradient at each step is too small or too large, the summed gradients in long sequences will vanish or explode due to the chain rule with the operation of multiplication over time. The backpropagation algorithm cannot learn from vanished gradients. An RNN will turn out to be unstable if the weights of the network become too large, e. g., a small change of input will lead to a very big change of output.

Long Short-Term Memory (LSTM) proposed by Hochreiter and Schmidhuber (1997) is one of the many architectures to remedy the so-called "gradient vanishing and exploding problem", and

[1] For simplicity's sake, the bias is not mentioned in the networks.

to facilitate the training process using the gradient descent method. Instead of constructing a single layer, LSTM constructs four layers that interact with each other through dedicated gates, as denoted by the green blocks in Fig. 2.4. Gates are used to control the amount of information to be passed through the time axis. The key idea of LSTM is the memory cells that store information at each step. A memory cell c_{t-1} at step $t-1$ interacts with the output h_{t-1} and the new input x_t through different gates, generating the updated memory cell c_t, which—in turn—is then used to generate the new output h_t for the next step. In comparison to a conventional RNN, memory cells are found to be better at finding and exploiting long-range time dependencies in sequential data (Graves, 2013).

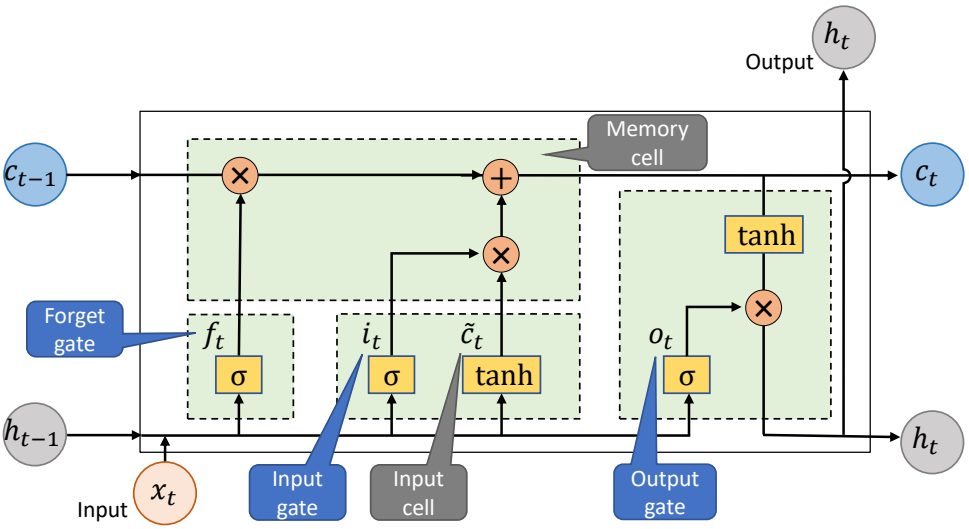


Figure 2.4: Architecture of Long Short-Term Memory.

The following equations demonstrate the process of updating of the memory cell and the output this creates at each step.

$$f_t = \sigma(\mathbf{W}_f x_t + \mathbf{U}_f h_{t-1} + b_f), \tag{2.16}$$
$$i_t = \sigma(\mathbf{W}_i x_t + \mathbf{U}_i h_{t-1} + b_i), \tag{2.17}$$
$$o_t = \sigma(\mathbf{W}_o x_t + \mathbf{U}_o h_{t-1} + b_o), \tag{2.18}$$
$$\tilde{c}_t = \tanh(\mathbf{W}_c x_t + \mathbf{U}_c h_{t-1}), \tag{2.19}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t, \tag{2.20}$$
$$h_t = o_t \circ \tanh(c_t), \tag{2.21}$$

where f stands for the forget gate, i for the input gate, and o for the output gate. $\tilde{c}$ is the current input cell before the input gate and c is the memory cell. h denotes the output of the hidden layer. $\mathbf{W}$ and $\mathbf{U}$ are trainable weights, and b is trainable bias. The subscript of each variable denotes the time step and $\circ$ stands for the element-wise product. σ denotes the Sigmoid activation function and tanh the Hyperbolic tangent activation function. All of the above gates are non-linear transformation of the input and output using the Sigmoid activation function. The forget gate f, defined by Eq. (2.16), determines how much the memory cell c_{t-1} at the previous step is retained to the memory cell c_t at the current step. The input gate i, defined by Eq. (2.17), determines how much of the input x_t of the network is saved to the current memory cell c_t. The output gate o, defined by Eq. (2.18), determines how much of the memory cell c_t is used to produce the output h_t of the hidden layer.

It should be noted that Gated Recurrent Unit (GRU) (Cho et al., 2014) is another widely used architecture of RNNs. However, GRU is not adopted, and thereby the comparison of LSTM and GRU is not presented in this thesis.

2.2 Approaches for Object Detection and Classification

In this section, two deep learning approaches for object detection and classification are introduced. They are both applied in the study of interaction detection for extracting object information from video frames.

2.2.1 You Only Look Once

YOLO (Redmon et al., 2016) has been one of the most widely used deep learning object detectors in recent years. It reframes object detection as a single regression problem of predicting the bounding box coordinates (center x, y, width and height) as well as the probability of an object class from image pixels. The input image is divided into a $S \times S$ grid, and then a neural network uses the features from the entire image to detect objects. The name "YOLO" is derived from the prediction process used by the network: "you only look once (YOLO) at an image to predict what objects are present and where they are". Compared to other two-stage object detectors, such as R-CNN (Girshick et al., 2014), which first uses region proposals for detecting potential bounding boxes and then runs a classifier for each box, the one-stage detector YOLO is much faster and achieves a relatively good performance at real-time speed (45 frames per second). Eq. (2.22) denotes the detection process:

$$P(\text{Class}_i|\text{Object}) * P(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = P(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}. \tag{2.22}$$

The conditional probability represents the probability of an object which belongs to a certain class $P(\text{Class}_i|\text{Object})$ being detected within a given grid. In this case, $P(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$ is the confidence of the bounding box containing that object. If there is no object detected in the background, the confidence is set to zero. Otherwise, the confidence should—ideally—to be equal to the Intersection of Union (IOU) between the predicted bounding box and the ground truth, denoted as $\text{IOU}_{\text{pred}}^{\text{truth}}$.

The second version YOLOv2 (Redmon and Farhadi, 2017) was proposed to further enhance the performance of detection. The first version of YOLO struggles with localization (i. e., with fitting the predicted bounding box correctly to the ground truth object) and relatively low recall (i. e., objects often remain undetected). In order to maintain the computational speed while at the same time improving the detection performance, several strategies are implemented in the later version as follows:

- Batch normalization is added to all of the convolutional layers in YOLO to help the model converge. Batch normalization (Ioffe and Szegedy, 2015) is a commonly used technique that reduces internal covariate shift over batches, i. e., the change in the distributions of internal nodes of a deep network via training. This technique allows for a much higher learning rate and less careful initialization to accelerate the training process.
- A higher resolution classifier trained on large images is used for detection. In YOLOv2, the classification network is fine-tuned at the full 448×448 resolution, instead of 224×224.
- A convolutional prediction layer with anchor boxes is used for predicting bounding boxes with predefined box shapes. Instead of predicting the box coordinates directly, this strategy

is used to predict the offset and confidence for anchor boxes, which facilitates the training process.

- The clustering algorithm K-means is used to select the representative anchor boxes that cover most of the ground truth bounding boxes of the data.
- Multi-scale training is used to guide YOLO to learn objects of different sizes by randomly scaling the input images to different sizes.
- The feature extraction backbone network Darknet-19 that has 19 covolutional layers and 5 max pooling layers is designed as the base for detection with faster speed, compared to the previous YOLO.
- In addition, a variant of YOLOv2, named YOLO9000, is jointly trained for detection and classification on different datasets, enabling the model to detect more classes of objects than the previous version.

In the third version YOLOv3 (Redmon and Farhadi, 2018), the performance of object detection has been further improved. The most important change in YOLOv3 compared to YOLOv2 is the base model Darknet. The new Darknet (Darknet-53) has 53 convolutional layers that make the model more accurate for object classification and detection. In this thesis, YOLOv3 is applied for detecting different road users, e. g., pedestrians, cyclists, motorbikes, cars, trucks, and buses.

2.2.2 Multi-Level Feature Pyramid Network

Another deep learning object detector adopted in this thesis is called M2Det. It is a Multi-Level Feature Pyramid Network (MLFPN) for detecting objects of different scales (Zhao et al., 2019a). In practice, different objects may appear in similar size, while—conversely—objects of the same class may appear in different sizes, a fact which proves problematic for many object detectors including the early version of YOLO. To address this challenge, MLFPN was proposed to extract features in different representation hierarchies from the input images after the backbone network and predicts the bounding box and category score for the detected objects. Similar to YOLO, M2Det is also an one-stage object detector. On the other hand, M2Det uses an established image classification backbone, VGG (Simonyan and Zisserman, 2014b) or ResNet-101 (He et al., 2016) for processing the initial input images, whereas YOLO uses the dedicated classification model Darknet for this purpose.

MLFPN consists of three modules, namely, the Feature Fusion Module (FFM), the Thinned U-shape Module (TUM) and the Scale-wise Feature Aggregation Module (SFAM), as shown in Fig. 2.5.

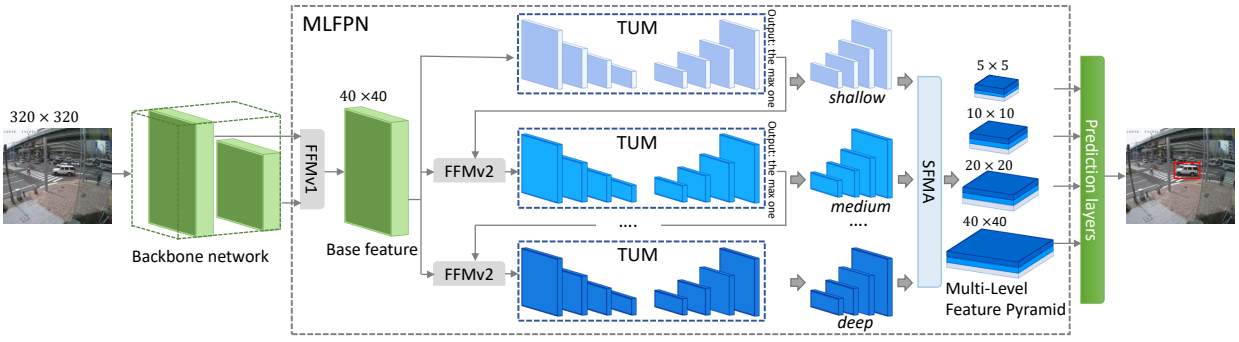


Figure 2.5: The framework of M2Det. The figure is adapted from (Zhao et al., 2019a).

There are two Feature Fusion Modules that fuse features from different levels. FFMv1 fuses the different levels of feature maps from the backbone network and enriches the Base feature with semantic information. FFMv2 takes the Base feature and the largest output feature map from the previous TUM as input and generates the fused feature for the next TUM. TUM, in turn, adopts a thinned U-shape encoder–decoder structure and generates a group of multi-scale features. SFAM aggregates these multi-level multi-scale features (e. g., shallow, medium and deep) into multi-level feature pyramids, which are used to predict the bounding box and category score for the detected objects. Given its state-of-the-art performance and comparatively high processing speed, it is also applied in this thesis for detecting different road users.

2.3 Optical Flow

Horn and Schunck (1981) defined optical flow as the distribution of apparent velocities of movement of brightness patterns in an image. It is widely used in e. g., motion based segmentation and video stabilization. With the assumptions that the brightness at a point in the image does not change and the brightness of the neighboring points changes smoothly, optical flow can be calculated from one frame to the consecutive frame.

Given the assumption of constant brightness, the brightness of a point (x, y) does not change after a short time. This is denoted by the equation

$$f(x, y, t) = f(x + dx, y + dy, t + dt), \tag{2.23}$$

where f is the brightness function. The above equation can be written as the Taylor Series

$$\begin{aligned} f(x, y, t) &= f(x, y, t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt, \\ \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt &= 0, \\ f_x u + f_y v + f_t &= 0, \end{aligned} \tag{2.24}$$

where u and v are the optical flow (velocities) in the x- and y-axis on a 2D plane.

In this thesis, the Gunnar-Farnebäck method is applied to compute optical flow, which is also called *dense optical flow*. Farnebäck (2003) proposed the two-frame motion estimation based on polynomial expansion to approximate the neighborhood of each pixel with a polynomial. The local signal model in quadratic polynomials is expressed as

$$\begin{aligned} f(x, y) &\sim r_1 + r_2 x + r_3 y + r_4 x^2 + r_5 y^2 + r_6 xy, \\ &= \begin{pmatrix} x & y \end{pmatrix}^{\mathbf{T}} \begin{pmatrix} r_4 & r_6/2 \\ r_6/2 & r_5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} r_2 \\ r_3 \end{pmatrix}^{\mathbf{T}} \begin{pmatrix} x \\ y \end{pmatrix} + r_1, \\ &= \mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathbf{T}}\mathbf{x} + c, \end{aligned} \tag{2.25}$$

where r_i denotes the polynomial coefficients, $\mathbf{x}$ stands for a 2D vector in a local coordinate system, $\mathbf{A}$ is a symmetric matrix, $\mathbf{b}$ a vector and c a scalar. $\mathbf{T}$ stands for the transpose operation. A weighted leastsquares fit is applied to estimate the coefficients in the neighborhood. There are two components carried by the weights: certainty and applicability. According to the assumption of smoothness, the certainty is set in such a way that the close neighborhood points have more impact on the coefficient estimation than the outside points. In accordance with the position, the applicability determines the relative weight of the neighborhood points.

Displacement Estimation is used to analyze how a polynomial undergoes an ideal translation. A quadratic polynomial for the first image is denoted as

$$f_1(x, y) = \mathbf{x}^\mathbf{T}\mathbf{A}_1\mathbf{x} + \mathbf{b}_1^\mathbf{T}\mathbf{x} + c_1. \tag{2.26}$$

After a global displacement of $\mathbf{d}$, the new signal's quadratic polynomial for the second image is denoted as

$$\begin{aligned} f_2(x, y) &= f_1(\mathbf{x} - \mathbf{d}), \\ &= (\mathbf{x} - \mathbf{d})^\mathbf{T}\mathbf{A}_1(\mathbf{x} - \mathbf{d}) + \mathbf{b}_1^\mathbf{T}(\mathbf{x} - \mathbf{d}) + c_1, \\ &= \mathbf{x}^\mathbf{T}\mathbf{A}_1\mathbf{x} + (\mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d})^\mathbf{T}\mathbf{x} + \mathbf{d}^\mathbf{T}\mathbf{A}_1\mathbf{d} - \mathbf{b}_1^\mathbf{T}\mathbf{d} + c_1, \\ &= \mathbf{x}^\mathbf{T}\mathbf{A}_2\mathbf{x} + \mathbf{b}_2^\mathbf{T}\mathbf{x} + c_2, \end{aligned} \tag{2.27}$$

from which one can derive that

$$\begin{aligned} \mathbf{A}_2 &= \mathbf{A}_1, \\ \mathbf{b}_2 &= \mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d}, \\ c_2 &= \mathbf{d}^\mathbf{T}\mathbf{A}_1\mathbf{d} - \mathbf{b}_1^\mathbf{T}\mathbf{d} + c_1. \end{aligned}$$

If $\mathbf{A}_1$ is non-singular, the translation $\mathbf{d}$ can be solved as

$$\mathbf{d} = -\frac{1}{2}\mathbf{A}_1^{-1}(\mathbf{b}_2 - \mathbf{b}_1). \tag{2.28}$$

Due to e. g., signal noises, it is not realistic to calculate the global translation based on two signals, where $\mathbf{A}_1 = \mathbf{A}_2$. To remedy this in a more practical way, the global polynomial in Eq. (2.26) is substituted by

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\mathbf{x})}{2}, \tag{2.29}$$

and

$$\Delta\mathbf{b}(\mathbf{x}) = -\frac{1}{2}(\mathbf{b}_2 - \mathbf{b}_1). \tag{2.30}$$

By plugging Eq. (2.30) into Eq. (2.28), the following constraint can be derived:

$$\mathbf{A}(\mathbf{x})\mathbf{d}(\mathbf{x}) = \Delta\mathbf{b}(\mathbf{x}), \tag{2.31}$$

$$\mathbf{d} = (\mathbf{A}^\mathbf{T}\mathbf{A})^{-1}(\mathbf{A}^\mathbf{T}\Delta\mathbf{b}). \tag{2.32}$$

The error of the estimation can be calculated as

$$e(\mathbf{x}) = \|\mathbf{A}\mathbf{d} - \Delta\mathbf{b}\|^2. \tag{2.33}$$

The above equations solve the point-wise displacement. However, the point-wise displacement often turns out to be noisy due to the varying errors. Farnebäck (2003) made the assumption that the displacement field is only slowly varying, which is in line with the smoothness assumption made in (Horn and Schunck, 1981). Therefore, the information over a neighborhood of each pixel

is integrated, i. e., finding $\mathbf{d}(\mathbf{x})$ that satisfies Eq. (2.31) as much as possible over the neighborhood of $\mathbf{x}$, denoted as I. Then the error of the estimation is depending on the displacement field.

$$e(\mathbf{x}) = \sum_{\Delta\mathbf{x}\in I} w(\Delta\mathbf{x}) \|\mathbf{A}(\mathbf{x}+\Delta\mathbf{x})\mathbf{d} - \Delta\mathbf{b}(\mathbf{x}+\Delta\mathbf{x})\|^2, \tag{2.34}$$

where $w(\Delta\mathbf{x})$ is a function to weigh the impact of the points in the neighborhood I of $\mathbf{x}$, and the error is the sum of the weighted errors over the neighborhood. The minimum error is solved by the weighted leastsquares fit.

In this thesis, the dense optical flow algorithm is used to compute the motion of dynamic objects over consecutive video frames.

2.4 Spatial Clustering

Density-Based Spatial Clustering of Applications with Noise (DBSCAN), is a prominent approach to clustering spatial objects (points). Other approaches are partitioning algorithms (e. g., k-means and k-medoids) and hierarchical algorithms. As its name states, DBSCAN clusters spatial data points based on density (Ester et al., 1996). Within each cluster, the density of points is considerably higher than the density outside the cluster. In other words, the density within the areas of noise (non-clustered points) is lower than the density in any of the clusters. Mathematically, the density in the neighborhood of a given radius of a cluster has to exceed a threshold MinPts, containing a minimum number of points. A distance function $\mathrm{dist}(p,q)$ measures the distance between the points p and q and determines the shape of the neighborhood. Commonly used distance functions are, e. g., Manhattan distance and Euclidean distance. The definition of the ϵ-neighborhood of the point p is denoted by $\mathsf{N}_\epsilon(p)$ as

$$\mathsf{N}_\epsilon(p) = \{q \in D | \mathrm{dist}(p,q) \leq \epsilon\}, \tag{2.35}$$

where ϵ stands for the threshold of the maximum distance, determining whether the given point q from the dataset D belongs to the neighborhood of p.

Two kinds of points, *border* and *core* points, are characterized differently according to their reachability and connectivity. An ϵ-neighborhood of a border point contains significantly fewer points than an ϵ-neighborhood of points inside the cluster. The following definitions specify the reachability and connectivity of a point.

(1) If every point q is inside the ϵ-neighborhood of p and $\mathsf{N}_\epsilon(p)$ contains more than MinPts points, then q is *directly density-reachable* from p, and p satisfies the core point condition.

$$\begin{aligned} &q \in \mathsf{N}_\epsilon(p) \text{ and,} \\ &|\mathsf{N}_\epsilon(p)| \geq \text{MinPts} \quad \text{(core point condition).} \end{aligned} \tag{2.36}$$

(2) If point q is connected through a chain of points $\{q_1, \cdots, q_n\}$, such that $q_1 = p$, $q_n = q$, and q_{i+1} is directly density-reachable from q_i, then q is *density-reachable* from p both w. r. t. ϵ and MinPts.

(3) If there is a point o such that both q and p are density-reachable from o w. r. t. ϵ and MinPts, then q and p are *density-connected.*

Based on the above definitions of reachability and connectivity, a cluster is defined as a set of density-connected points. The cluster is maximal w. r. t. density-reachability. Points in the dataset D that do not belong to any cluster are called noise.

The DBSCAN algorithm uses a recursive two-step process to find all the clusters:

- For step one, it starts with an arbitrary point p from the dataset D and checks if p satisfies the core point condition regarding the given parameters ϵ and MinPts. Otherwise, if p is a border point and no points prove to be density-reachable from p, the DBSCAN algorithm moves on to the next point of the dataset.
- For step two, if p is a core point, the DBSCAN algorithm retrieves all points that are density-reachable from p. This procedure yields a cluster w. r. t. ϵ and MinPts.

Compared to other partitioning or hierarchical clustering algorithms, DBSCAN requires no prior knowledge of the number of clusters or a cutoff of the hierarchy, making it more efficient when dealing with clusters with arbitrary shapes. This is essential for automatically clustering road users according to group behavior, when the size of each group is not known as a prior. Therefore, DBSCAN is used in this thesis for pedestrian group detection.

2.5 Transformer Encoder with Self-Attention

The Transformer network has been proposed by Vaswani et al. (2017) for natural language processing and sequence modeling. One commonly used structure for mapping an input sequence to an output sequence is an encoder–decoder structure. First, the encoder maps the input sequence $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ to a sequence of representations $(\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n)$, then the decoder takes over, generating an element at a step for the output sequence $(\boldsymbol{y}_1, \cdots, \boldsymbol{y}_m)$ using the given representations. For example, in an RNN, this process is done one step after another. However, the time dependency leads to increasing difficulty for information propagation, especially over a long sequence. This recurrent architecture also causes problems for optimization via stochastic gradient descent (LeCun et al., 1989), the so-called gradient vanishing and exploding problem already mentioned in Sec. 2.1.3. Even though there are many sophisticated approaches, such as LSTM and GRU to remedy this problem, none of them changes the underlying recurrent structure. The Transformer network uses a totally different architecture to address this problem. It eschews recurrence and relies entirely on a self-attention mechanism to draw global dependencies between input and output. Parallel computing is another advantage of the Transformer network compared to RNNs, since the time dependency is no longer computed one step after another.

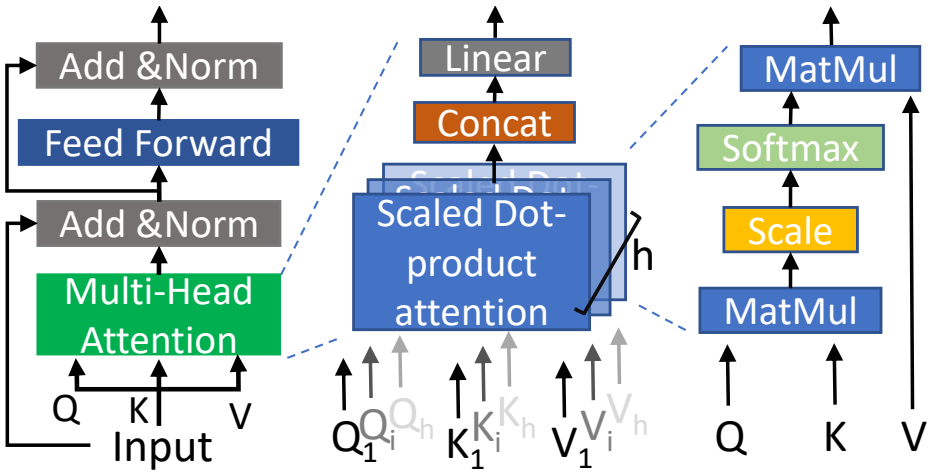


Figure 2.6: The Transformer encoder with a self-attention mechanism. Add stands for adding an identity mapping using a shortcut connection (He et al., 2016), Norm for layer normalization (Ba et al., 2016), Feed Forward for a feed-forward network, Concat for concatenation, and MatMul for matrix multiplication.

The self-attention mechanism is analogous to retrieving desirable data from a database by a query that matches the keys connected to the corresponding values. In other words, the self-attention mechanism can be described as mapping a query and a set of key–value pairs to an output, with the query (Q), keys (K) and values (V) all being vectors derived from the input[2] itself through linear transformation, as denoted in Fig. 2.6. First, Q, K and V are separately obtained by three linear transformations with the same input:

$$\begin{aligned} Q &= \mathbf{X}W_Q, \\ K &= \mathbf{X}W_K, \\ V &= \mathbf{X}W_V, \end{aligned} \tag{2.37}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d_{\mathbf{X}} \times d_k}$ are the trainable parameters and $d_{\mathbf{X}}$ is the dimensionality of the input $\mathbf{X}$. Then the weight assigned to each value is calculated as the dot-product of the query with the corresponding key:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^{\mathbf{T}}}{\sqrt{d_k}}\right)V, \tag{2.38}$$

where $\sqrt{d_k}$ is the scaling factor, d_k is the dimensionality of the vector K and $\mathbf{T}$ is the transpose operation. This operation is also called *scaled dot-product attention* (Vaswani et al., 2017).

Unlike LSTM that takes the input in the order of a sequence by which the temporal information is retained explicitly, the self-attention mechanism takes the complete input at once. Hence, the global dependencies between the input and output are not restricted by the element's position in the sequence and the length of the sequence. On the other hand, since there is no recurrence, the order of the sequence is no longer obtained. But this information is often critical for sequence modeling, such as a specific word is usually tightly connected to the words before it in a sentence. To this end, position encodings are added to the Q, K and V vectors at the bottom of each self-attention layer. The sine and cosine functions of different frequencies (varying in time here) are the most widely used position encodings:

$$\mathbf{p}^t = \{p_{t,d}\}_{d=1}^{D},\ p_{t,d} = \begin{cases} \sin\left(\frac{t}{10000^{d/D}}\right), & \text{for } d \text{ even;} \\ \cos\left(\frac{t}{10000^{d/D}}\right), & \text{for } d \text{ odd,} \end{cases} \tag{2.39}$$

where $D = d_k$ ensures the position encodings to have the same dimensionality as Q, K and V.

Besides, *multi-head attention* (Vaswani et al., 2017) is proposed to improve the expressiveness of the self-attention mechanism. One head is an independent dot-product attention module and attends to a representation sub-space. As denoted in Fig. 2.6, multiple heads are jointly learned in parallel for different representations. The following equation denotes the multi-head attention (Vaswani et al., 2017) strategy:

$$\begin{aligned} \text{MultiHead}(Q,K,V) &= \text{ConCat}(\text{head}_1, \cdots, \text{head}_h)W_O, \\ \text{head}_i &= \text{Attention}(QW_{Qi}, KW_{Ki}, VW_{Vi}), \end{aligned} \tag{2.40}$$

where W_{Qi}, W_{Ki}, $W_{Vi} \in \mathbb{R}^{D \times d_{ki}}$ are the same linear transformation parameters as in Eq. (2.37) and W_O are the linear transformation parameters for aggregating the extracted information from

[2]Since in this thesis, only the self-attention mechanism in the Transformer encoder is adopted, the structure of the Transformer decoder is not presented in more detail than this.

different heads. With regard to this equation, it should be noted that $d_{ki} = \frac{d_k}{h}$ must be an aliquot part of d_k. h is the total number of attention heads.

In addition to the multi-head attention, the Transformer encoder also adopts residual learning (He et al., 2016), layer normalization (Ba et al., 2016), and a feed-forward network to ease the training of the encoder network. Residual learning is carried out by adding an identity mapping of the input using a shortcut connection to the network (as shown in Fig. 2.6), so the multi-head attention only learns the residuals. This makes training the network more efficient than directly learning a mapping from the input (He et al., 2016). Layer normalization is a technique to fix the mean and variance of the summed inputs within each layer while improving the training speed (Ba et al., 2016).

2.6 Conditional Generative Model

The problem of interaction detection and trajectory prediction can be formulated as building a desirable function f taking as input the observed data $\mathbf{X}$, and generating the anticipated results $\mathbf{Y}$, as denoted by Eq. (2.41).

$$\mathbf{Y} = f(\mathbf{X}). \tag{2.41}$$

Given the stochastic characteristics of human behavior, the output $\mathbf{Y}$ is often multimodal. Hence, the function should be able to map one single input to many possible outputs, i. e., to perform probabilistic inference and make diverse predictions.

The one-to-many problem is defined as modeling the distribution of a high-dimensional output space as a deep generative model conditioned on input observation. One solution for this problem is the Conditional Variational Auto-Encoder (CVAE) model (Sohn et al., 2015). It is a conditional graphical model whose input observations modulate the prior on Gaussian latent variables $\mathbf{z}$ that generate the outputs (Sohn et al., 2015), as denoted by Eq. (2.42).

$$p(\mathbf{Y}|\mathbf{X}) = \mathcal{N}(f(\mathbf{z}, \mathbf{X}),\, \sigma^2 * \mathbf{I}). \tag{2.42}$$

The conditional probability of the output is an isotropic Gaussian distribution. The mean $\mu = f(\mathbf{z}, \mathbf{X})$ is a function of the input $\mathbf{X}$ and the latent variables $\mathbf{z}$, and the covariance matrix is an identity matrix $\mathbf{I}$ multiplied by some scalar σ^2.

CVAE is built upon variational inference and learning of directed graphical models (Kingma and Welling, 2014; Rezende et al., 2014; Kingma et al., 2014), as an extension of the Variational Auto-Encoder (VAE) (Kingma and Welling, 2014). In order to solve Eq. (2.42), it is necessary to revisit the variational inference and VAE to understand the solution.

2.6.1 Variational Auto-Encoder

The Variational Auto-Encoder (VAE) assumes that the dataset $\mathbf{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N\}$ with $N \in \mathbb{N}_0$, contains i.i.d. samples[3] of some continuous or discrete variable $\boldsymbol{X}$. The dataset can be generated from unobserved continuous random variables $\mathbf{z}$, the so-called *latent variables*. Eq. (2.43) denotes the integral of the marginal likelihood, where $p_\theta(\mathbf{z})$ is the prior distribution of the latent variables and θ are the generative parameters.

$$p_\theta(\boldsymbol{X}) = \int p_\theta\left(\boldsymbol{X}|\mathbf{z}\right) p_\theta(\mathbf{z}) d\mathbf{z}. \tag{2.43}$$

[3]The abbreviation "i.i.d." refers to independent and identically distributed random variables.

However, the equation cannot be solved analytically due to the intractable posterior $p_\theta(\mathbf{z}|\boldsymbol{X}) = p_\theta(\boldsymbol{X}|\mathbf{z})p_\theta(\mathbf{z})/p_\theta(\boldsymbol{X})$, nor can it be solved efficiently due to the expensive sampling over a large dataset. To solve the problem, a variational approximation of the true posterior is introduced as the recognition model $q_\phi(\mathbf{z}|\mathbf{X})$, where ϕ are the variational parameters. Then, Eq. (2.43) can be rewritten as:

$$\log p_\theta(\boldsymbol{X}_1, \cdots, \boldsymbol{X}_N) = \sum_{i=1}^{N} \log p_\theta(\boldsymbol{X}_i), \tag{2.44}$$

$$\text{with } \log p_\theta(\boldsymbol{X}_i) = D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}_i)||p_\theta(\mathbf{z}|\boldsymbol{X}_i)) + \mathcal{L}(\theta, \phi; \boldsymbol{X}_i). \tag{2.45}$$

The sum over the marginal likelihoods of individual data points is denoted by Eq. (2.44) and (2.45). Here, $D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X})||p_\theta(\mathbf{z}|\boldsymbol{X}))$ is the Kullback-Leibler divergence that measures the error of the approximation to the true posterior. The Kullback-Leibler divergence is always non-negative. In an ideal solution, the approximation can map the true posterior correctly when the Kullback-Leibler divergence reaches zero. In practice,

$$\log p_\theta(\boldsymbol{X}_i) \geq \mathcal{L}(\theta, \phi; \boldsymbol{X}_i), \tag{2.46}$$

where $\mathcal{L}$ is called the (variational) *lower bound* on the marginal likelihood of the data point i.

With the variational approximation, Bayes' theorem is applied to solve $\mathcal{L}(\theta, \phi;\ \boldsymbol{X}_i)$ as follows:

$$\begin{aligned}
\mathcal{L}(\theta,\phi; \boldsymbol{X_i}) &= \log \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)} \frac{p_\theta(\boldsymbol{X}_i, \mathbf{z})}{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}, \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}[-\log q_\phi(\mathbf{z}|\boldsymbol{X}_i) + \log p_\theta(\boldsymbol{X}_i, \mathbf{z})], \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}[-\log q_\phi(\mathbf{z}|\boldsymbol{X}_i) + \log p_\theta(\boldsymbol{X}_i|\mathbf{z}) + \log p_\theta(\mathbf{z})], \\
&= -\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}[\log \frac{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}{p_\theta(\mathbf{z})}] + \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}[\log p_\theta(\boldsymbol{X}_i|\mathbf{z})], \\
&= -D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}_i)||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}[\log p_\theta(\boldsymbol{X}_i|\mathbf{z})],
\end{aligned} \tag{2.47}$$

where $-D_{KL}(\,\cdot\,)$ is the negative Kullback-Leibler divergence of the approximate posterior from the prior $p_\theta(\mathbf{z})$ and acts as a regularizer. $\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i)}(\,\cdot\,)$ is an expected negative reconstruction loss. When one optimizes the log likelihood on the left side of Eq. (2.46), the Kullback-Leibler divergence and reconstruction loss are jointly minimized. Hence, the recognition model parameters ϕ and the generative parameters θ can be learned jointly.

The structure of an "Auto-Encoder" becomes intuitive when viewed as in Eq. (2.47): the recognition model $q_\phi(\mathbf{z}|\boldsymbol{X_i})$ *encodes* the input into the latent variables and the generative model $\log p_\theta(\boldsymbol{X_i}|\mathbf{z})$ *decodes* the output from the latent variables. Fig. 2.7 illustrates the graphical structure of the VAE model.

An analytical solution for the Kullback-Leibler divergence in Eq. (2.47) of two distributions can be found in the Gaussian case (Kingma and Welling, 2014). The posterior $p_\theta(\mathbf{z}|\boldsymbol{X})$ is assumed to be a normal distribution $\mathcal{N}(\mu, \sigma^2\mathbf{I})$, and $\mathbf{I}$ is the identity matrix. It should be noted that for $p_\theta(\mathbf{z}|\boldsymbol{X})$ other distributions, such as uniform distribution, can also be assumed. The normal distribution, however, is the most widely used one and its effectiveness has been empirically proven in many studies. Then, the prior $p_\theta(\mathbf{z})$ can be derived as:

$$p_\theta(\mathbf{z}) = \sum_{\boldsymbol{X}} p_\theta(\mathbf{z}|\boldsymbol{X})p(\boldsymbol{X}) = \sum_{\boldsymbol{X}} \mathcal{N}(0, \mathbf{I})p(\boldsymbol{X}) = \mathcal{N}(0, \mathbf{I}), \tag{2.48}$$

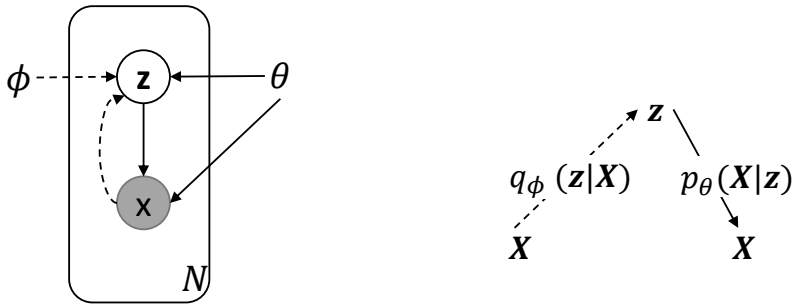


Figure 2.7: Variational Auto-Encoder graphical model. The generative model $p_\theta(\boldsymbol{X}|\mathbf{z})p_\theta(\mathbf{z})$ is denoted by the solid lines, where $p_\theta(\mathbf{z})$ is the prior of the latent variables. The variational approximation $q_\phi(\mathbf{z}|\boldsymbol{X})$ to the intractable posterior of $p_\theta(\mathbf{z}|\boldsymbol{X})$ is denoted by the dashed lines. The diagrams are adapted from (Kingma and Welling, 2014).

i. e., the prior $p_\theta(\mathbf{z})$ is also a normal distribution. Therefore, the Kullback-Leibler divergence term in Eq. (2.47) in the dimensionality of J can be rewritten as:

$$\begin{aligned}
-D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X_i})||p_\theta(\mathbf{z})) &= \int q_\theta(\mathbf{z})(\log p_\theta(\mathbf{z}) - \log q_\theta(\mathbf{z}))d\mathbf{z}, \\
&= \int \mathcal{N}(\mathbf{z};\mu,\boldsymbol{\sigma^2})\log\mathcal{N}(\mathbf{z};0,\mathbf{I})d\mathbf{z} - \int \mathcal{N}(\mathbf{z};\mu,\boldsymbol{\sigma^2})\log\mathcal{N}(\mathbf{z};\mu,\boldsymbol{\sigma^2})d\mathbf{z}, \\
&= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2+\sigma_j^2) + \frac{J}{2}\log(2\pi) + \frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2), \\
&= \frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2 - \mu_j^2 - \sigma_j^2).
\end{aligned} \tag{2.49}$$

The expected reconstruction error $\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X_i})}[\log p_\theta(\boldsymbol{X_i}|\mathbf{z})]$ in Eq. (2.47) requires sampling, such as Monte Carlo sampling.

$$\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X_i})}[\log p_\theta(\boldsymbol{X_i}|\mathbf{z})] \simeq \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\boldsymbol{X_i}|\mathbf{z}_i^l)), \tag{2.50}$$

where the sample $\mathbf{z}_i^l$ is the input of $\log p_\theta(\boldsymbol{X_i}|\mathbf{z}_i^l)$. It can be seen that the generative model $p_\theta(\boldsymbol{X}_i|\mathbf{z}_i^l)$ is the probability density of the data point $\boldsymbol{X_i}$ given the sample $\mathbf{z}_i^l$.

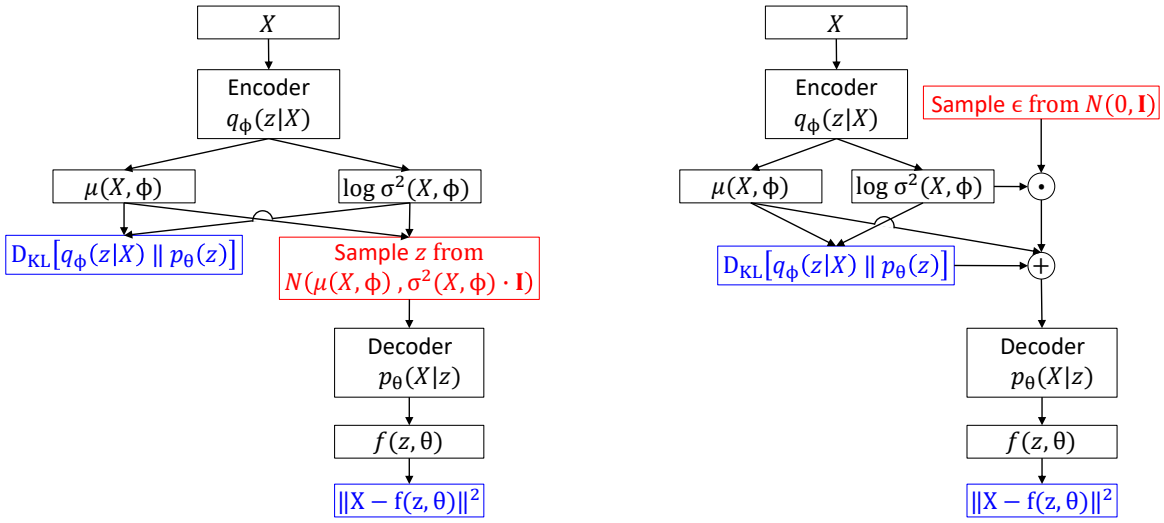
There is one remaining problem in the sampling process. Neural networks can be used to parameterize the mapping of θ and ϕ, which works in the forward pass of the VAE model. However, there is no gradient to the sampling when the neural networks have to be optimized via gradient descent in backpropagation. To solve this problem, a *re-parameterization* trick (Kingma and Welling, 2014; Rezende et al., 2014) is introduced to mimic the stochastic property of the latent variables while maintaining the gradients for backpropagation at the same time, as shown in Fig. 2.8.

$$\mathbf{z}_i^l = g_\phi(\epsilon_i^l, \boldsymbol{X}_i^l), \text{ where } \mathbf{z}_i^l \sim q_\phi(\mathbf{z}|\boldsymbol{X}_i^l), \tag{2.51}$$

ϵ is a noise vector drawn from the distribution of $\mathcal{N}(0, \mathbf{I})$. Assuming the posterior approximation $q_\phi(\mathbf{z}|\boldsymbol{X_i}) = \mathcal{N}(\mu, \sigma^2)$, a valid function of $g_\phi(\epsilon_i^l, \boldsymbol{X_i})$ can be formulated as:

$$\mathbf{z}_i^l = g_\phi(\epsilon_i^l, \boldsymbol{X_i^l}) = \mu_i^l + \sigma_i^l \odot \epsilon_i^l, \tag{2.52}$$

where $\odot$ is element-wise multiplication.



(a) The general forward pass of VAE without application of the re-parameterization trick

(b) The general forward pass of VAE with application of the re-parameterization trick, which enables back-propagation

Figure 2.8: Flowcharts of Variational Auto-Encoder. The blue boxes indicate model losses (i. e., Kullback-Leibler divergence loss and reconstruction loss) and the red boxes indicate sampling operations.

To summarize, deriving from the equations presented in this chapter, the lower bound of the estimation of VAE can be solved by

$$\mathcal{L}(\theta,\phi; \boldsymbol{X_i}) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2) + \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\boldsymbol{X_i}|\mathbf{z}_i^l)), \tag{2.53}$$

where $\mathbf{z}_i^l$ is sampled from Eq. (2.52).

2.6.2 Conditional Variational Auto-Encoder

The Conditional Variational Auto-Encoder (CVAE) model is proposed by Sohn et al. (2015) for structured output prediction. Differently from the VAE model that reconstructs the input variables with a variational recognition of the posterior, the CVAE model generates desirable outputs conditioned on the input variables. To be more specific, given the observation that $\boldsymbol{X}$, the latent variables $\mathbf{z}$ are drawn from $p_\theta(\mathbf{z}|\boldsymbol{X})$, and the output $\boldsymbol{Y}$ is generated from $p_\theta(\boldsymbol{Y}|\mathbf{z}, \boldsymbol{X})$. A latent variable z can be drawn multiple times from the distribution $p_\theta(\mathbf{z}|\boldsymbol{X})$. The multi-sampling process of the latent variables $\mathbf{z}$ allows for modeling multiple modes in the conditional distribution of the output variables $\boldsymbol{Y}$, the so called one-to-many mapping. In Eq. (2.54), the latent variables $\mathbf{z}$ can be made statistically independent of the input variables in a way that $p_\theta(\mathbf{z}|\mathbf{X}) = p_\theta(\mathbf{z})$ (Kingma et al., 2014), as shown in Eq. (2.48).

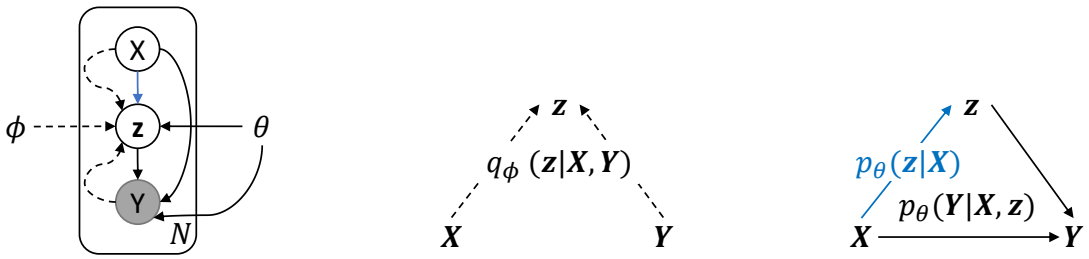


Figure 2.9: Conditional Variational Auto-Encoder graphical model. The generative model $p_\theta(\boldsymbol{Y}|\boldsymbol{X},\mathbf{z})\,p_\theta(\mathbf{z})$ is denoted by the solid black lines, with $p_\theta(\mathbf{z})$ being the prior of the latent variables and made independent from the input variables in such a way that $p_\theta(\mathbf{z}|\boldsymbol{X}) = p_\theta(\mathbf{z})$, as denoted by the solid blue lines. The variational approximation $q_\phi(\mathbf{z}|\boldsymbol{Y},\boldsymbol{X})$ to the intractable posterior of $p_\theta(\mathbf{z}|\boldsymbol{Y},\boldsymbol{X})$ is presented by the dashed lines. The diagrams are adapted from (Sohn et al., 2015).

$$p_\theta(\boldsymbol{Y}|\boldsymbol{X}) = \int p_\theta(\boldsymbol{Y}|\boldsymbol{X},\mathbf{z})\,p_\theta(\mathbf{z})d\mathbf{z}. \tag{2.54}$$

Eq. (2.54) denotes the integral of the conditional likelihood given the input $\boldsymbol{X}$. Similar to the VAE model, variational approximation estimation is used to solve Eq. (2.54) for the intractable posterior. At the data point i, the log likelihood is denoted by Eq. (2.55), where $q_\phi(\mathbf{z}|\boldsymbol{Y}_i,\boldsymbol{X}_i)$ is the variational approximation of the true posterior $p_\theta(\mathbf{z}|\boldsymbol{Y}_i,\boldsymbol{X}_i)$.

$$\log p_\theta(\boldsymbol{Y}_i|\boldsymbol{X}_i) = D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{Y}_i,\boldsymbol{X}_i)||p_\theta(\mathbf{z}|\boldsymbol{Y}_i,\boldsymbol{X}_i)) + \mathcal{L}(\theta,\phi;\boldsymbol{Y}_i,\boldsymbol{X}_i). \tag{2.55}$$

The lower bound of the log likelihood at the data point i can be solved by applying Bayes' theorem and sampling:

$$\begin{aligned}\log p_\theta(\boldsymbol{Y}_i|\boldsymbol{X}_i) &\geq \mathcal{L}(\theta,\phi;\boldsymbol{Y}_i,\boldsymbol{X}_i),\\ &= -D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}_i,\boldsymbol{Y}_i)||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i,\boldsymbol{Y}_i)}[\log p_\theta(\boldsymbol{Y}_i|\boldsymbol{X}_i,\mathbf{z})],\\ &\simeq -D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}_i,\boldsymbol{Y}_i)||p_\theta(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\boldsymbol{Y_i}|\boldsymbol{X_i},\mathbf{z}_i^l)).\end{aligned} \tag{2.56}$$

The complete derivation for Eq. (2.56) is analogous to the solution of the VAE. Similarly, the Kullback-Leibler divergence of two Gaussian distributions can be solved analytically, and the non-linear mapping of the generative parameters θ and the variational recognition parameters ϕ can be parameterized by neural networks and optimized via backpropagation with the re-parameterization trick described above.

3 Related Work

The discussion of the related studies focuses on two different aspects, interaction detection (Sec. 3.1) and trajectory prediction (Sec. 3.2), with respect to both conventional and modern approaches. The first section provides a brief review of the early work that largely relied on manual observations for safety-related analyses and the recent works on automated detection of road users' behavior at intersections. The second section first compares the conventional approaches with deep learning approaches for trajectory prediction, and then further discusses the state-of-the-art deep learning approaches.

3.1 Interaction Detection

3.1.1 Collisions, Conflicts, and Interactions

Interaction detection at intersections is a topic closely related to collision and conflict analysis in the domain of traffic safety assessment. According to Svensson and Hydén (2006), the interaction between road users can be described as "a continuum of safety-related events" denoted as a pyramid (Fig. 3.1) containing different types of events. In everyday traffic, collisions or accidents fortunately only account for a very small number of these events—the tip of the pyramid of interactions. More frequent events are conflicts with different degrees of severity (serious, slight, and potential) and undisturbed passages. In this thesis, a high level of classification is adopted to classify events into non-interactions (undisturbed passages) and interactions (all the other events), as shown Fig. 3.1.

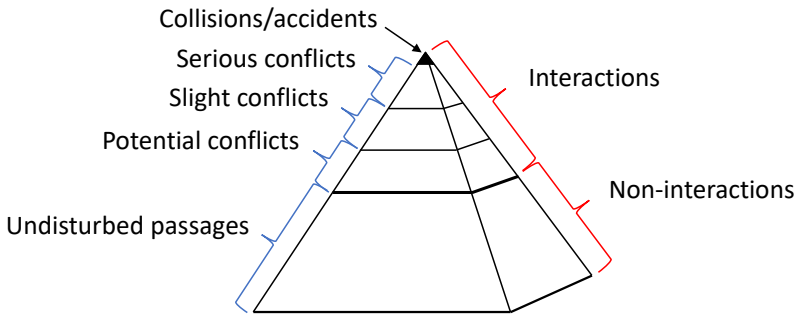


Figure 3.1: The pyramid of interactions. The figure is partially adapted from (Svensson and Hydén, 2006).

Early collision analyses focus on collision events that have already happened. For example, Allen et al. (1978) manually observed and studied a total of 25 collision scenes taped by video cameras at an intersection over a period of one year. Compton and Milton (1994) conducted a study of the safety impact of permitting turns on red lights (TOR) based on data collected from crashes. Examining actual scenes of collision and conflict, however, proves not to be ideal when aiming to prevent these events in the first place. First, crashes and accidents are a very rare occurrence in daily traffic and vary from case to case, meaning that data taken from these events cannot represent the majority of road users' behavior (Sayed et al., 2013). As pointed out by Ismail et al. (2009), collision-based safety analysis is a reactive approach which requires a significant number of collisions to be collected before an action is warranted. Second, this data is likely to be

incompletely documented or protected for legal and privacy reasons, which leads to data acquisition being complicated or even impossible. Taken together, these are the main reasons which make it almost impossible to automatically analyze the behavior of road users.

In recent years, conflict analysis as a surrogate for or proactive study in the field of collision analysis (Kaparias et al., 2010) has gained more attention. Parker Jr and Zegeer (1989) defined conflict as "an event involving two or more road users, in which the action of one user causes the other user to make an evasive maneuver to avoid a collision". Kaparias et al. (2010) proposed a method to measure and grade vehicle–pedestrian conflicts based on their severity. Four factors were used to quantify the severity of conflicts. Factor A is Time-to-Collision (TTC), which measures the time between an approach point and the potential collision between a vehicle and a pedestrian (Hayward, 1972). Factor B is the severity of the evasive action, which refers to the action that a road user, either a pedestrian or a vehicle, takes to avoid a collision. The severity is indicated by the value of the deceleration rate. Factor C is the complexity of the aforementioned evasive action (simple or complex). Factor D is the distance to collision, which measures the distance of a vehicle from the point of conflict after an evasive action has taken place. A similar method of conflict-based assessment of pedestrian safety was also used to assess the accessibility of complex intersections (Salamati et al., 2011). Shirazi and Morris (2016) provided a detailed survey of studies that analyze the behavior of road users as well as potential safety issues at intersections. The limitations of conflict analysis can be summarized as follows: (1) This method depends on the position information of road users in both space and time, i. e., trajectories. The acquisition of such information at a high level of reliability requires great effort, such as road user detection and classification, and trajectory tracking (Saunier and Sayed, 2006). (2) The studies mentioned above focused on collision and conflict events, which do not account for all traffic activities as undisturbed passages (also defined as non-interactions in this thesis) are not included. Moreover, extracting conflict data out of massive daily traffic requires domain expertise and is time-consuming. None of these studies provided an efficient way to automatically differentiate between interactions and non-interactions.

3.1.2 Automated Detection Using Computer Vision Methods

With the development of computer vision techniques in object detection and motion extraction, these techniques allow for automated analysis of road users' behavior at intersections. One of the earliest studies in this domain is the work from Ismail et al. (2009). Pedestrian–vehicle conflicts at an intersection were automatically analyzed using video data. What was considered as a breakthrough in this domain was the automation of trajectory extraction for vehicles and pedestrians directly from videos (Saunier and Sayed, 2006). Later, several works used trajectories extracted from videos to analyze before-and-after vehicle–pedestrian conflicts (Ismail et al., 2010) in street designs with elements of shared space (Kaparias et al., 2013), in less organized traffic environments (Tageldin and Sayed, 2016), and in the context of vehicle–bicycle conflicts at intersections (Sayed et al., 2013).

The general pipeline of automated conflict analysis using video data can be summarized as *data acquisition*, *pre-processing*, *data storage* and *analysis*, as shown in Fig. 3.2. The first step is to acquire adequate high-quality data. In order to provide such data, several aspects have to be considered (Jackson et al., 2013). For example, both the position and the elevation of the camera are important for setting up a proper field-of-view of the traffic. In addition to this, the power supply and the storage on the memory card of the camera have to be considered before the installation of the camera. Last but not least, the intrinsic and extrinsic parameters of the camera are necessary for camera calibration and removing distortions. The second step of data preprocessing, then, consists of three important processes: object detection, tracking and projection.

In recent years, object detection techniques have made huge progress and their accuracy has increased significantly (Girshick et al., 2014, 2015). Some state-of-the-art deep learning detectors have achieved real-time detection rate (Redmon et al., 2016; Redmon and Farhadi, 2017, 2018; Zhao et al., 2019a), a development which is presented in more details in Sec. 2.2.1 and 2.2.2. Tracking is the follow-up process to detection. It identifies the same object in a sequence of consecutive video frames. The step of projection transfers the trajectories from a local coordinate system (i. e., image pixels) to a global coordinate system (e. g., in meters) in such a way that an accurate analysis based on trajectories can be carried out. In the third step, trajectory data and conflict data are stored in a database for later usage. In the fourth and final step of conflict analysis, important events that may lead to collisions are identified via conflict indicators, such as TTC, Post-Encroachment Time (PET), Deceleration-to-Safety Time (DST), and Gap Time (GT) (Ismail et al., 2009). PET is the interval between the moment an offending road user leaves an area of potential collision and the moment of arrival of a conflicted road user possessing the right-of-way (Cooper, 1984). DST is the necessary deceleration to reach a non-negative PET value if movements of the conflicting road users remain unchanged (Hupfer, 1997). GT is similar to PET. It projects the movement of the interacting road users into space and time (Archer, 2004). The work carried out by Ni et al. (2016) analyzed pedestrian–vehicle interaction patterns using the aforementioned conflict indicators. First, trajectories were extracted by the semi-automated image processing tool Traffic Analyzer (Suzuki and Nakamura, 2006), and then interactions were classified into three categories (hard interaction, soft interaction and no interaction) according to the speed profiles extracted from the trajectories and the above indicators TTC and GT. On the one hand, the work carried out by these authors is very similar to the studies presented in this thesis. Both interactions and non-interactions are studied at permissive right-turn intersections. On the other hand, this work for analyzing interaction patterns is not fully automated in terms of data processing. Moreover, it only considers interactions between vehicles and pedestrians.

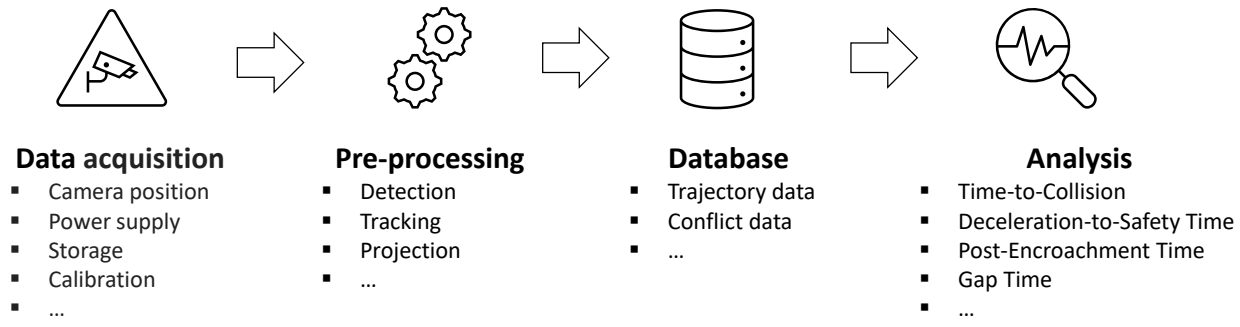


Figure 3.2: The process of traffic conflict analysis via video data.

In very recent years, deep learning methods have been successfully applied to understand road users' behavior at intersections. Rasouli et al. (2017) released a large-scale dataset which was created in urban environments from the perspective of a self-driving car in order to facilitate traffic scene analysis and the study of pedestrians' behavior. They provided a method based on CNNs to classify whether a pedestrian will cross at an intersection using the information of gait/attention and environment context (e. g., width of the road, pedestrian crossing, signs and traffic lights). Similarly, Hoy et al. (2018) proposed to use variational tracking networks to estimate whether a pedestrian will cross or stop in front of a car. The detected position of the pedestrian and the image taken from a front camera mounted on the car were used as the input for the binary classification task. The survey paper by Rasouli and Tsotsos (2019) reviewed the modern deep learning methods and identified several relevant factors for pedestrians' intent detection from an ego-perspective (self-driving car). For example, dynamic features (e. g., past movements), vision-

based features (e. g., 3D body post of pedestrians and structure of the street), and social contextual features (e. g., social connections to other road users) are used to help understand the behavior of pedestrians interacting with self-driving cars. Most of the studies were carried out from an ego-perspective. There appears, however, to be a lack of deep learning methods to automatically classify interactions between vehicles and all the other road users at intersections from a third-person perspective. This is not only important for autonomous driving in urban traffic, but also for traffic management and safety monitoring at uncontrolled or less controlled intersections.

To summarize, there are several research gaps in the existing studies: (1) The video data captured by a camera at intersections is often of a low quality because of e. g., bad light conditions, and the distortion of the images cannot be removed in the absence of camera parameters. Sometimes, the videos are collected by a third party and the parameters for projecting the videos to a global coordinate system are inaccurate or difficult to recover. In addition, acquiring reliable trajectory data is often costly and time-consuming. Furthermore, the quality of the data is difficult to guarantee. For example, tracking multiple objects from frame to frame is very challenging due to, e. g., abrupt object motion, change of appearance, and occlusions (Yilmaz et al., 2006). Errors and failures in the process of detection will propagate to the process of tracking, which will later lead to wrong conclusions in the analysis (Sayed et al., 2013). (2) The majority of the non-deep learning studies mentioned above focused on vehicle–pedestrian interactions (Ismail et al., 2009, 2010; Kaparias et al., 2013; Tageldin and Sayed, 2016; Ni et al., 2016) and only a few studies on vehicle–bicycle interactions (Sayed et al., 2013). However, in real-world traffic situations at big intersections, other heterogeneous road users are often involved. (3) There is a lack of deep learning methods for automated interaction detection in the temporarily shared spaces of intersections. Therefore, in this thesis, a deep learning pipeline that does not rely on the process of trajectory data is proposed, instead learning the interactions directly from videos.

3.2 Trajectory Prediction

Trajectory prediction has been studied for decades. In this section, both the relevant conventional and modern approaches are discussed.

3.2.1 Expert vs. Data Driven

In the paradigm of trajectory prediction for the near future (measured in seconds), most of the approaches can be generally categorized into two classes: expert-based and data-driven approaches.

Expert-based approaches integrate the knowledge of so-called domain experts (e. g., traffic engineers) into a model for solving a pre-defined problem. Rule-based models, e. g., Cellular Automata (CA) (Bandini et al., 2017b), and force-based models, most notably the Social Force Model (SFM) (Helbing and Molnar, 1995), are the most prominent expert-based approaches to simulating the motion behavior (conceived as trajectories) of road users.

In CA models, the environment is gridded into identical discrete cells and a set of pre-defined rules (e. g., acceleration/deceleration and randomization) is used to control the movement of road users. CA models were applied to simulate car movement in freeway traffic (Nagel and Schreckenberg, 1992) and a driver's decision-making process (car-following, lane changing, amber running, and right-turn filtering) at intersections (Chai and Wong, 2015). A two-dimensional CA model was used to simulate pedestrian dynamics (Burstedde et al., 2001) and a stochastic mechanism was introduced to a CA model to simulate pedestrian movement with heterogeneous speed profiles (Bandini et al., 2017b). CA models were also used to simulate the interactions between pedestrians and

vehicles at crosswalks (Zhang and Duan, 2007) and at non-signalized intersections (Bandini et al., 2017a).

Concurrent with CA models, SFMs were proposed for modeling pedestrian dynamics (Helbing and Molnar, 1995; Asano et al., 2010; Moussaïd et al., 2010). In the classical SFM (Helbing and Molnar, 1995), different motivations and interactions of road users are described by differential equations adding up a set of simple attractive and repelling forces influencing the movement of road users at a specific place and time. For example, the attractive force drives a pedestrian towards her or his goal and the repelling force pushes the road user away from obstacles to avoid collision. SFM was extended for modeling the movement of vehicles—therefore, separate forces had to be added to simulate the influence of vehicles on pedestrians (Zeng et al., 2014; Yang et al., 2018). New forces were added to simulate short-range conflicts and rule-based constraints were set for long-range conflicts (Anvari et al., 2015). Long-range collision avoidance mechanisms were adopted to extend SFM for modeling the behavior of both vehicles and bicycles (Rinke et al., 2017).

Furthermore, other types of rule-based approach have been proposed to model complex interactions between road users. In simple scenarios, both CA-based and SFM-based models work well. However, it proves difficult to mimic the complex decision-making process of human road users through either of these models. To this end, decision-theoretical models such as probabilistic and game-theoretical models (Osborne and Rubinstein, 1994; Schönauer, 2017) were proposed. They were used for handling complex conflict scenarios in which agents need to choose an action among different alternatives to maximize their performance (Bjørnskau, 2017).

Data-driven approaches are those trained/learned from the data provided to them, rather than based on the rules manually designed by experts. Modeling trajectories as sequences is one of the most common data-driven approaches. The 2D/3D positions of an agent are predicted step by step. The widely applied models include linear regression and Kalman filter (Harvey et al., 1990), Gaussian processes (Tay and Laugier, 2008) and Markov decision processing (Kitani et al., 2012). These approaches largely rely on the quality of manually designed features and have limited performance in tackling large-scale data.

In recent years, deep learning models have become the most prominent models for trajectory prediction. As a subset of data-driven models, deep learning empowers a machine with stacked layers (neural networks) to automatically discover representations from raw data for, e. g., detection or classification tasks (LeCun et al., 2015). The most distinctive attribute of deep learning models is that they do not require careful feature engineering and considerable domain expertise to design a set of rules, which is very different from the aforementioned expert-based approaches. With proper structure, i. e., RNNS with the LSTM architecture (as showin in Sec. 2.1.3), they can be used for sequence modeling and solving time series tasks. The most cited and pioneering deep learning model for predicting pedestrian trajectory while considering interactions is the so-called S-LSTM model (Alahi et al., 2016). It uses an LSTM to encode the position at each discrete time step for a pedestrian agent based on its past trajectory, and a social pooling layer to model interactions between target and neighboring agents based on an occupancy grid map.

In order to investigate the advantages and disadvantages of expert-based and deep learning approaches, Cheng et al. (2020a) proposed a common framework to compare them. In this framework, both types of model take the same data as input and generate their respective output for comparison. Namely, the S-LSTM model was extended to build a model called LSTM-DBSCAN (Cheng and Sester, 2018b), taking into account the social connections among pedestrian group members by using the clustering algorithm DBSCAN (as introduced in Sec. 2.4). LSTM-DBSCAN was compared with a rule-based model called GSFM (Johora and Müller, 2018), which combines a

game-theoretical model with an SFM for trajectory prediction of heterogeneous road users in shared spaces. Based on the empirical results for dealing with real-world trajectory data, the strengths and weaknesses of the GSFM and LSTM-DBSCAN models are summarized in Table 3.1. In addition, the expert-based and the deep learning models described above were combined for trajectory prediction (Johora et al., 2020). Simulation results showed the combined model to outperform both pure approaches in predicting realistic and collision-free trajectories.

Table 3.1: Pros and cons of GSFM and LSTM-DBSCAN.

Model	GSFM (expert-based)	LSTM-DBSCAN (deep learning)
Pros	transparent, explainable, collision-free trajectories, no need for labeled data, easy to control	less domain knowledge, not based on rules, good short-term predictions, realistic predictions in simple scenarios
Cons	domain knowledge, complicated rules, homogeneous predictions, inflexible when dealing with scaled problems, limited in dense traffic	not transparent, not explainable, collision-free trajectories not guaranteed, computationally inefficient, might be overfitted, limited in dense traffic, requires labeled data, hard to control

3.2.2 State-of-the-Art Deep Learning Approaches

With the fast development of deep learning architectures (Goodfellow et al., 2014; Mirza and Osindero, 2014; Kingma and Welling, 2014; Kingma et al., 2014; Sohn et al., 2015; Vaswani et al., 2017) and the increasing availability of large-scale real-world trajectory benchmarks (Lerner et al., 2007; Pellegrini et al., 2009; Robicquet et al., 2016; Bock et al., 2019), deep learning approaches have continuously improved the performance of trajectory prediction (Alahi et al., 2016; Lee et al., 2017; Vemula et al., 2018; Gupta et al., 2018; Xue et al., 2018; Zhang et al., 2019; Giuliari et al., 2021). In the following section, state-of-the-art deep learning approaches developed for trajectory prediction in recent years will be further discussed, particularly with respect to interaction modeling, attention mechanisms, and generative models for multi-path trajectory prediction.

Modeling Interactions between Agents and Environment The behavior of an agent can be crucially affected by others and the environment. Therefore, effectively modeling the interactions between dynamic context (neighboring agents) and static constraints created by the environment is important for accurate trajectory prediction.

Dynamic context is related to the impacts created by dynamic objects (i. e., road user agents), either via individual agents or via agents within a group. As discussed above, the negotiation between agents could be simulated by Game Theory or SFM. Such rule-based interaction models have been incorporated into deep learning models. S-LSTM employs an occupancy grid to map the positions of close neighboring agents and uses a social pooling layer to encode the interaction information for trajectory prediction. Many authors designed their specific "occupancy" grid for modeling interactions among agents (Lee et al., 2017; Xue et al., 2018; Hasan et al., 2018). In the early studies, Cheng et al. incorporated agents' field-of-view according to their mode of transport (Cheng and Sester, 2018a) and collision probability density mapping based on safety distance (Cheng and Sester, 2018b) to improve the occupancy grid for interaction modeling. Apart from occupancy gird, different pooling mechanisms are proposed. The S-GAN model (Gupta et al., 2018) embeds relative positions between the target and all of the other agents and then uses element-wise pooling to extract the interaction of the partners for each pair of agents, respectively; The SR-LSTM (States

Refinement LSTM) model (Zhang et al., 2019) employs a states refinement module to align all of the agents and adaptively refines the state of each agent through a message passing framework. However, only the position information is leveraged in most of the deep learning models presented in this chapter. The interaction dynamics are not fully captured, neither in spatial nor temporal domains. On the other hand, group agents are treated the same as individual agents, rather than as one connected component. Close interaction between group members therefore may erroneously be interpreted as a risk of collision. This will, consequently, degrade the prediction performance and may even lead to failure in real-world traffic situations.

Among individual road users, clustering pedestrian groups is essential for understanding road users' collective behavior and predicting their trajectory. Pedestrians belonging to the same group also tend to walk at the same speed and maintain a certain distance from each other, thereby synchronizing movement within the same group to facilitate communication and visibility between the group members (Yamaguchi et al., 2011; Rinke et al., 2017). Most of the early work on group detection was carried out by manual observations based on the distance maintained between road users and their coexisting time (Aveni, 1977; Moussaïd et al., 2010), or describing specific movement patterns (Laube et al., 2005; Sester et al., 2012). These detection methods are expensive to scale up to a large number of pedestrians. In the context of trajectory prediction, Bisagno et al. (2018) extended the S-LSTM model with connections between group members. They proposed to cluster the trajectories of agents that conduct coherent motion using the Kanade-Lucas-Tomasi tracker (Tomasi and Kanade, 1991). Conflict interactions between those agents in the same cluster are excluded in the occupancy grid map. Similarly, this thesis proposes to use DBSCAN to cluster pedestrians with similar motion patterns (Cheng et al., 2019). Interactions between group members and non-group agents are treated differently. Both (Bisagno et al., 2018) and (Cheng et al., 2019) that consider grouping behavior reported improved performance for trajectory prediction in comparison to approaches that consider no grouping behavior.

Static contexts are the constraints created by the environment, such as the layout of a space and the deployment of traffic facilities. A large body of recent deep learning models have proven that exploring scene information from the environment can improve the performance of pedestrian trajectory prediction. Bartoli et al. (2018) extended the occupancy map in the S-LSTM model with a "context-aware" pooling to consider the static objects in the neighborhood of a person for predicting her or his future trajectory. The static objects are manually located in the cell of the occupancy map based on the relative position of the target agent. Xue et al. (2018) proposed to use a CNN to automatically extract the scene information from a top–down image of the space. Such scene information is concatenated with the motion information of the target agent's past trajectory and its interactions with neighboring agents for predicting its future trajectory. In the work of (Sadeghian et al., 2018b), static information is also leveraged from top–down images. The scene information is first extracted from a pre-trained deep neural network VGGnet-19 (Simonyan and Zisserman, 2014b) for object detection. Then, such information is encoded by two types of attention mechanisms. A single-source attention mechanism attends to the visual information that is located in one particular area and a multi-source attention mechanism attends to the scattered visual cues in the scene. Similarly, this approach was adopted by Sadeghian et al. (2019) to learn the physical constraints from top–down images for trajectory prediction. Zhao et al. (2019b) proposed a Multi-Agent Tensor Fusion network for contextual trajectory prediction. The Multi-Agent Tensor spatially aligns the scene encoding from the image of the environment with the motion encoding learned from each agent in the scene. Then these two encodings are fused by a fully convolutional mapping to obtain both agent-to-agent interaction and agent-to-environment interaction. In this thesis, CNNs are also used to explore scene information from the environment for trajectory prediction in shared spaces.

Modeling with Attention Recently, different attention mechanisms (Bahdanau et al., 2015; Xu et al., 2015; Luong et al., 2015; Vaswani et al., 2017; Wang et al., 2018) have been incorporated into neural networks for learning complex spatio–temporal interconnections. Their effectiveness has been proven in learning powerful representations from sequence information, e. g., in neural machine translation (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017) and image caption generation (Xu et al., 2015).

Attention mechanisms have been utilized to extract semantic information for predicting trajectories. A soft attention mechanism (Xu et al., 2015) is incorporated into LSTMs to learn the spatio–temporal patterns from the position coordinates (Varshneya and Srinivasaraghavan, 2017). The model named SoPhie (Sadeghian et al., 2019) applies two separate soft attention modules: The physical attention learns salient agent-to-scene features and the social attention learns agent-to-agent interactions. In the MAP model (Move, Attend, and Predict) (Al-Molegi et al., 2018), an attentive network is implemented to learn the relationships between location and time information. The most recent work Ind-TF (Giuliari et al., 2021) utilizes the Transformer network (Vaswani et al., 2017) for modeling trajectory sequences. The Transformer network is a special neural network architecture that purely depends on the self-attention mechanism for modeling sequences and is widely applied in machine translation for sequence prediction. In this thesis, dynamic interactions among all road users are attentively learned by utilizing the Transformer self-attention mechanism. Sec. 2.5 presents this concept in more detail.

Generative Models Generative models incorporate a set of random variables that enable the models to map a single input to many possible outputs, the so called one-to-many mapping (Sohn et al., 2015). To date, VAE (Kingma and Welling, 2014) and GANs (Goodfellow et al., 2014) and their variants, e. g., Conditional VAE (Kingma et al., 2014; Sohn et al., 2015) and Conditional GAN (Mirza and Osindero, 2014), are the most popular generative models. Each of them is able to generate diverse outputs by sampling from noise. The essential difference is that GAN trains a generator to generate a sample from noise and a discriminator to decide whether this sample is real enough or not. The generator and discriminator enhance each other during training. In contrast, VAE is trained by maximizing the lower bound of training data likelihood for learning a latent space that approximates the distribution of the training data (as shown in Sec. 2.6).

Predicting one single trajectory may not be sufficient due to the dynamic and complex behavior of agents, as well as interactions between agents. Hence, generative models are proposed for multi-path trajectory prediction. Lee et al. (2017) proposed a CVAE model to predict multi-path trajectories conditioned on the observed trajectories. Gupta et al. (2018) trained a generator to generate future trajectories from noise and a discriminator to judge whether these trajectories are fake or not. The performances of the two modules enhance each other and the generator is able to generate realistic trajectories. Amirian et al. (2019) proposed a GAN-based model for generating multiple plausible trajectories for pedestrians.

To summarize this section, the discussion of the related work on trajectory prediction identified the most important aspects that impact the movement of an agent. (1) How an agent moves in the near future not only depends on its past trajectory, but also on its interactions with other neighboring agents, either in a group or as individuals, and on its interactions with the environment as well. (2) Attention mechanisms are adopted to extract the salient features for modeling interactions. (3) Generative models are used to predict multiple possible trajectories for an agent while taking into account any uncertainty associated with this task. However, based on the majority of the work discussed in this section, most of the approaches are designed for pedestrian trajectory prediction. The behavior of other types of agents, especially in mixed traffic with heterogeneous agents, has

not been fully studied yet. In a shared-space environment, e. g., the level of uncertainty increases when a pedestrian or a cyclist has to directly interact with vehicles. This makes the prediction task more challenging than predicting trajectories for homogeneous agents. To this end, this thesis investigates all of the above aspects and designs novel deep generative models to predict the behavior of heterogeneous agents in shared spaces.

4 Methodological Contributions

This chapter presents the methodology for interaction detection (Sec. 4.1) and trajectory prediction (Sec. 4.2) in shared spaces. The methodology employs Conditional Variational Auto-Encoder (CVAE) for probabilistic inference of interaction detection and multi-path trajectory prediction.

4.1 Interaction Detection

In this section, the methodology of interaction detection is explained in detail. Sec. 4.1.1 gives the mathematical definition of the problem and introduces the model proposed in this thesis. Then the following sections contain the most important processes that are necessary for implementing the model. Namely, Sec. 4.1.2 explains how the task is converted into a sequence-to-sequence problem, with Sec. 4.1.3 providing the estimation of the model's uncertainty, and Sec. 4.1.4 describing the process of feature extraction.

4.1.1 Problem Formulation and the Proposed Model

With regard to road users' behavior, interaction detection is defined as a classification problem using the information extracted from videos of vehicles turning and VRUs crossing in the temporarily shared spaces of intersections. It is mathematically formulated as follows: Given a set of observed vehicle turning sequences of video frames $\mathbf{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_i, \cdots\}$, the *input* of the i-th vehicle turning sequence is characterized as $\boldsymbol{X}_i^{(T)} = \{X_i^t\}_{t=0}^{T-1}$, where $X^t \in \mathbb{R}^{W \times H \times C}$ is the frame at time step t, and T is the total number of observed frames for the i-th sequence. W, H, and C denote the width, height, and number of channels of the frame. Instead of using raw images, object information and optical flow information (Sec. 4.1.4 provides a more detailed approach to this issue) are extracted from the video frames and used as input data sequence. $\boldsymbol{Y}_i$ is the corresponding *ground truth* interaction label and $\hat{\boldsymbol{Y}}_i$ is the *prediction*. It should be noted that the interaction label is a dichotomous class that represents the interaction level of the whole sequence and does not provide detailed information of how the interaction level changes at each time step (frame).

The model for predicting the probability of interaction between a turning vehicle and other crossing road users is defined as:

$$\begin{aligned} \hat{\boldsymbol{Y}} &= f(\boldsymbol{Y}|\boldsymbol{X}), \\ &= \arg\max_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \mathbf{z}), \\ &\simeq -D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}, \boldsymbol{Y})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}, \boldsymbol{Y})}[\log p_\theta(\boldsymbol{Y}|\boldsymbol{X}, \mathbf{z})], \end{aligned} \tag{4.1}$$

where f is a CVAE model (as introduced in Sec. 2.6). It encodes the information of interactions into a latent space and predicts the interaction label $\hat{\boldsymbol{Y}}$ conditioned on the input $\boldsymbol{X}$. The model jointly trains a recognition model, $q_\phi(\cdot)$, which is also called encoder, and a generative model, $p_\theta(\cdot)$, which is also called decoder. In the training phase, $q_\phi(\cdot)$ encodes the observed information and the ground truth label into the latent variables $\mathbf{z}$. $p_\theta(\cdot)$ then decodes the prediction of the interaction label conditioned on the input and the latent variables. The Kullback-Leibler divergence $D_{KL}(\cdot)$ pushes the approximate posterior $q_\phi(\cdot)$ to the prior distribution $p_\theta(\mathbf{z})$. The generation

error $\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X},\boldsymbol{Y})}(\cdot)$ measures the distance between the generated output $\hat{\boldsymbol{Y}}$ and the ground truth $\boldsymbol{Y}$. In the inference phase, the decoder predicts the interaction label conditioned on the input of the observed information and a latent variable directly sampled from the Gaussian prior $p_\theta(\mathbf{z})$.

The overall loss function, Eq. (4.2), consists of two losses—the Kullback-Leibler divergence loss and the reconstruction loss. The binary cross–entropy loss is used as the reconstruction loss, as denoted by Eq. (4.3).

$$\mathcal{L} = D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X},\boldsymbol{Y})||\mathcal{N}(0,\mathbf{I})) + \mathcal{H}(\hat{\boldsymbol{Y}} - \boldsymbol{Y}), \tag{4.2}$$

where

$$\mathcal{H}(\hat{\boldsymbol{Y}} - \boldsymbol{Y}) = -\{\boldsymbol{Y}\log\hat{\boldsymbol{Y}} + (1-\boldsymbol{Y})\log(1-\hat{\boldsymbol{Y}})\}. \tag{4.3}$$

4.1.2 Sequence-to-Sequence Processing

Sequence-to-sequence processing maps an input sequence to an output sequence, e. g., speech recognition of phonemes (Graves et al., 2006) or translation from one language into another (Sutskever et al., 2014). The classification task of interaction detection as defined above is initially a sequence-to-one problem due to the way the data structure is labeled: An interaction class label refers to the whole sequence of frames. It is not feasible to manually label each frame due to the tremendous amount of work required. Without knowing the exact fine-grained frame-wise interaction label, the sequence-wise label is duplicated at each frame. Hence, the form of the output is converted to obtain the time steps, denoted as $\boldsymbol{Y}_i^{(T)} = \{Y_i^t\}_{t=0}^{T-1}$ for the i-th turning sequence. Thereafter, the input and output sequences are aligned at each frame. The initial problem is turned into a sequence-to-sequence classification problem, denoted by Fig. 4.1 and Eq. (4.4).

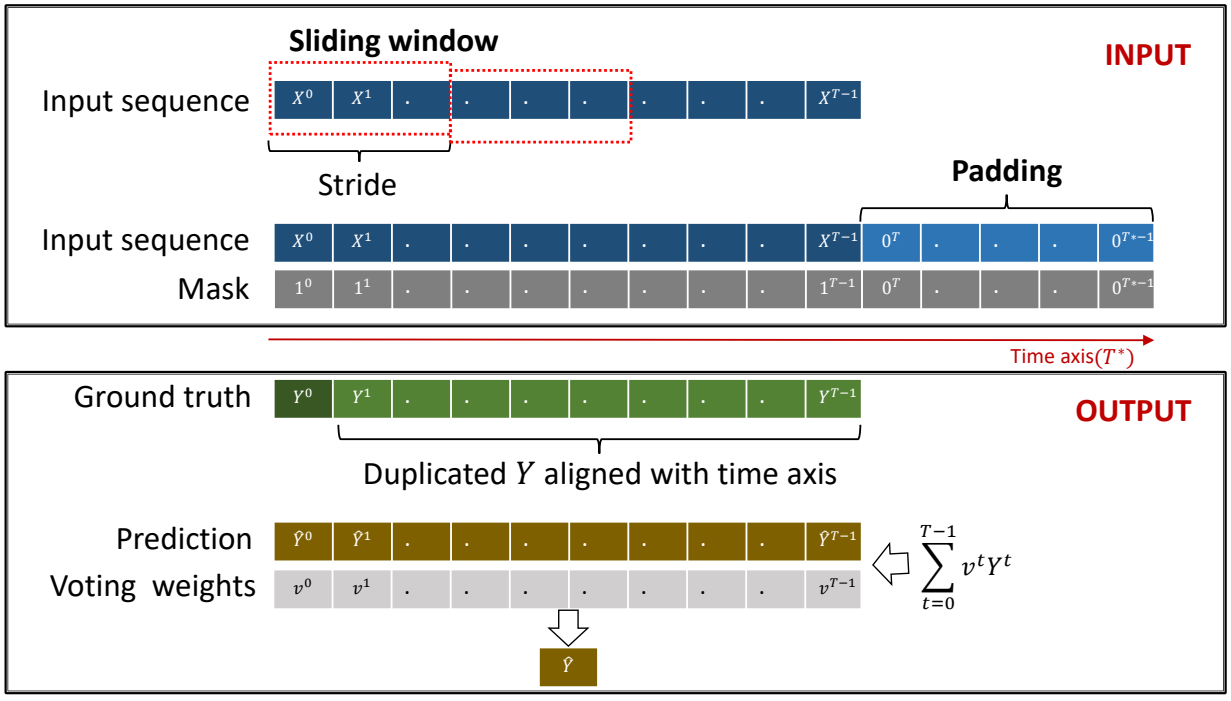


Figure 4.1: Sequence-to-sequence modeling using sliding window or padding method.

$$p(\boldsymbol{Y}_i|\boldsymbol{X}_i,\mathbf{z}) = \lambda\sum_{t=0}^{T-1} p(\boldsymbol{Y}_i^{(T)}|\boldsymbol{X}_i^{(T)},\mathbf{z}), \tag{4.4}$$

where λ is a voting scheme that summarizes the frame-wise interaction predictions to the sequence-wise interaction prediction.

The conversion of sequence-to-sequence is carried out with the following considerations: (1) The sequence-wise label only represents the overall interaction level for the given sequence, while the intensity of interaction is likely to change over time due to the modification of distance and orientation between a turning vehicle and VRUs. (2) Over a large dataset, sequence lengths are bound to differ from one to another, which provides rich interaction information including both long and short sequences. (3) The reconstruction loss between prediction and ground truth is still computed at the sequence level, because each frame-wise prediction only partially contributes to the sequence-wise prediction using the voting scheme. This mechanism enables the model to automatically learn the frame-wise dynamics at each frame in training time.

As already discussed in a prior paper (Cheng et al., 2020c), there are different voting schemes that weigh each frame equally or proportionally depending on its order along the time axis. In this thesis, the *average voting scheme* that weighs the prediction at each frame equally is adopted. The sequence-wise prediction is then the class label voted by the majority.

Different methods are proposed to deal with the variation in sequence lengths. The most commonly used RNNs for sequence modeling often require a fixed sequence length. However, at an intersection, some vehicles can quickly complete their turning maneuver if the space happens to be free. But some vehicles may have to wait for a long time to let pedestrians and cyclists cross first. Two methods, *sliding window* and *padding*, are proposed to deal with varying sequence lengths, as shown in Fig. 4.1. The sub-sequences divided by the sliding window or padded sequences with a fixed sequence length are treated as the input for training an RNN-based CVAE model for all the sequences.

The sliding window method divides each sequence into small sub-sequences with a fixed window size w. Eq. (4.5) denotes the sliding window method, in which the stride is set to be the same as w. The overlap between two consecutive windows is allowed when the stride is set to be smaller than the size of the window.

$$\boldsymbol{X_i}^{(T)} = \{X_i^0, \cdots, X_i^{\mathsf{w}1}, \cdots, X_i^{\mathsf{w}k}\}, \text{ where } k = \frac{T}{\mathsf{w}}. \tag{4.5}$$

The padding method uses zero-paddings to extend the sequences shorter than a predefined sequence length T^* by adding zero values at the end of the sequences. T^* can be adapted to cover most of the sequences, e. g., $T^* = \text{Max}\{T_1, \cdots, T_i, \cdots\}$, where T_i denotes the length of an arbitrary vehicle turning sequence. A padding mask is used to annotate the exact sequence length so that the padded zero values are treated differently to mitigate the negative impact on the learning process.

$$\boldsymbol{X}_i^{(T)} = \{X_i^0, \cdots, X_i^{T-1}, 0^T, \cdots, 0^{T^*-1}\}, \tag{4.6}$$

$$\text{Mask}_i^{(T)} = \{1_i^0, \cdots, 1_i^{T-1}, 0^T, \cdots, 0^{T^*-1}\}, \text{ where } T < T^*. \tag{4.7}$$

4.1.3 Estimation of Uncertainty

Due to the stochastic property of the latent variables, the above sequence-to-sequence model is not deterministic. In inference time, a latent variable can be sampled multiple times from the Gaussian prior and each of these samples will be different. This process can lead to different predictions. To this end, Kernel Density Estimation (KDE) (Parzen, 1962; Loftsgaarden et al., 1965) is used to measure the uncertainty of the predictions made through this multi-sampling process.

At each frame, the predictions $\{Y_{i,1}^t, \cdots, Y_{i,N}^t\}$ are assumed to be i.i.d. samples drawn from an unknown density function $f(Y)$, where N is the total number of predictions. Here, $t \leq T$, and T is the total steps of the given sequence i. The kernel density estimator is calculated as:

$$\hat{f}(Y)^t = \frac{1}{N}\sum_{i=1}^{N} K_h(Y - Y_i) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{Y - Y_i}{h}\right), \tag{4.8}$$

where $K(\cdot)$ is the Gaussian kernel function and h is the smoothing parameter (also called *bandwidth*). The likelihood of the average prediction at step t is determined by $\mathcal{L}(\log(\hat{f}), \bar{Y_i})^t$, where $\bar{Y_i}$ is the average prediction. The uncertainty is defined as the residual of the normalized log-likelihood averaged over all the steps for the given sequence i, as denoted by Eq. (4.9):

$$\Gamma_i = 1 - \frac{1}{T}\sum_{t=0}^{T-1} \omega\mathcal{L}(\log(\hat{f}), \bar{Y_i})^t, \tag{4.9}$$

where ω is the normalization parameter that scales the values to [0,1], and Γ_i stands for the degree of uncertainty for the prediction over the sequence i.

4.1.4 Feature Extraction

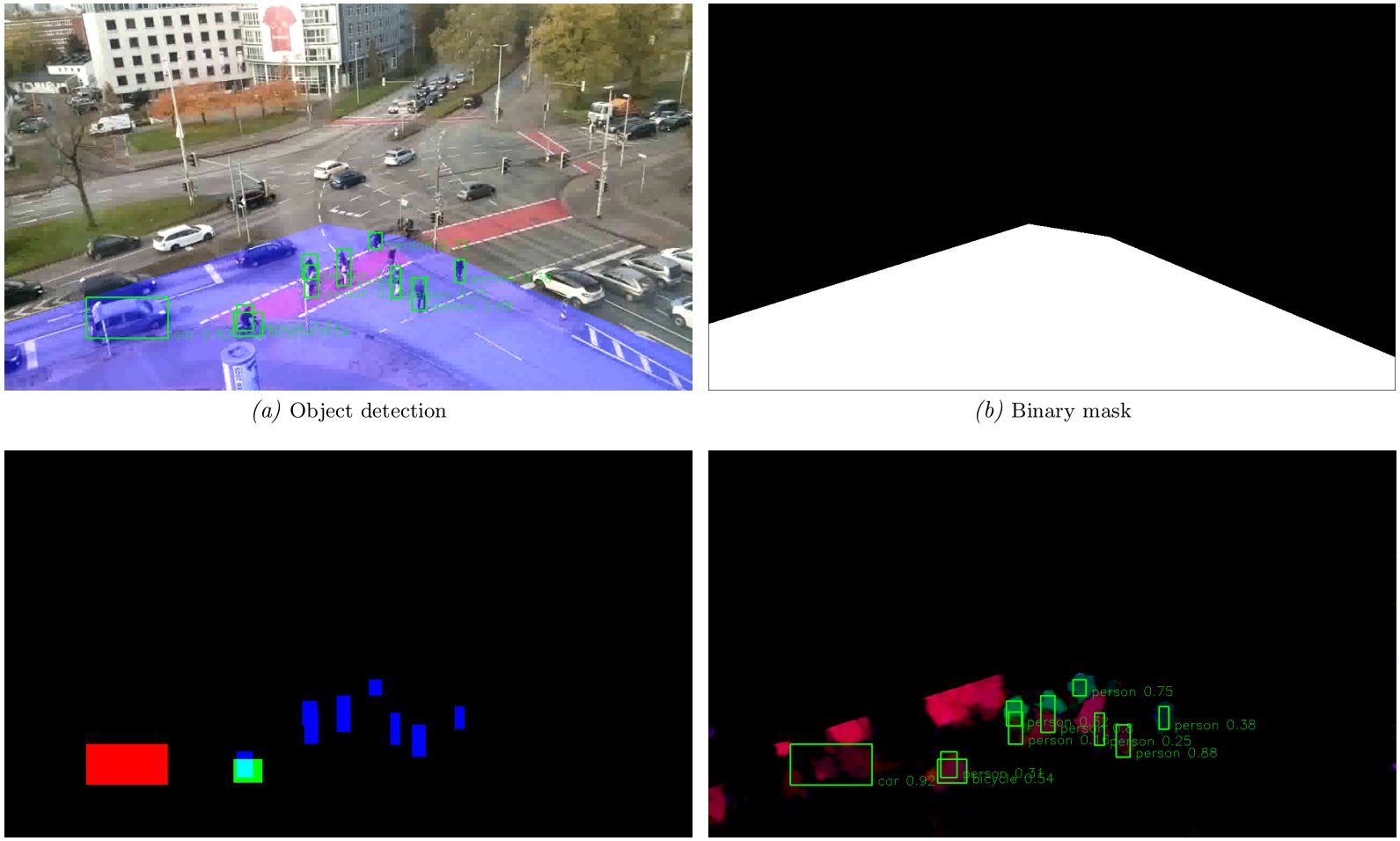
Two types of information, object and optical flow information, which have both been extracted from video frames, are used as input features for the interaction detection task, as shown in Fig. 4.2 and Table 4.1.

Object information contains road users' type and location. The deep learning object detectors YOLOv3 (Redmon et al., 2016) and M2Det (Zhao et al., 2019a), as described in Sec. 2.2.1 and Sec. 2.2.2, respectively, are leveraged for detecting all of the relevant road users at each frame[1]. Namely, the road users are classified as pedestrians, cyclists, motorbikes, cars, trucks, and buses. Different channels are used to store the position information for each of these road users and each channel is then dedicated to one or two similar road user types, as visualized in Table 4.1. The location of the detected road users (Fig. 4.2a) is mapped by the corresponding bounding boxes with values of one in each frame (Fig. 4.2c, they are color-coded for different types of road users). Non-detected areas in the frame are set with values of zero, as shown in black.

Optical flow is used to capture the motion of road users. It describes the distribution of velocities of moving pixels' brightness in two consecutive images (Horn and Schunck, 1981). Moving objects are captured by optical flow, whereas and static objects and background are ignored. The dense optical flow algorithm (Farnebäck, 2003), as presented in Sec. 2.3, is applied to map the displacement of moving objects and remove the static background information. An example of this distinction can be seen in Fig. 4.2d. Similarly, respective frame channels are dedicated to the orientation and velocity of the moving objects, as shown in Table 4.1.

The area of interest covering the temporarily shared space of the intersection is marked by a binary mask (Fig. 4.2b); other areas—while interesting to the analysis of road users' behavior in general—are not taken into account as this study focuses entirely on the turning intersections. There is, however, one disadvantage of the mask—a disadvantage caused by the camera's perspective. The mask of the area of interest slightly extends into the through lane next to the turning lane, as

[1] Comparing the advantages and disadvantages of these two object detectors remains an issue for future study; as far as this thesis is concerned, it would go beyond the scope of what the proposed model is meant to achieve. Therefore, only one detector is used for each dataset.



(a) Object detection

(b) Binary mask

(c) Object information

(d) optical flow information

Figure 4.2: Input features for interaction detection. (a) Detection was done on the original video frame. The area of interest covering the temporarily shared space of the intersection is marked in the transparent blue color. Areas of no interest are blocked out by the binary mask (b) in both (c) and (d). (c) The bounding boxes are used to locate the positions of all detected objects. To improve the visibility of the different channels within the images, the figure shown here only exemplifies three channels. Pedestrians are denoted in blue, bicycle(s) in green and car(s) in red. (d) Here, the optical flow depicts the motion of dynamic objects. Static objects are treated as background. The optical flow information is coded through HSV color representation. Orientation of motion is represented by color hue and velocity by color value. In the resulting representation, the overlaid bounding boxes in the optical flow figure only serve the purpose of showing the location of the objects, including the static ones. They are not integrated into the optical flow information itself.

Table 4.1: Object and optical flow information extracted by object detectors and the dense optical flow algorithm, respectively. W, H, and C denote the width, height, and number of channels of the video frame. Based on the acquired data that only very few motorbikes were detected, they are stored in the same channel as bicycles. Cars/trucks are stored in one shared channel, too, due to the very similar trace of their turning sequence.

Feature type	Frame size	Channel-1	Channel-2	Channel-3	Channel-4	value
Object	$W \times H \times C$	pedestrians	bicycles/motorbikes	cars/trucks	buses	{0, 1}
Optical flow*	$W \times H \times C$	orientation	1	velocity		[0, 1]

*The HSV (Hue, Saturation, Value) color representation is used to store optical flow information. The hue channel (Channel-1) is used to store orientation, the saturation channel (Channel-2) is set to its maximum, and the value channel (Channel-3) is used to store velocity.

shown in Fig. 4.2a and 4.2b. Due to the oblique view of the camera, the upper parts of the vehicles in the turning lane are partially projected into the through lane. The extended mask therefore must aim to include the upper parts of the turning vehicles to be detected as well. The lower middle point of the bounding boxes of detected vehicles is used to filter out the vehicles in the through lane.

The extended mask, however, also introduces noise to the optical flow information. For instance, as shown in Fig. 4.2d, the motion of the vehicles in the through lane is captured by the optical flow as well and cannot be easily filtered out due to the irregular shapes and occlusion. Fortunately, it turns out that this noise is not problematic when both the object information and the optical flow information are combined as the input for training the interaction detection model. Interactions between vehicles and VRUs only happen in the crossing zone and the noise of optical flow for the through lane vehicles is ignored by the model in the learning process.

4.2 Trajectory Prediction

In this section, the methodology of trajectory prediction is explained in detail. Sec. 4.2.1 gives the mathematical definition of the problem and introduces the proposed model for multi-path trajectory prediction. Sec. 4.2.2 then explains how the trajectories predicted for an agent are ranked, and Sec. 4.2.3 describes the process of feature extraction.

4.2.1 Problem Formulation and the Proposed Model

The task of trajectory prediction is treated as modeling a conditional distribution $p(\boldsymbol{Y}_n|\boldsymbol{X})$, where $\boldsymbol{X}$ is the previous trajectory information and $\boldsymbol{Y}_n$ is one of its possible future trajectories. As discussed in the previous chapters, the future trajectory of an agent depends on both internal and external stimuli that influence an agent's behavior (Rudenko et al., 2020). In this thesis, the internal stimulus mainly focuses on physical motion of the target agent and the external stimulus is mainly coming from two aspects: agent-to-agent interaction $\boldsymbol{I}$ and agent-to-environment interaction $\boldsymbol{S}$ (more details in Sec. 4.2.3.2 and Sec. 4.2.3.3). To this end, trajectory prediction for an agent is extended to predicting its multiple future trajectories conditioned on the observed trajectory and the interactions between agents and environment. It is defined as follows: agent i receives as input its observed trajectories $\boldsymbol{X}_i = \{X_i^1, \cdots, X_i^T\}$ for predicting its n-th plausible future trajectory $\hat{\boldsymbol{Y}}_{i,n} = \{\hat{Y}_{i,n}^1, \cdots, \hat{Y}_{i,n}^{T'}\}$. $n \leq N$, and N denotes the total number of the predicted trajectories; T and T' denote, respectively, the total steps of the past and future trajectories. The trajectory position of i is characterized by the coordinates as $X_i^t = ({x_i}^t, {y_i}^t)$ at step t and $\hat{Y}_{i,n}^{t'} = (\hat{x}_{i,n}^{t'}, \hat{y}_{i,n}^{t'})$ at step t'. 3D coordinates are also possible, but in this thesis only 2D coordinates are considered. The objective is to predict multiple plausible future trajectories $(\hat{\boldsymbol{Y}}_{i,1}, \cdots, \hat{\boldsymbol{Y}}_{i,N})$ that are as accurate as possible with respect to the corresponding ground truth $\boldsymbol{Y}_i$.

The CVAE model is adopted to solve this problem. The model encodes the uncertainty of an agent's behavior into a latent space and predicts its trajectory in the near future conditioned on the agent's past trajectory information and interaction information, as denoted by Eq. (4.11).

$$\begin{aligned} \hat{\boldsymbol{Y}}_{i,n} &= f(\boldsymbol{Y}_i|\boldsymbol{X}_i, \boldsymbol{I}_i, \boldsymbol{S}_i), \\ &= \arg\max_{\boldsymbol{Y}} p(\boldsymbol{Y}_i|\boldsymbol{X}_i, \boldsymbol{I}_i, \boldsymbol{S}_i, \mathbf{z}), \end{aligned} \tag{4.10}$$

where

$$\begin{aligned} \log p(\boldsymbol{Y}_i|\boldsymbol{X}_i, \boldsymbol{I}_i, \boldsymbol{S}_i, \mathbf{z}) \simeq & -D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}_i, \boldsymbol{Y}_i, \boldsymbol{I}_i, \boldsymbol{S}_i)||p_\theta(\mathbf{z})) \\ & + \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}_i, \boldsymbol{Y}_i, \boldsymbol{I}_i, \boldsymbol{S}_i)}[\log p_\theta(\boldsymbol{Y}_i|\boldsymbol{X}_i, \boldsymbol{I}_i, \boldsymbol{S}_i, \mathbf{z})]. \end{aligned} \tag{4.11}$$

Similar to the interaction detection model (Sec. 4.1.1), the trajectory prediction model jointly trains a recognition model $q_\phi(\cdot)$ and a generative model $p_\theta(\cdot)$. In the training phase, $q_\phi(\cdot)$ encodes $\boldsymbol{X}_i$, $\boldsymbol{Y}_i$, $\boldsymbol{I}_i$, and $\boldsymbol{S}_i$ into the latent variables $\mathbf{z}$. Then, $p_\theta(\cdot)$ decodes the future trajectory conditioned on the input of $\boldsymbol{X}_i$, $\boldsymbol{I}_i$ and $\boldsymbol{S}_i$ in the observation time concatenated with the latent variables $\mathbf{z}$. For simplicity's sake, the time notation is omitted. $q_\phi(\cdot)$ accesses the interaction information $\{\boldsymbol{I}_i, \boldsymbol{S}_i\}_{t=1}^{T+T'}$ from both observation and prediction time, while $p_\theta(\cdot)$ only accesses the interaction information $\{\boldsymbol{I}_i, \boldsymbol{S}_i\}_{t=1}^{T}$ from the observation time[2]. The Kullback-Leibler divergence $D_{KL}(\cdot)$ pushes the approximate posterior $q_\phi(\cdot)$ to the prior distribution $p_\theta(\mathbf{z})$. The generation error $\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{I}_i, \boldsymbol{S}_i)}(\cdot)$ measures the distance between the generated output $\hat{\boldsymbol{Y}}$ and the ground

[2]The step is counted from one instead of zero, exactly as it is counted for the interaction detection task defined in Sec. 4.1.1. This is because the zeroth step has no offset of the position information, as it shown in Sec. 4.2.3.1.

truth $\boldsymbol{Y}$. In the inference phase, the decoder predicts future trajectory conditioned on $\boldsymbol{X}_i$, $\boldsymbol{I}_i$, $\boldsymbol{S}_i$ and a latent variable directly sampled from the Gaussian prior $p_\theta(\mathbf{z})$. The sampling process of the latent variable is repeated N times to generate N future trajectories for multi-path trajectory prediction.

The overall loss function consists of two losses—the Kullback-Leibler divergence loss and the reconstruction loss. $L2$ loss (Euclidean distance) is used as the reconstruction loss.

$$\mathcal{L} = D_{KL}(q_\phi(\mathbf{z}|\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{I}, \boldsymbol{S})||\mathcal{N}(0, \mathbf{I})) + \|\hat{\boldsymbol{Y}} - \boldsymbol{Y}\|_2. \tag{4.12}$$

4.2.2 Trajectory Ranking

For tasks of single-path prediction, a ranking strategy is proposed to select the *mostlikely* predicted trajectory out of the multiple predictions. A bivariate Gaussian distribution is applied to rank the predicted trajectories $(\hat{\boldsymbol{Y}}_{i,1}, \cdots, \hat{\boldsymbol{Y}}_{i,N})$ for each agent. This method was first proposed by Graves (2013) for generating handwriting, a process which can be seen as generating connected points on a 2D plane and therefore quite similar to trajectory prediction. Instead of considering a Gaussian mixture distribution at each step as in (Graves, 2013), a bivariate Gaussian distribution with a single component is implemented in this thesis. At step t', all of the predicted positions for the agent i are stored in the vector $|\hat{\mathsf{X}}_i, \hat{\mathsf{Y}}_i|^{t'}$ for estimating a bivariate Gaussian probability density function:

$$f(\hat{x}_i, \hat{y}_i)^{t'} = \frac{1}{2\pi\sigma_{\hat{\mathsf{X}}_i}\sigma_{\hat{\mathsf{Y}}_i}\sqrt{1-\rho^2}} \exp \frac{-Z}{2(1-\rho^2)}, \tag{4.13}$$

where

$$Z = \frac{(\hat{x}_i - \mu_{\hat{\mathsf{X}}_i})^2}{{\sigma_{\hat{\mathsf{X}}_i}}^2} + \frac{(\hat{y}_i - \mu_{\hat{\mathsf{Y}}_i})^2}{{\sigma_{\hat{\mathsf{Y}}_i}}^2} - \frac{2\rho(\hat{x}_i - \mu_{\hat{\mathsf{X}}_i})(\hat{y}_i - \mu_{\hat{\mathsf{Y}}_i})}{\sigma_{\hat{\mathsf{X}}_i}\sigma_{\hat{\mathsf{Y}}_i}}. \tag{4.14}$$

μ denotes the mean and σ the standard deviation. ρ is the correlation between $\hat{\mathsf{X}}_i$ and $\hat{\mathsf{Y}}_i$. A predicted trajectory is scored as the sum of the relative likelihoods over all of this particular trajectory's steps:

$$S(\hat{\boldsymbol{Y}}_{i,n}) = \sum_{t'=1}^{T'} f(\hat{x}_i, \hat{y}_i)^{t'}. \tag{4.15}$$

The predicted trajectories are ranked according to this score. The one with the highest score stands out for the single-path prediction.

4.2.3 Feature Extraction

This subsection discusses in detail how to extract features for modeling motion, agent-to-agent, and agent-to-environment interactions.

4.2.3.1 Motion of Agents

The motion information for each agent is parameterized by the position coordinates at each step. Specifically, the offset of the trajectory positions between two consecutive steps $(\Delta x^t, \Delta y^t) = (x^{t+1} - x^t,\ y^{t+1} - y^t)$ is used as the motion information, which has been widely applied in this domain (Gupta et al., 2018; Becker et al., 2018; Zhang et al., 2019). Compared to coordinates, the offset is independent from the given space and less sensitive with respect to overfitting a model to a particular space or travel directions. It is interpreted as speed over steps which are defined

with a constant duration. The coordinates at each position are calculated back by cumulatively summing the sequence offsets of the given original position.

The class information of an agent's type is useful for analyzing its motion. One-hot encoding is applied to combine the type information with the motion information. For example, $\{\Delta x^t,\Delta y^t,1,0,0\}$ is used to represent the offset for a pedestrian agent, $\{\Delta x^t,\Delta y^t,0,1,0\}$ for a cyclist agent, and $\{\Delta x^t,\Delta y^t,0,0,1\}$ for a vehicle agent. This strategy is only adopted if the agent's type information is given by the experimental datasets. Otherwise, only the offset $(\Delta x^t, \Delta y^t)$ is used for the motion information instead.

Random rotation is used as a data augmentation technique. Namely, trajectories are randomly rotated to prevent the model from only learning certain directions. In order to maintain the relative positions and angles between agents, the trajectories of all agents coexisting in a given period are rotated by the same angle.

4.2.3.2 Agent-to-Agent Interaction

Agent-to-agent interaction models how the target agent's motion is influenced by its neighboring agents, i. e., closely connected as a group or separated by a certain distance to avoid collision between non-group members. For two different agents i and j that co-exist with each other within a certain distance, the agent-to-agent interaction is modeled as $I_{i,j} = \pi(\boldsymbol{X}_i, \boldsymbol{X}_j)$ based on their position information over time. I is the notation for interaction and π is the mapping function. A density-based clustering method is used for group detection and two methods—occupancy grid and attentive dynamic maps—are investigated for modeling interactions between agents.

Grouping models the connection relationship between the target agent and its neighboring agents. It is used in addition to the interaction mapping functions for collision avoidance.

The definition of *group* follows (Moussaïd et al., 2010): a pedestrian group normally forms a convex hull (linear formation or V-like pattern) with close distance maintained by at least two members over a certain duration of time. Instead of the complex sociological collective behavior also often described with the same term (Campbell, 1958), here, the concept of grouping is purely focused on the similar motion shared by group members, i.e, moving close to each other in space and time at synchronized speed. Due to the very similar behavior of group members, if the neighboring agent is a group member of the target agent, this neighboring agent is excluded from the interaction mapping functions. This way, the mapping functions are designed to focus on learning interactions for collision avoidance, rather than learning mixed interactions for both collision avoidance between non-group members and connections between group members.

A density-based clustering algorithm (Cheng et al., 2019) based on DBSCAN (Ester et al., 1996) is used to detect the group members (if there are any) for the target agent i during its total observed steps T, denoted by Algorithm 1. Agent j is assigned to the group set $\mathsf{G}(i)$ of i, only if its coexisting time ratio with i being in the same *cluster* exceeds a predefined threshold ζ.

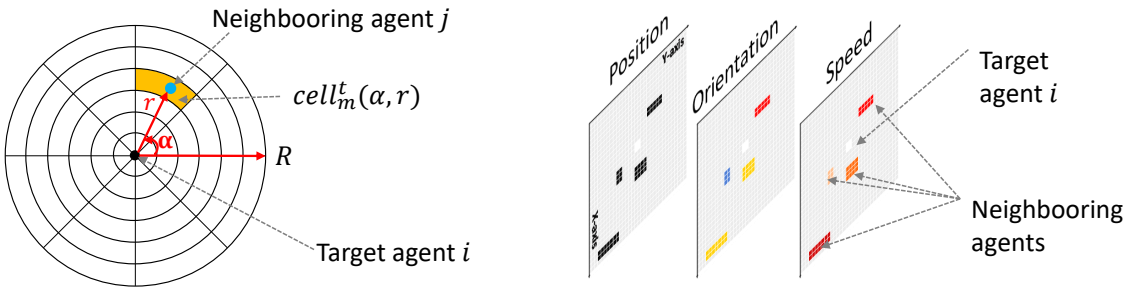
Occupancy grid models the interactions between the target agent and its neighboring non-group member agents by a grid map at each step. The surrounding area of the target agent i is polar-shaped, centered by its current location, and divided into grid cells. The radius of the map is denoted as R, as can be seen in Fig. 4.3a. The occupancy grid is calculated using the following equation:

$$\begin{aligned} \mathrm{cell}_m^t(\alpha, r) = \sum 1[j \in \mathsf{N}(i) \ \wedge \ j \notin \mathsf{G}(i)], \\ \text{where } \alpha \in [0°, 360°) \ , \ r \leq R. \end{aligned} \tag{4.16}$$

Algorithm 1: Grouping

Result: $\mathsf{G}(i)$
Agents i, $\exists j$ with $i \neq j$, $count = 0$, $\mathsf{G}(i) = \emptyset$;
for $t \leq T$ **do**
  if $\exists$ *cluster* $(i,j)_t$ **then**
    count $\leftarrow$ count $+ 1$;
  end
end
if $(count\,/\,T) \geq \zeta$ **then**
  $\mathsf{G}(i) \leftarrow \mathsf{G}(i) + j$;
end

$\text{cell}_m^t(\alpha,r)$ stands for the m-th cell with an angle of α and a distance of r to the centroid at step t. $\mathsf{N}(i)$ and $\mathsf{G}(i)$ stand, respectively, for the set of neighbors at the time t, and for the set of group members with respect to the target agent. If there is a non-group member in $\text{cell}_m^t(\alpha, r)$, its value will be added by one. At each step in time, the occupancy density of each cell changes according to the position between the target and neighboring agents. This process, therefore, maps the interactions. It should be noted that this occupancy grid method is more similar to the polar mapping method proposed by Xue et al. (2018) than it is to the mapping method of S-LSTM as proposed by Alahi et al. (2016). S-LSTM uses a rectangular shape and predefined channel numbers to model a limited amount of neighboring agents, i. e., each channel is dedicated to one neighboring agent. The modified form is able to map all neighboring agents using only one channel through the addition function.



(a) Occupancy grid *(b)* Dynamic map

Figure 4.3: Agent-to-agent interaction modeled via occupancy grid and dynamic map at each step.

Attentive dynamic maps extend the occupancy grid method described above. Agent-to-agent interaction at each step is modeled via three layers dedicated to orientation, speed, and position information. Each map changes from one step to the next and therefore the spatio–temporal interaction information between agents is interpreted dynamically over time.

The map is defined as a rectangular area around the target agent, divided into grid cells centered on the agent's current location. This grid is shown in Fig. 4.3b. W and H denote the width and height. Instead of a polar-shaped map, the dynamic map with three layers is analogous to an RGB image with three color channels, and a convolutional neural network can be easily employed for extracting the spatial features from the map. The following three steps explains the mapping process in detail.

First, a target agent i is chosen. In relation to this particular agent, the neighboring non-group agents are mapped into the closest grid cells$^t_{w\times h}$ according to their relative position. At the same time, they are also mapped into the cells reached by their anticipated relative offset (speed) in x and y direction.

$$\begin{aligned} &\text{cells}^t_w = x^t_j - x^t_i + (\Delta x^t_j - \Delta x^t_i), \\ &\text{cells}^t_h = y^t_j - y^t_i + (\Delta y^t_j - \Delta y^t_i), \\ &\text{where } w \leq W,\, h \leq H,\, j \in \mathsf{N}(i) \wedge j \notin \mathsf{G}(i). \end{aligned} \tag{4.17}$$

Second, orientation, speed, and position information for each neighboring agent are stored in the mapped cells of the respective layers. The *orientation* layer O stores the heading direction. For the neighboring agent j, its orientation from the current to the next position is the angle ϑ_j in the Euclidean plane and calculated as $\vartheta_j = \arctan2(\Delta y^t_j, \Delta x^t_j)$. Similarly, the *speed* layer S stores the travel speed and the *position* layer P stores the positions using a binary flag in the cells mapped above for each neighboring agent.

Last, layer-wise, a Min-Max normalization scheme is applied for normalization. Compared to the above occupancy grid, in addition to modeling the position information, the dynamic map at each step adds two more channels to explicitly model both orientation and speed information.

The map covers a large vicinity area. Empirically, $32 \times 32\,\mathrm{m}^2$ is found to be a proper setting considering both the coverage and the computational cost. There is a trade-off between a high and a low resolution map. A cell is filled by a maximum of one agent if its size is small. But the high resolution will lead to a very sparse map in which most of the cells have no value, and the surrounding areas of the neighboring agent will be treated as having no impact on the target agent. On the other hand, there may exist an overlap of multiple agents in one cell with a very different travel speed or orientation if the cell size is too big. In this thesis, this problem is solved by setting the cell size to $1 \times 1\,\mathrm{m}^2$. Based on the distribution of the experimental data, there are only a few cells with overlapping agents, which is also supported by the preservation of personal space (Gérin-Lajoie and McFadyen, 2005).

Furthermore, an attention mechanism is incorporated to learn complex spatial–temporal patterns from the sequence of the dynamic maps. Interactions between different agents are dynamic in various situations from one step to another in a sequence. Some steps may impact the agents' behavior more than the other steps. To explore such varying information, the Transformer encoder with the self-attention mechanism (Vaswani et al., 2017) is employed to learn the interaction information from the dynamic maps over time. They are called *attentive dynamic maps*. The Transformer encoder (detailed in Sec. 2.5) takes the dynamic maps defined above as input. The self-attention module is trained to assign a weight to the dynamic information derived from the dynamic map of each step (denoted as Values (V)) based on how much the latent state of the current step (denoted as Query (Q)) matches the latent states of the other steps (denoted as Key (K)). The latent states of Q, K, and V are computed via three learnable linear transformations which—separately—use the same input:

$$\begin{aligned} Q =& \pi(\text{Map}) W_Q,\, W_Q \in \mathbb{R}^{D\times d_q}, \\ K =& \pi(\text{Map}) W_K,\, W_K \in \mathbb{R}^{D\times d_k}, \\ V =& \pi(\text{Map}) W_V,\, W_A \in \mathbb{R}^{D\times d_v}, \end{aligned} \tag{4.18}$$

where W_Q, W_K, and W_V are trainable parameters and $\pi(\cdot)$ indicates the encoding (mapping) function of the dynamic maps. The attention module outputs a weighted sum of the values V, where the weight assigned to each value is determined by the dot-product of Q and K:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^{\mathbf{T}}}{\sqrt{d_k}}\right)V, \tag{4.19}$$

where $\sqrt{d_k}$ is the scaling factor, d_k is the dimensionality of the vector K, and $\mathbf{T}$ stands for transpose. The *multi-head attention* (Vaswani et al., 2017) strategy is adopted to enhance the expressiveness of the self-attention module. With this strategy, the information extracted from the dynamic maps is jointly attended to through different representation sub-spaces in different positions. Sec. 2.5 provides a more detailed description of this strategy.

4.2.3.3 Agent-to-Environment Interaction

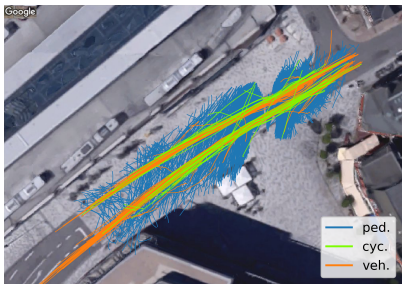
Agent-to-environment interaction models how the target agent's motion is influenced by the environmental scene context of the space. To create the model proposed in this thesis, three types of scene context are studied for agent-to-environment interaction: *aerial photograph*, *accessibility map* and *heat map*. The scene context information is automatically extracted using CNNs. More specific settings of the CNNs can be found in Chapter 6.

An aerial photograph is an RGB image taken from a bird's-eye view to provide the global context of the space. It represents the arrangement of, e. g., buildings, trees, and streets, as seen in Fig. 4.4b.

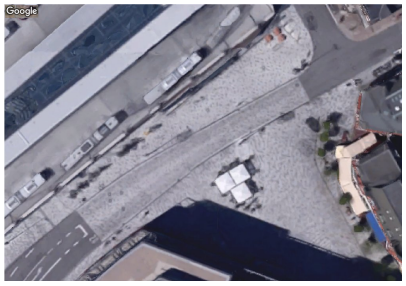
Accessibility maps are binary masks showing the different areas that can be accessed by road users depending on their transport mode. In this thesis, the accessibility maps are manually annotated based on the deployment of traffic facilities and road geometry. The second row of Fig. 4.4 shows the examples of accessibility maps for pedestrians (Fig. 4.4c), cyclists (Fig. 4.4d), and vehicles (Fig. 4.4e). The accessible areas (e. g., the road and the sidewalk) are presented in white, while inaccessible areas (e. g., walls and vegetation) are presented in black.

Heat maps are statistical distributions of the accumulated observed trajectories in the past that are used for trajectory prediction. With the assumption that road users tend to follow the trajectories of others, the areas visited more often in the past are more likely to be visited in the future. Hence, heat maps for different transport modes are generated by the visited trajectories (e. g., Fig. 4.4a). A Gaussian filter with a large kernel is used to expand the possible areas to the contiguity that can be covered in order to reduce strong statistical bias. The last row of Fig. 4.4 shows exemplary heat maps for pedestrians (Fig. 4.4f), cyclists (Fig. 4.4g), and vehicles (Fig. 4.4h).

Among the three types of scene context, an aerial photograph can be easily acquired, e. g., by a camera from a bird's-eye view or from a map provider such as Google. But an aerial photograph only provides raw image information. Consequently, the semantic information of the space layout and geometry has to be learned by CNNs. In contrast to this, accessibility maps provide explicit semantic information about the space. But they require manual work for extracting this information. Meanwhile, both the aerial photograph and accessibility maps are out of date when the design of the space changes, and additional effort is required to update them. On the other hand, heat maps provide explicit information of the occupancy distribution within the space. This information is updated when more observations of trajectories become available.

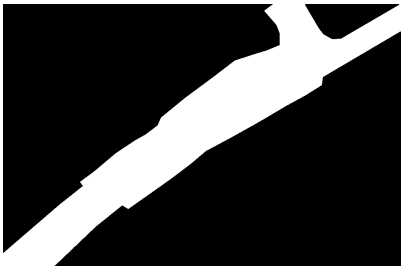


(a) Trajectories

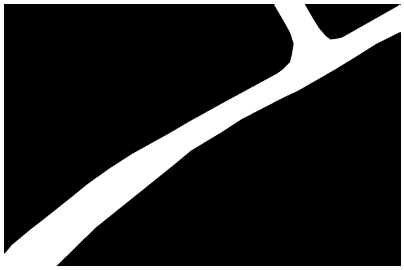


(b) Aerial photograph

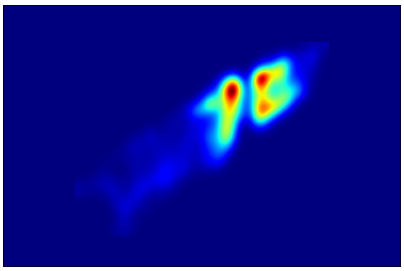
(c) Accessibility map for pedestrians



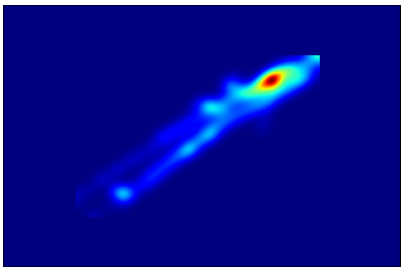
(d) Accessibility map for cyclists



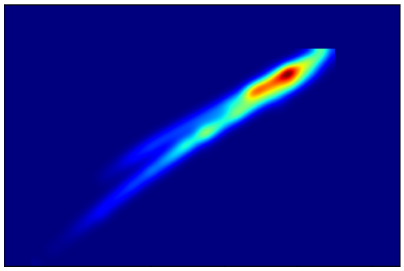
(e) Accessibility map for vehicles



(f) Heat map for pedestrians



(g) Heat map for cyclists



(h) Heat map for vehicles

Figure 4.4: Examples for agent-to-environment interaction. The trajectory data was acquired by Pascucci et al. (2017) from a shared space near a train station in the German city of Hamburg. The aerial photograph of (a) and (b) is taken from Imagery ©2017 Google. In the accessibility maps, accessible areas are presented in white while inaccessible areas are presented in black. In the heat maps, more frequently visited areas are presented in a warmer color (e. g., red) while less visited areas are presented in a colder color (e. g., blue).

5 Interaction Detection

The results presented in this Chapter 5 were submitted to the IEEE Transactions on Intelligent Transportation Systems and published in (Cheng et al., 2021a) as a preprint.

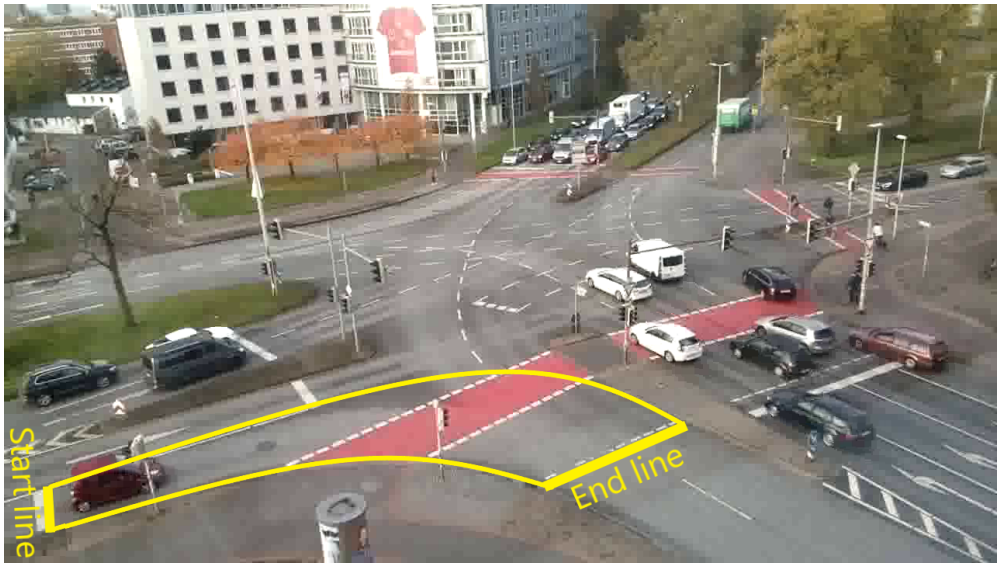
In this chapter, interaction detection using the CVAE model conditioned on the object and optical flow information extracted from videos is presented in detail. First, the data acquisition at two intersections in both right- and left-hand traffic for testing the performance of the proposed model is provided in Sec. 5.1. Then, the experiments are presented in Sec. 5.2 and the empirical results are analyzed in Sec. 5.3. Finally, Sec. 5.4 discusses the limitations of the model and the challenges of transferring the model from one intersection to the other.

5.1 Data Acquisition and Pre-processing

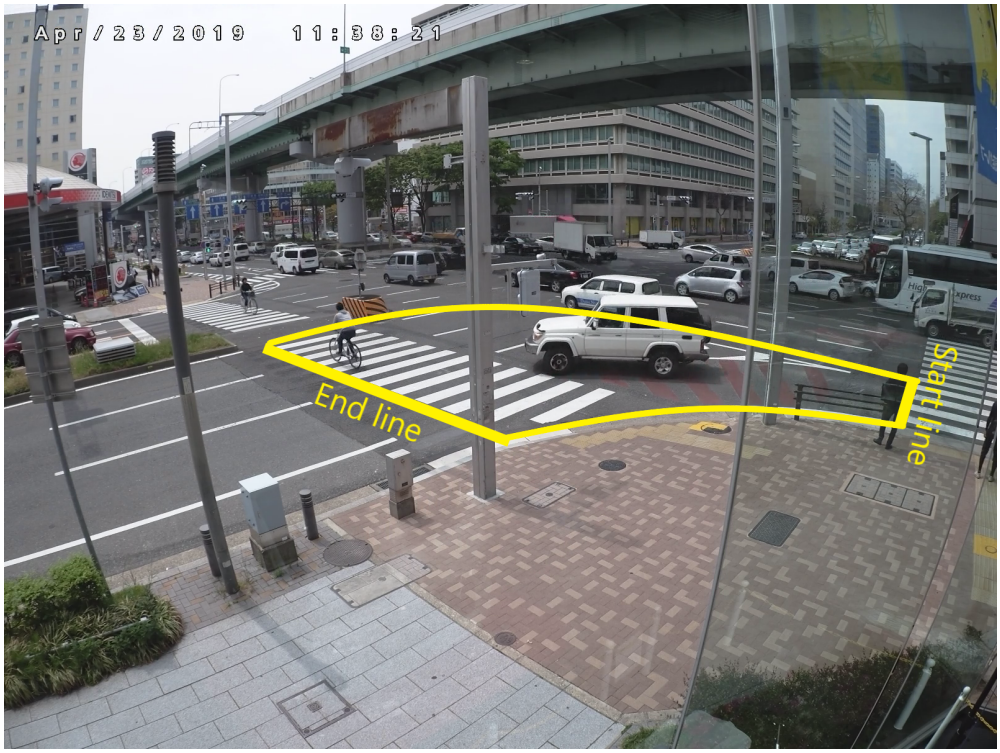
Real-world datasets of both right- and left-turn intersections are leveraged to test the performance of the proposed model for interaction detection between vehicles and VRUs in the temporarily shared spaces. The KoW dataset was acquired by Koetsier et al. (2019) from a very busy right-turn intersection in a German city. The videos recorded traffic conditions over two days from 00:02 a. m. to 11:58 p. m. on November 8th and 9th, 2019 in Hannover and were stored in five-minute segments. The videos were recorded in 1280×720 pixels at 25 fps by a camera module[1] installed inside a building (ca. 20 meters above the ground) facing the intersection, and stored in .h264 format. This thesis uses an approximately 14-hour sub-footage from two seven-hour segments (8 a. m. to 3 p. m. on both the 8th and 9th), when there was sufficient traffic and adequate ambient light to perform stable image processing for feature extraction. The NGY dataset was provided by Murase Lab at Nagoya University and Nagoya Toyopet Corporation. It was acquired from an extremely busy left-hand intersection in a Japanese city. In total, approximately 24 hours of traffic footage from an oblique view at one of the major intersections in Nagoya were recorded from 11 a. m. to 11 a. m. on April 23rd and 24th, 2019. The videos were recorded in 1600×1200 pixels at 30 fps using a camera[2] installed inside a building (ca. 3 meters above the ground) adjacent to the intersection, and stored in .mp4 format. Similarly, this thesis uses a twelve-hour sub-footage recorded from 11 a. m. to 6 p. m. on the 23rd and from 6 a.m. to 11 a.m. on the 24th. Fig. 5.1 shows the video screenshots of (a) the KoW intersection and (b) the NGY intersection.

Both datasets were pre-processed for later use. Due to the camera's lack of intrinsic and extrinsic parameters, no projection was done in order to extract the trajectory data. The pre-process aimed to identify vehicle turning sequences and to extract all the road users' type, position, and motion information. First, two annotators for each dataset manually detected the sequence scenes where vehicles turned right at the KoW intersection or left at the NGY intersection, and extracted the time interval of each vehicle staying in the yellow contour at each intersection, as depicted in Fig. 5.1. The annotators independently determined whether or not interactions occurred in each scene. Afterwards, they revised their annotations and agreed with each other[3] and labeled each scene "non-interaction" or "interaction". Then, YOLOv3 (Redmon et al., 2016) (Sec. 2.2.1)

[1] Raspberry Pi Camera Module v2
[2] Panasonic WV-SF781L
[3] Less than 1 % of the sequences were initially annotated differently



(a) The KoW intersection in 1280 × 720 pixels



(b) The NGY intersection in 1600 × 1200 pixels

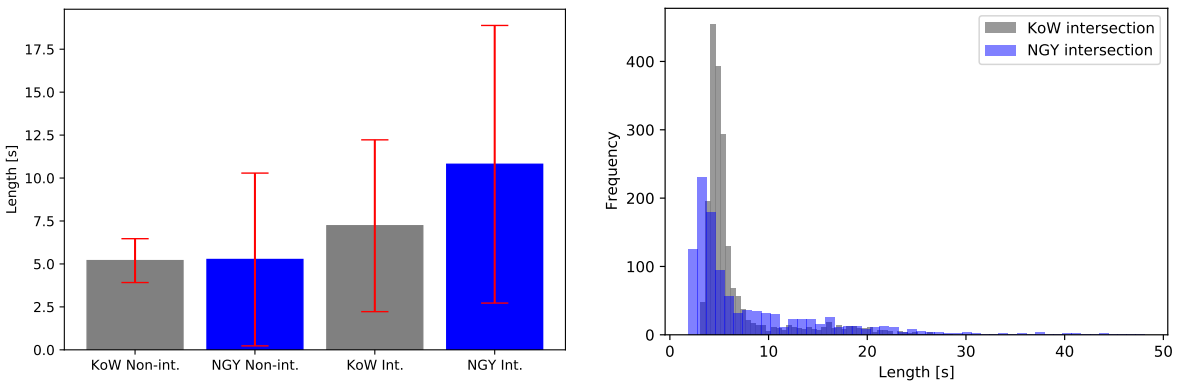
Figure 5.1: The screenshots of two intersections with different driving directions. A vehicle turning sequence is counted as the time interval of the vehicle staying in the yellow contour at each intersection.

and M2Det (Zhao et al., 2019a) (Sec. 2.2.2) were used to detect all traffic related objects at the original frame rate of the KoW (25 fps) and NGY (30 fps) datasets, respectively[4]. Considering that the change between two consecutive frames is small, the failed detection in the current frame is supplemented by the detection in the previous or the next frame if either of them was available. Otherwise, the sequences with failed detection and no supplementation were discarded. In parallel to the object detection process, the dense optical flow algorithm (Farnebäck, 2003) (Sec. 2.3) was used to extract the optical flow information from the sequences. Unlike in the object detection above, in the process of extracting optical flow the sequences were down-sampled to half the original frame rate—12.5 fps for KoW and 15 fps for NGY. The down-sampling was carried out in order to reduce the computational cost and to increase the offset of moving pixels between two consecutive frames, so as to improve the extraction performance (Farnebäck, 2003). In the end, both the object information and the optical flow information were aligned with the down-sampled frame rate in each dataset, which is used as the time step for the experiments of interaction detection. The method of feature extraction is explained in detail in Sec. 4.1.4.

The lengths of both non-interaction and interaction vehicle turning sequences are compared within and across the two datasets. Table 5.1 lists the statistics and Fig. 5.2 visualizes the sequence length distributions for each class.

Table 5.1: Statistics of the vehicle turning datasets in the temporarily shared spaces for interaction detection.

Dataset	Class label	#Sequences	Mean length (s)	Std. deviation	Frame size	Frame rate after down-sampling
KoW (Germany)	non-interaction	648	5.2	2.3	1280 × 720	12.5 fps
	interaction	1350	7.2	5.0	1280 × 720	12.5 fps
NGY (Japan)	non-interaction	545	5.3	5.0	1600 × 1200	15 fps
	interaction	551	10.8	8.1	1600 × 1200	15 fps



(a) Sequence lengths of non-interaction vs. interaction. Standard deviation is denoted by the red error bar

(b) All sequence lengths distributed over each dataset

Figure 5.2: Sequence length distributions measured in seconds.

Within each dataset, sequence lengths measured in seconds are very different. Firstly, the non-interaction sequences are significantly shorter than the interaction sequences (Mann-Whitney U test, $U = 382142$, $p \ll 0.01$ for KoW and $U = 67199$, $p \ll 0.01$ for NGY), and the standard

[4]Due to the fact that these two sources of data were sourced from different providers, the camera settings and the object detection algorithms were not unified.

deviation in each class in both datasets varies by a large range, as presented in Fig. 5.2a and Table 5.1. The distributions of the sequence lengths indicate that the duration of a sequence is not an accurate feature for the detection task, i. e., a short sequence duration does not necessarily imply no interaction. Secondly, the sequence lengths of all the sequences over each dataset are very unevenly distributed, i. e., long-tail distributed, especially for the NGY dataset, as shown in Fig. 5.2b. In addition to the varying sequence lengths, the dynamics of interactions, e. g., waiting time, travel speed, location, and the number and type of involved road users, also vary from sequence to sequence. These factors taken together make the detection task very challenging, because a detection model has to be able to learn different interaction patterns in both long and short sequences.

Across the datasets, the sequences in the KoW and NGY datasets are different not only in terms of travel direction but also in frame size and rate, as well as sequence length in general. On the one hand, the non-interaction sequences from both datasets have similar sequence lengths, i. e., on average, non-interaction sequences have a length of 5.2 seconds in the KoW dataset and 5.3 seconds in the NGY dataset. On the other hand, interaction sequences in the NGY dataset have a longer average sequence length (mean = 10.8 seconds) than the ones in the KoW dataset (mean = 7.2 seconds). Due to the higher density of traffic at the NGY intersection compared to the KoW intersection, vehicles often had to wait for more pedestrians and cyclists to cross. The above differences make cross-dataset validation very difficult (more details in Sec. 5.4.2).

The acquired data (video frame sequences) was prepared for interaction detection. The preparation consists of three steps: sample balancing, sequence padding, and dataset partitioning. Table 5.2 lists the statistics of the final data used for the experiments after the preparation steps.

Table 5.2: Video frame sequences used for interaction detection. The numbers of non-interaction/interaction samples were balanced in training, validation, and test sets for both the KoW and NGY datasets.

Intersection	Input form	Max. seq. length*	Training	Validation	Test	Total
KoW	sliding win.	500	360/360	90/90	192/192	642/642
KoW	padding	100	352/352	88/88	188/188	628/628
NGY	sliding win.	500	291/291	74/74	159/159	530/530
NGY	padding	100	132/132	33/33	70/70	235/235

*sequence length is measured by the number of frames

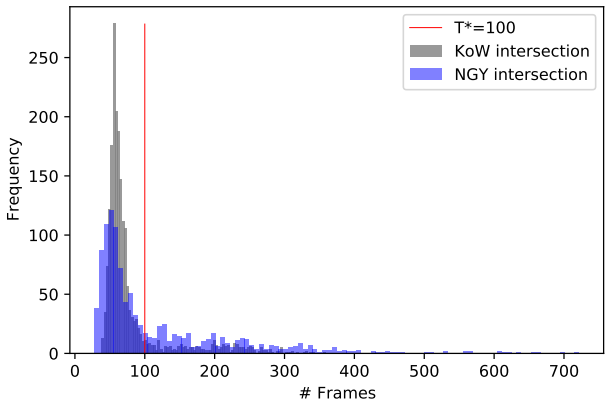


Figure 5.3: Sequence length measured by the number of frames. The red vertical line indicates the length threshold for the padding method.

Sample balancing: The number of samples in each class was balanced to guarantee unbiased training. Firstly, for both datasets the maximum number of sequences in each class was set to a value that is close to the capacity of the smaller class, namely, the non-interaction class. Secondly, the small number of very long sequences (i. e., > 500 frames, as shown in Fig. 5.3) was not used for the experiments. All sequences with over 500 frames are from the class of interaction, i. e., vehicles had to wait for a long time to let other road users cross the intersection. Removing these long interaction sequences balances sample size and sequence length in both classes in order to prevent a model being biased towards the interaction class.

Sequence padding: Sequences with a smaller number of frames than the threshold T^* are padded with zeros for the padding method (Sec. 4.1.2). There is a trade-off between setting T^* to a large value in order to include all the sequences, or to a small value in order to reduce the noise of the padded values. In other words, if T^* is too large, most of the sequences will be padded with zeros, whereas if T^* is too small, many long sequences will be excluded. Based on the sequence length distributions over the datasets (Fig. 5.3), T^* was set to 100 frames for both KoW and NGY so that the majority of the sequences were included. The sequences whose length is less than or equal to T^* (denoted as the left part of the red vertical line in Fig. 5.3) were preserved for the tests of those models that use the padding method. Sequences longer than T^* exceed the maximum length of the input size that the models can handle. Therefore, these long sequences were discarded.

Data partitioning: Under the above-mentioned balanced criteria for each class, both datasets were then randomly split into training and test sets by the ratio of $70 : 30$. Additionally, 20 % of the training data was separated as an independent validation set to monitor the process of training, as shown in Table 5.2.

5.2 Experiments

5.2.1 Pipeline

The pipeline for interaction detection between turning vehicles and VRUs using the above datasets consists of three components: *feature extraction*, *sequence processing*, and *sequence-to-sequence modeling* with the CVAE model, as shown in Fig. 5.4.

(a) The features extracted from video sequences are object information by using the deep learning object detectors YOLOv3 or M2Det, and optical flow information by using the dense optical flow algorithm. More details about feature extraction are shown in Sec. 4.1.4.

(b) These two types of information are stored as a time series frame sequence using the sliding window or padding method, and treated as the input data for the CVAE model. The interaction label is replicated at each step to align with the time steps. More details about the sliding window and padding methods are shown in Sec. 4.1.2.

(c) The sequence-to-sequence CVAE model predicts the interaction label for vehicle-turning sequences in the temporarily shared spaces of the KoW and NGY intersections as non-interaction or interaction conditioned on the object and optical flow information combined. More details about the sequence-to-sequence model are presented in Sec. 4.1.2.

5.2.2 CVAE Model for Interaction Detection

The CVAE model has an encoder–decoder structure and uses a latent space to mimic the stochastic patterns of road users' behavior. Fig. 5.5 shows the detailed structure of the model. The encoder has two branches: X-Encoder and Y-Encoder. They are dedicated to extracting low-level features

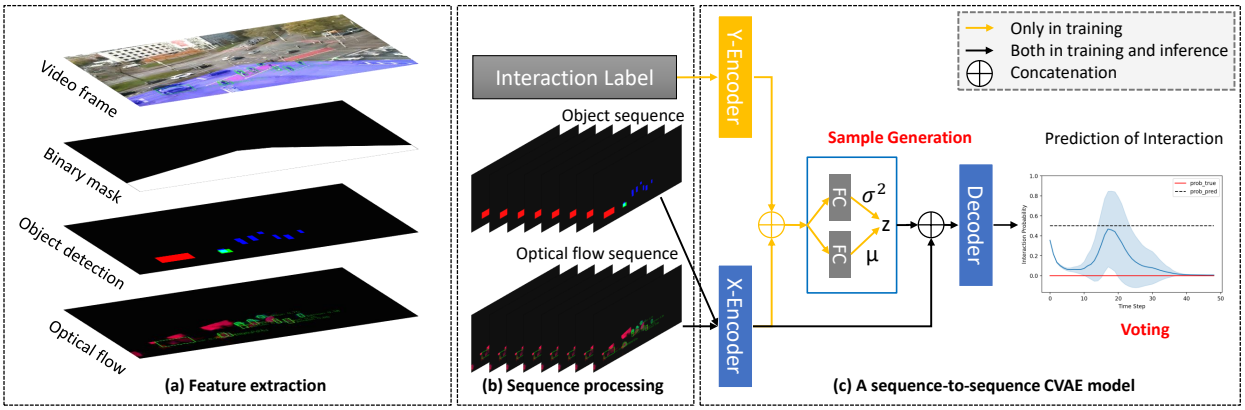


Figure 5.4: The pipeline of interaction detection. Areas of no interest are blocked out by a binary mask.

from the condition (object and optical flow information) and the interaction label information, respectively. In training time, the encoders are trained to learn an approximate recognition model $p_\phi(\mathbf{z}|\mathbf{Y}, \mathbf{X})$ of the latent variables $\mathbf{z}$. Then, the decoder is trained to generate the interaction label using the encoded condition and the latent variables. In inference time, the learned decoder is used to predict the interaction label for unseen turning sequences conditioned on the object and optical flow information, as well as from a latent variable sampled from the prior distribution $p_\theta(\mathbf{z})$. The latent variable can be sampled multiple times and may lead the decoder to generate different probabilities for each sequence at a frame level. As illustrated in Fig. 5.5, the variance of the probabilities (denoted by the cyan-shaded area in the decoder's output) describes the uncertainty of the model's prediction. Each module (i. e., X-Encoder, Y-Encoder, Latent Space, and Decoder) of the CVAE model is explained in detail as shown below.

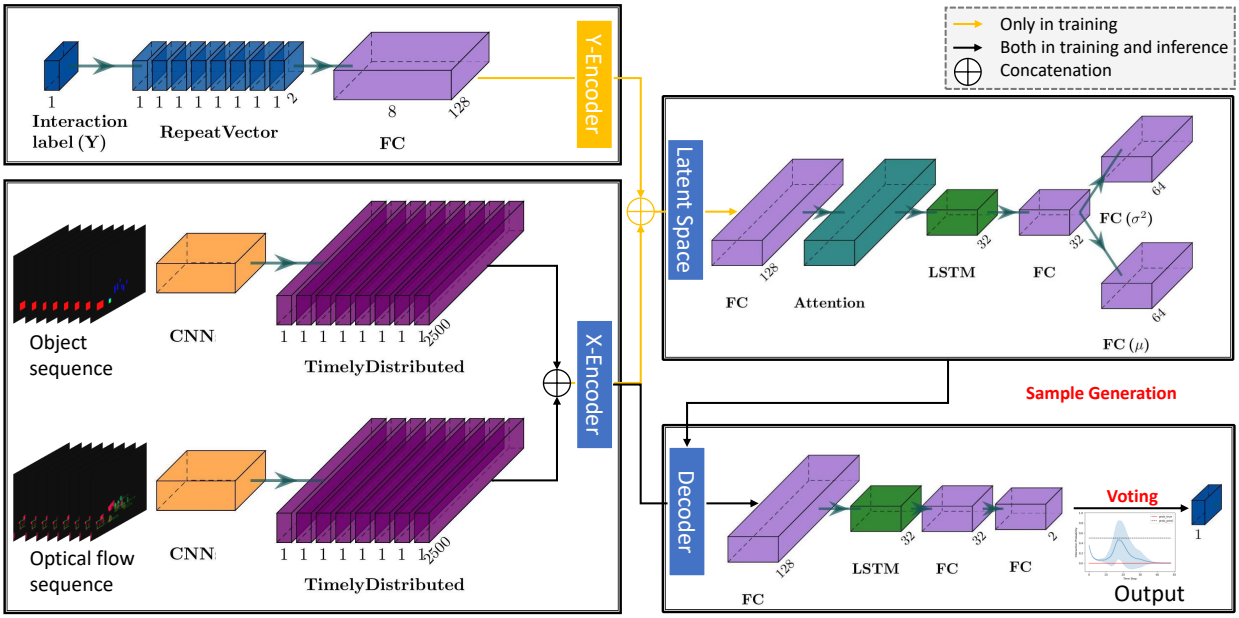


Figure 5.5: The CVAE model for interaction detection. FC stands for fully connected layer, CNN stands for convolution neural network, Attention stands for self-attention layer, and LSTM stands for Long Short-Term Memory.

The X-Encoder manipulates two CNNs for learning spatial features from object frame sequences and optical flow frame sequences, respectively. Without loss of generality, the object frame sequence

using the sliding window (e. g., $w = 8$) method is taken as an example for explaining the learning process.

- First, each frame from the sliding window is passed to a CNN to learn spatial features. As shown in Fig. 5.6, the CNN has three 2D convolutional (CONV) layers with each one followed by a Maximum Pooling (MP) layer and Batch Normalization (BN). It tasks the frame that contains object information as input and outputs a flattened feature vector. This process is repeated frame by frame for all the frames in the sliding window.
- Then, the output feature vectors of all the frames are timely distributed as a sequence that maintains the same length as the window size, as shown in Fig. 5.5 for the X-Encoder[5]. The optical flow frame sequence is processed by another CNN in a similar way to get the sequence of optical flow feature vectors.
- In the end, the sequence of object feature vectors and the sequence of optical flow feature vectors are concatenated into a 2D feature vector as the final output of the X-Encoder.

It should be noted that the CNN for the optical flow frame sequence has a similar structure, except for the input channel number. The CNN for the object frame sequence has four channels that are dedicated to different types of road user, while the CNN for the optical flow frame sequence has three channels that are dedicated to the motion information (as shown in Table. 4.1).

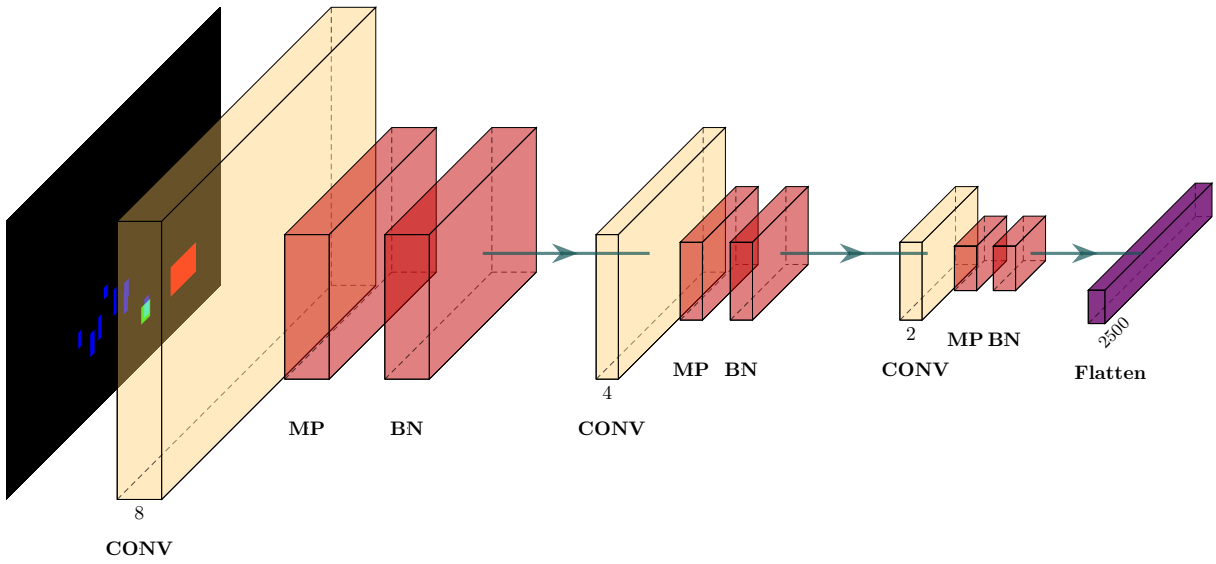


Figure 5.6: The structure of the CNN used for learning spatial features from an object frame. CONV stands for 2D convolutional layer, MP for Maximum Pooling layer and BN for Batch Normalization. In total, the CNN has three CONV layers and each is followed by an MP and BN.

The Y-Encoder embeds the interaction label for each sequence. First, the sequence-wise label is replicated to align with the sequence length. Then, a fully connected (FC) layer is used to embed the replicated labels into a label vector. The original dimensionality of the label is only two after the one-hot encoding for the non-interaction and interaction classes, which is much smaller than the combined feature vector. The embedding increases the balance of the sizes of the label

[5]This process works in the same way for the padding method with the predefined sequence length instead of the sliding window size.

vector and the combined feature vector. The specific dimensionalities, as shown in Fig. 5.5, are hyper-parameters that can be changed in the experiment settings.

The prior Gaussian latent variables $\mathbf{z}$ are modulated by the encoded feature vector and the label vector from the X-Encoder and Y-Encoder, respectively. First, the outputs of the X-Encoder and Y-Encoder are concatenated along the time axis. Then, the concatenated features are passed to an FC layer. A self-attention layer (Vaswani et al., 2017), as described in Sec. 2.5, is added to attentively learn the interconnections of the features. After that, an LSTM with two stacked hidden layers is used for learning the temporal features into a hidden state. In the end, the hidden state is fully connected by an FC layer and then split by two FC layers side by side. The two FC layers are trained to learn the mean and variance of the distribution of the latent variables $\mathbf{z}$, respectively.

The Decoder is trained on the encoded feature vector from the X-Encoder and the latent variables. First, the encoded feature vector is concatenated with the latent variables and passed to an FC layer. Then, an LSTM with two stacked layers is used to learn the temporal dynamics. After that, two FC layers are used for fusion and dimension reduction. The Softmax activation function is added to the last FC layer for generating the probability of the interaction class at each frame. The outputs of the Decoder are the frame-wise predictions of the interaction class. In the end, a voting scheme (i. e., average voting) is used to summarize the frame-wise predictions to get the sequence-wise prediction for the interaction class.

In inference time, the interactions for unseen vehicle turning sequences are classified using the trained CVAE model. First, the object and optical flow information are encoded by the X-Encoder. A latent variable is sampled from the Gaussian distribution. Then the Decoder generates the probability of the interaction class for each sequence conditioned on the output of the X-Encoder and the latent variable sampled from the Gaussian prior. The sampling is repeated 100 times at each step in order for the Decoder to generate multiple divergent probabilities of the interaction class.

5.2.3 Baseline Model

To evaluate the performance of the proposed CVAE model, it is compared with a baseline model. The baseline model is a sequence-to-sequence encoder–decoder model that uses the same input features from the object and motion information for interaction detection, as shown in Fig. 5.7, which was developed in a previous study by Cheng et al. (2020c). The baseline model has the same structure as the X-Encoder and the Decoder that are implemented in the CVAE model (Fig. 5.5). The difference between these two models is the sample generation process. The baseline model is a discriminative model and does not use the class label information and the conditional information for learning the latent variables that mimic the stochastic behavior in vehicle–VRU interactions. Without randomly sampling from the Gaussian latent variables, the output of the sequence-to-sequence encoder–decoder model is deterministic.

5.2.4 Ablation Studies

A series of ablative models is designed to analyze the contribution of object information, optical flow information, and the self-attention mechanism. The ablative models are trained by removing one of the aforementioned parts and are compared with the complete CVAE model. Table 5.3 lists all the models with different input structures.

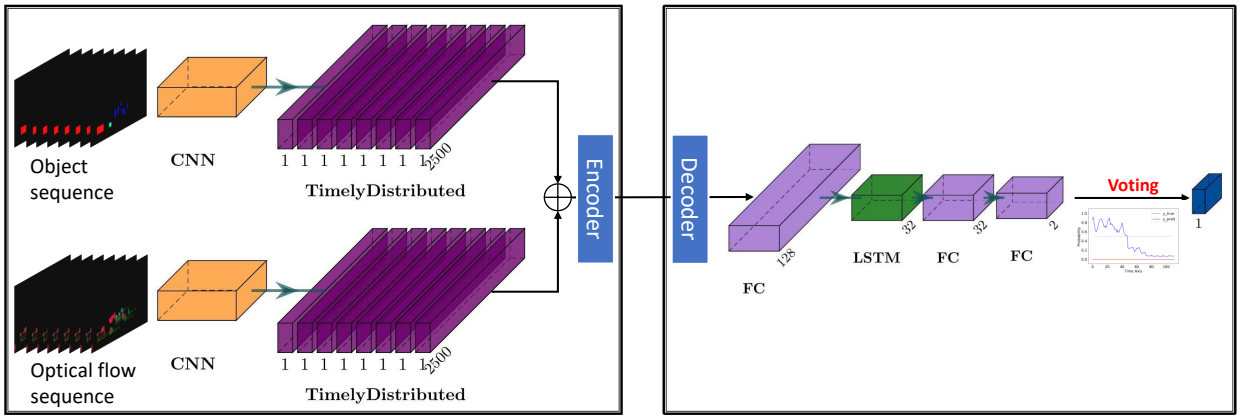


Figure 5.7: The Sequence-to-sequence encoder–decoder model for interaction detection.

Table 5.3: The models with different input structures.

Model name	Object info. (*ob*)	optical flow info. (*op*)	Self-attention (*att*)	Sample generation
[S2S+ob+op+att][1]	√	√	√	-
[CVAE+op+att]	-	√	√	√
[CVAE+ob+att]	√	-	√	√
[CVAE+ob+op]	√	√	-	√
[CVAE+ob+op+att][2]	√	√	√	√

[1]The baseline model
[2]The complete CVAE model

5.2.5 Evaluation Metrics

Accuracy, Precision, Recall, and F1-score are applied to measure the performance of interaction detection on the test data from both the KoW and NGY intersections. Tested samples are categorized according to the comparison between their ground truth class label and the predicted label, as listed in Table 5.4.

Table 5.4: Categories of tested samples.

Category name	Ground truth	Prediction
TP: true positive	interaction	interaction
TN: true negative	non-interaction	non-interaction
FP: false positive	non-interaction	interaction
FN: false negative	interaction	non-interaction

Accuracy is defined as the fraction of the number of correctly predicted samples over the total number of samples.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Precision is defined as the fraction of the number of TP samples over the number of predicted positive samples, including TP and FP samples.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall is defined as the fraction of the number of TP samples over the number of actual positive samples in the whole dataset, namely the number of TP samples plus the number of FN samples. This is a very important measurement for interaction detection. If the model fails to detect an actual interactive sequence, it might lead to a serious risk for any traffic application relying on detection performance.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} .$$

The measurements of precision and recall may represent very different performances. For example, a high value of precision does not indicate a high value of recall. Therefore, F1-score is used to provide a measurement of the overall performance of a model. It is defined as the so-called harmonic mean of precision and recall.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} .$$

5.3 Results

This section consists of three parts: the quantitative results measured by the above evaluation metrics, the qualitative results representing a typical scenario for each intersection, and finally a discussion of the results.

5.3.1 Quantitative Results

The quantitative results are summarized in Table 5.5 and 5.6 for the right-turn KoW intersection and the left-turn NGY intersection, respectively. Due to the multi-sampling process, the results of the CVAE-based models are not deterministic, hence the corresponding standard deviations are given in the tables.

Table 5.5 shows the results of the interaction detection at the right-turn intersection. (1) Both the sliding window and padding methods yield similar and very accurate results for interaction detection using the combined information from object detection and optical flow. The accuracy and F1-score are both above 0.95. (2) Compared to the baseline model *[S2S+ob+op+att]*, the proposed model *[CVAE+ob+op+att]* has a slightly better performance using the sliding window method and comparable performance using the padding method. (3) Compared to the ablative models, the combined information improves the performance using the sliding window method. However, the improvement is not obvious when using the padding method, especially compared to the ablative model that only uses object information. On the other hand, regardless of whether the sliding window or padding method is used, the ablative model that exclusively uses optical flow information only achieve an accuracy below 0.70. (4) The self-attention mechanism does not lead to an obviously better or worse performance using either the sliding window or padding method.

Table 5.5: Detection results of the right-turn intersection on the KoW dataset. Best values are highlighted in boldface.

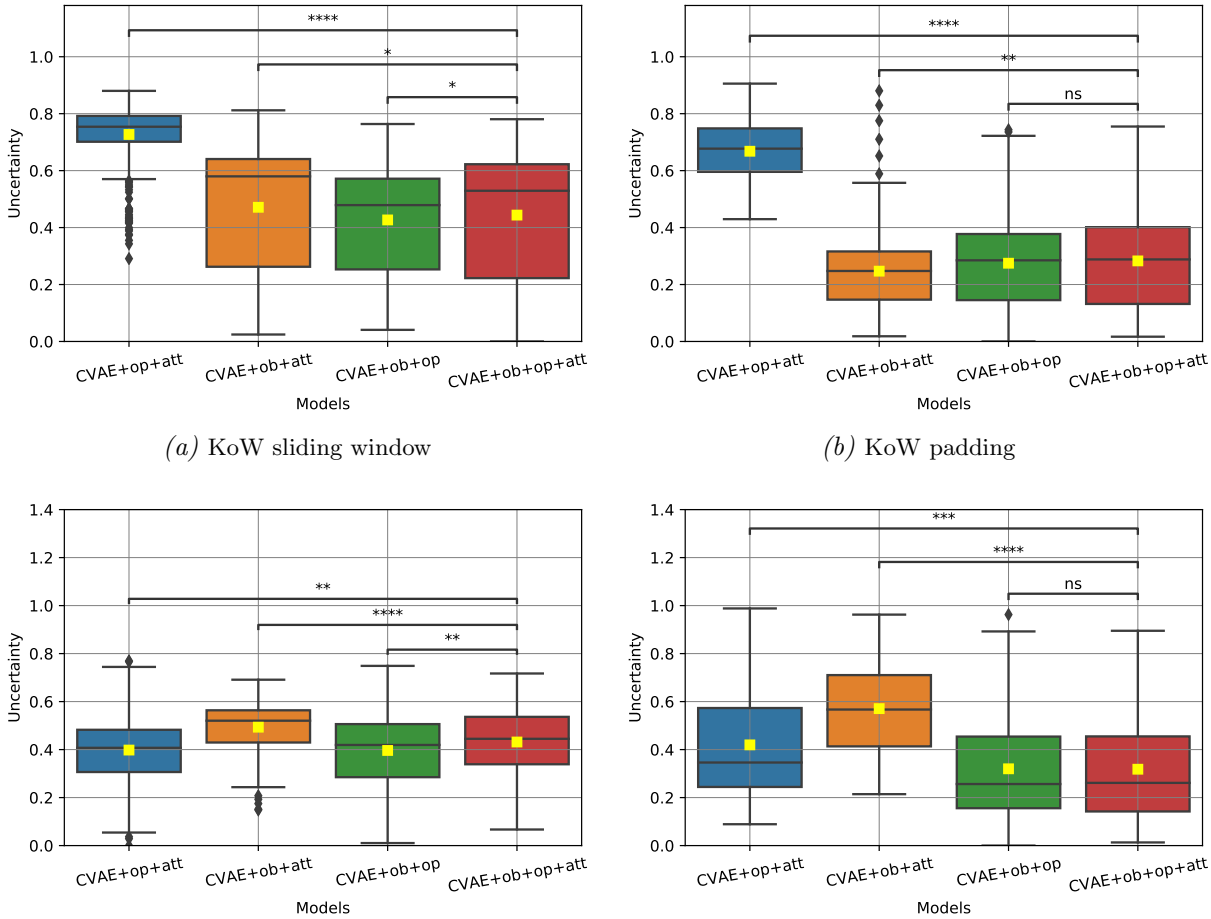
Model	shape	Accuracy	Precision	Recall	F1-score
[S2S+ob+op+att]	sliding win.	0.951	0.935	0.969	0.951
[CVAE+op+att]	sliding win.	$0.692_{\pm 0.006}$	$0.717_{\pm 0.007}$	$0.635_{\pm 0.011}$	$0.673_{\pm 0.008}$
[CVAE+ob+att]	sliding win.	$0.952_{\pm 0.002}$	$0.934_{\pm 0.002}$	$\mathbf{0.973}_{\pm 0.004}$	$0.953_{\pm 0.002}$
[CVAE+ob+op]	sliding win.	$\mathbf{0.965}_{\pm 0.001}$	$\mathbf{0.976}_{\pm 0.003}$	$0.953_{\pm 0.000}$	$\mathbf{0.964}_{\pm 0.001}$
[CVAE+ob+op+att]	sliding win.	$0.961_{\pm 0.002}$	$0.969_{\pm 0.004}$	$0.953_{\pm 0.000}$	$0.961_{\pm 0.002}$
[S2S+ob+op+att]	padding	0.963	0.944	0.984	0.964
[CVAE+op+att]	padding	$0.610_{\pm 0.008}$	$0.649_{\pm 0.011}$	$0.479_{\pm 0.012}$	$0.551_{\pm 0.010}$
[CVAE+ob+att]	padding	$\mathbf{0.967}_{\pm 0.002}$	$\mathbf{0.955}_{\pm 0.003}$	$0.980_{\pm 0.003}$	$\mathbf{0.967}_{\pm 0.002}$
[CVAE+ob+op]	padding	$0.966_{\pm 0.003}$	$0.946_{\pm 0.004}$	$\mathbf{0.989}_{\pm 0.001}$	$\mathbf{0.967}_{\pm 0.002}$
[CVAE+ob+op+att]	padding	$0.962_{\pm 0.002}$	$0.952_{\pm 0.003}$	$0.973_{\pm 0.000}$	$0.963_{\pm 0.002}$

Table 5.6 shows similar, but slightly different results of the interaction detection at the left-turn intersection. (1) Both the sliding window and padding methods yield reasonable results for interaction detection using the combined information. However, the predictions of the sliding window method are more accurate than those of the padding method. (2) Compared to the baseline model, the CVAE model using the combined information achieves better performance, especially when using the sliding window method (e. g., about 0.05 increment in F1-score). (3) Compared to the ablative models, the improvement by using the combined information can be found in both the sliding window and padding methods. (4) The best performance, especially measured by recall (0.916) and F1-score (0.892) on the NGY dataset, is achieved by the proposed CVAE model using the sliding window method with the self-attention mechanism.

Table 5.6: Detection results of the left-turn intersection on the NGY dataset. Best values are highlighted in boldface.

Model	Shape	Accuracy	Precision	Recall	F1-score
[S2S+ob+op+att]	sliding win.	0.849	0.878	0.811	0.843
[CVAE+op+att]	sliding win.	$0.878_{\pm 0.004}$	$0.854_{\pm 0.004}$	$0.912_{\pm 0.006}$	$0.882_{\pm 0.004}$
[CVAE+ob+att]	sliding win.	$0.734_{\pm 0.008}$	$0.698_{\pm 0.007}$	$0.824_{\pm 0.009}$	$0.756_{\pm 0.007}$
[CVAE+ob+op]	sliding win.	$0.882_{\pm 0.006}$	$\mathbf{0.915}_{\pm 0.004}$	$0.842_{\pm 0.009}$	$0.887_{\pm 0.006}$
[CVAE+ob+op+att]	sliding win.	$\mathbf{0.889}_{\pm 0.005}$	$0.869_{\pm 0.005}$	$\mathbf{0.916}_{\pm 0.007}$	$\mathbf{0.892}_{\pm 0.004}$
[S2S+ob+op+att]	padding	0.721	0.712	0.743	0.727
[CVAE+op+att]	padding	$0.764_{\pm 0.010}$	$0.756_{\pm 0.011}$	$0.808_{\pm 0.013}$	$0.781_{\pm 0.009}$
[CVAE+ob+att]	padding	$0.683_{\pm 0.012}$	$0.661_{\pm 0.011}$	$0.751_{\pm 0.019}$	$0.703_{\pm 0.013}$
[CVAE+ob+op]	padding	$\mathbf{0.782}_{\pm 0.007}$	$\mathbf{0.763}_{\pm 0.010}$	$\mathbf{0.819}_{\pm 0.014}$	$\mathbf{0.790}_{\pm 0.009}$
[CVAE+ob+op+att]	padding	$0.742_{\pm 0.007}$	$0.750_{\pm 0.009}$	$0.728_{\pm 0.010}$	$0.739_{\pm 0.007}$

The Kernel Density Estimation (KDE) function (Sec. 4.1.3) is used to measure the uncertainty levels of the CVAE-based models with different input structures. The uncertainties of the CVAE-based models are plotted in Fig. 5.8 and compared using the Mann-Whitney U test. Fig. 5.8a and 5.8b demonstrate that the CVAE-based model *[CVAE+op+att]* using only optical flow information generates significantly more uncertain predictions than the other models tested on the KoW dataset. This pattern is consistent with the prediction performance, according to which the model *[CVAE+op+att]* also yields less accurate predictions. A similar pattern can be observed from the CVAE-based model using only object information (Fig. 5.8c and 5.8d for the sliding and padding methods, respectively) tested on the NGY dataset. When the uncertainty level is high in the predictions, the accuracy level also drops.



(a) KoW sliding window

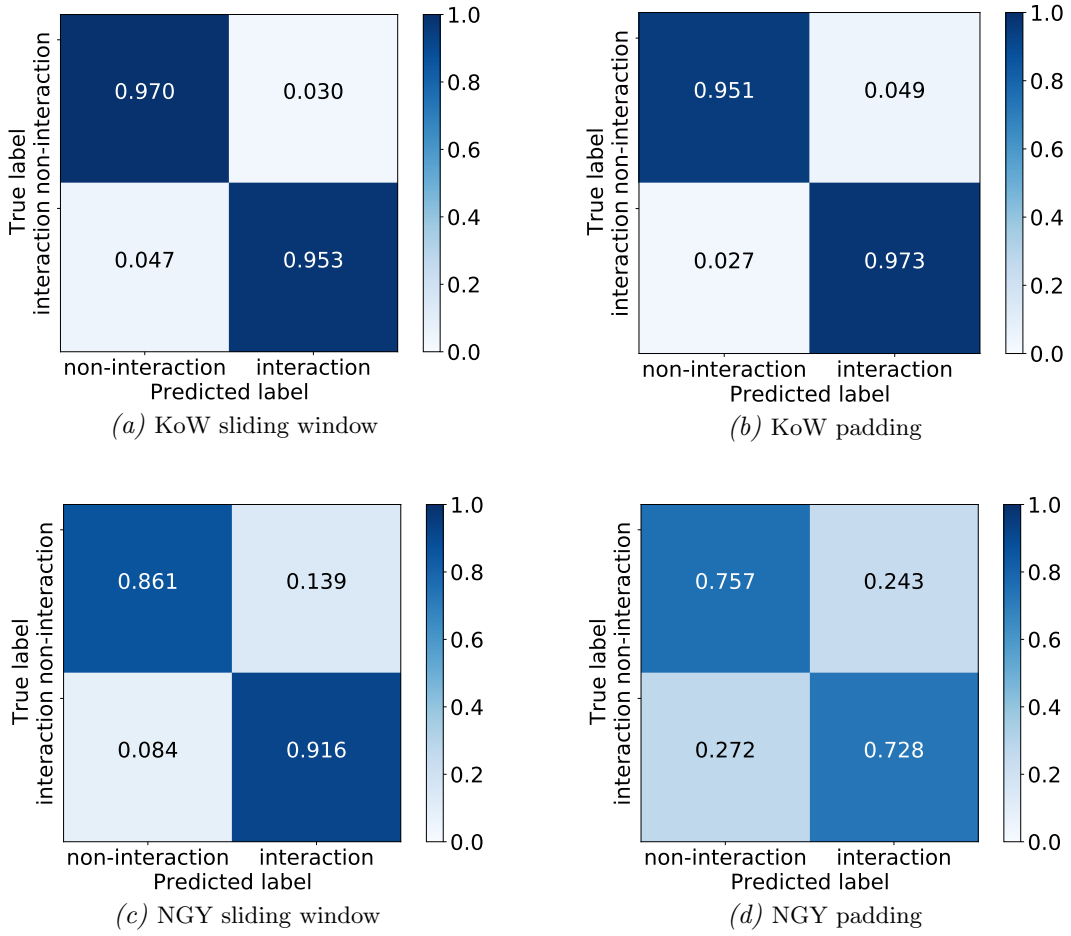
(b) KoW padding

(c) NGY sliding window

(d) NGY padding

*Figure 5.8: Uncertainty measure of the CVAE-based models tested on the KoW and NGY datasets. The mean value is denoted by the yellow square in each box-plot. The uncertainty levels across the models are compared using the Mann-Whitney U test. p-values are annotated using * or ns (not significant), where ns:* $0.05 < p <= 1.00$, **:* $10^{-2} < p <= 0.05$, ***:* $10^{-3} < p <= 10^{-2}$, ****:* $10^{-4} < p <= 10^{-3}$, *and ****:* $p <= 10^{-4}$.

The confusion matrices for the proposed CVAE model using both object and optical flow information are presented in Fig. 5.9. They show that the model using either the sliding window (true negative rate 0.970 and true positive rate 0.953) or the padding method (true negative rate 0.951 and true positive rate 0.973) achieves high performance for interaction detection tested on the KoW dataset. The proposed model achieves good performance using the sliding window method (true negative rate 0.861 and true positive rate 0.916) on the NGY dataset and maintains a relatively low false negative rate (0.084). Nevertheless the performance of the proposed CVAE model is inferior when tested on the NGY dataset compared to when it is tested on the KoW dataset. In contrast, the padding method only achieves mediocre performance (true negative rate 0.757 and true positive rate 0.728) tested on the NGY dataset. In other words, the false negative rate is relatively high (0.272).



(a) KoW sliding window

(b) KoW padding

(c) NGY sliding window

(d) NGY padding

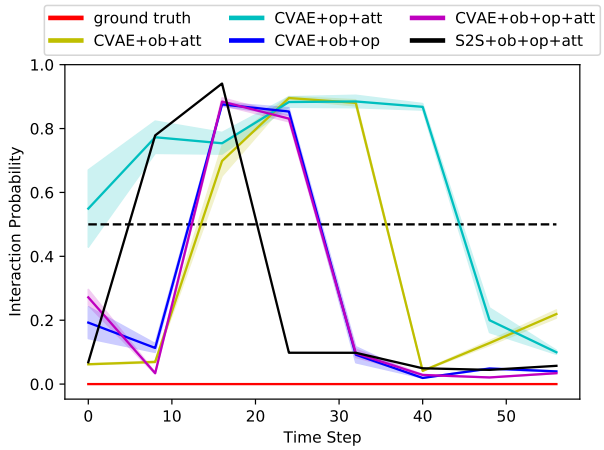
Figure 5.9: Confusion matrices for the proposed CVAE model tested on the KoW/NGY dataset using the sliding window (a)/(c) and padding (b)/(d) methods. The confusion matrices are normalized so that they can be compared across sliding window and padding methods, as well as across the datasets.

5.3.2 Qualitative Results

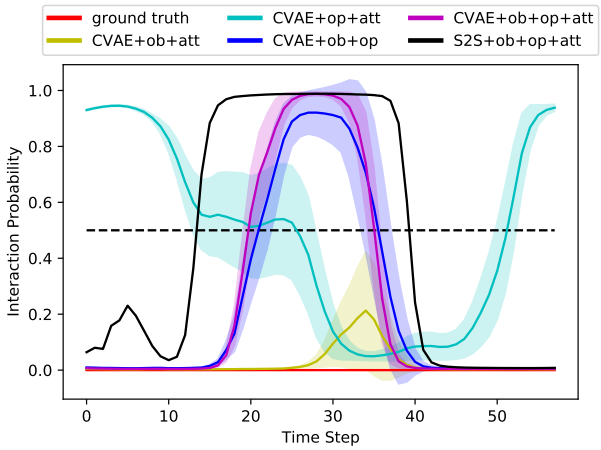
The qualitative results are presented with the aim to intuitively showcase the process of interaction detection using the aforementioned models. The fine-grained probabilities of the predicted interaction class label at each frame provide a clue of how the level of interaction intensity evolves over time.

Fig. 5.10 demonstrates a non-interaction scenario at the KoW intersection between the right-turning target vehicle (in the blue bounding box) and the still-standing pedestrian (in the red bounding box). There was no explicit interaction between them as the continuity of their behavior was not affected when the gap between them closed, so the sequence was annotated as "no interaction" required. The sequence level prediction is the average voting of the frame-level predictions. At the sequence level, all of the four models correctly predict this scenario as non-interaction using both the sliding window (Fig. 5.10a) and padding (Fig. 5.10b) methods, except the ablative model *[CVAE+op+att]* (in cyan) that only uses optical flow information. However, all the models predict a high probability of interaction when the vehicle approached the pedestrian. Also, the variance of the CVAE-based models in Fig.5.10b increases when the probabilities change from under 0.5 to a higher value of interaction. The sequence-to-sequence encoder–decoder baseline model generates a similar pattern in frame-wise predictions. But it is deterministic at each frame and does not have the mechanism required to represent the uncertainty of its predictions.

Fig. 5.11 demonstrates an interaction scenario at the NGY intersection between the left-turning target vehicle (in the blue bounding box) and the crossing cyclist (in the red bounding box). Interaction was required between them as the vehicle had to decelerate or even briefly stop, giving the way to the cyclist. All the models correctly predict this sequence as interaction using both the sliding window (Fig. 5.11a) and padding (Fig. 5.11b) methods. Similar to the scenario above, the variance of the probability for the CVAE-based models changes using the sliding window method with the modification of the distance between target vehicle and cyclist. For example, as the distance between them decreases, the probability is higher and the variance is smaller for an interaction, and vice versa. However, the variance is less visible in the padding method than the sliding window method. The ablative models based on object information using both the sliding window and padding methods have higher uncertainty levels than the other models of the frame-wise predictions.



(a) Sliding window method. The majority of frame-wise predictions of *[CVAE+op+att]* are above 0.5 and that of the other models are under 0.5



(b) Padding Method. The majority of frame-wise predictions of *[CVAE+op+att]* are above 0.5 and that of the other models are under 0.5

Time step = 0

Time step = 8

Time step = 16

Time step = 24

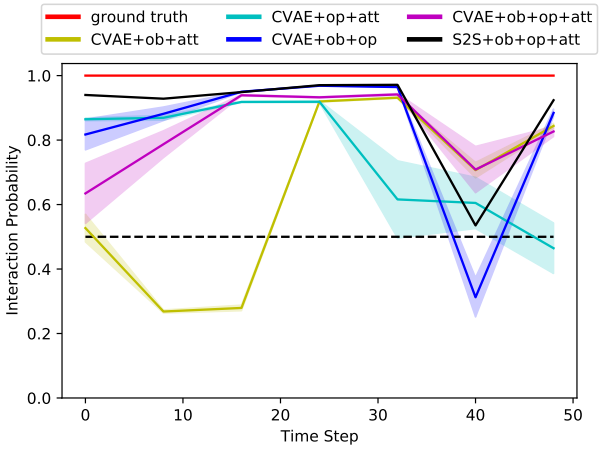
Time step = 32

Time step = 40

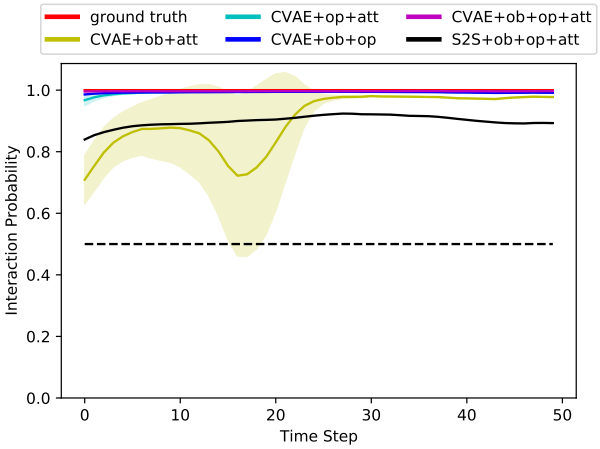
Time step = 48

Time step = 56

Figure 5.10: Examples of interaction probability at frame level using the sliding window (a) and padding (b) methods, tested on the KoW dataset. The variance of the probability by multi-sampling is visualized by the marginal shadow for the CVAE-based models. The corresponding video screenshots are aligned from upper left to lower right at the bottom with a time interval of eight frames. The target vehicle is highlighted by the blue bounding box and the still-standing pedestrian involved in the turning sequence is highlighted by the red bounding box.



(a) Sliding window method. The majority of frame-wise predictions of all models are above 0.5



(b) Padding Method. The majority of frame-wise predictions of all models are above 0.5

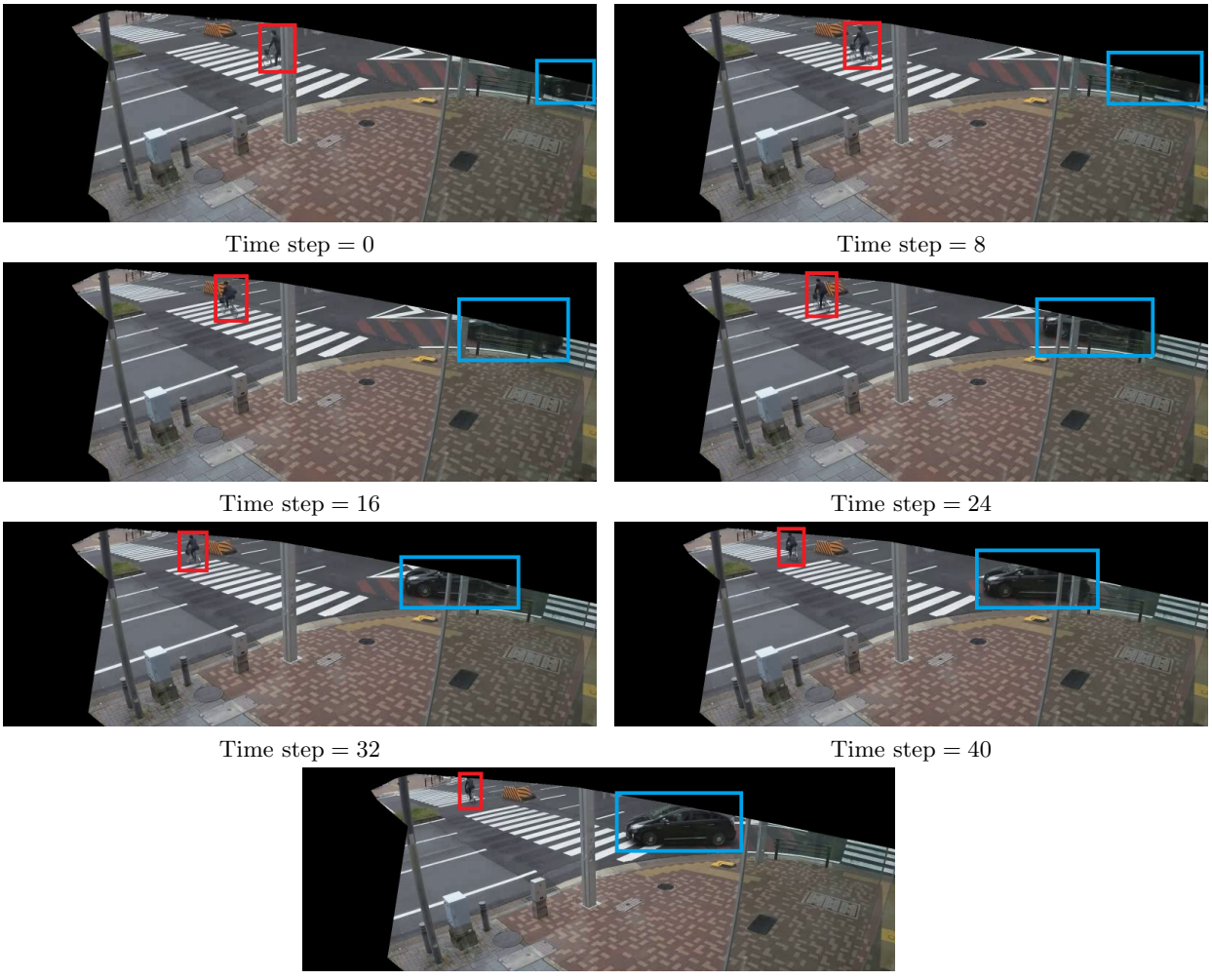


Figure 5.11: Examples of interaction probability at frame level using the sliding window (a) and padding (b) methods, tested on the NGY dataset. The variance of the probability by multi-sampling is visualized by the marginal shadow for the CVAE-based models. The corresponding video screenshots are aligned from upper left to lower middle at the bottom with a time interval of eight frames. The target vehicle is highlighted by the blue bounding box and the passing cyclist involved in the turning sequence is highlighted by the red bounding box.

5.3.3 Analysis of the Results

The results shown above are analyzed with regard to four aspects of the models: (I) the respective advantages and disadvantages of the sliding window and padding methods; (II) the difference in performance between the proposed CVAE model and the baseline model; (III) the contribution of the object information and the optical flow information via the ablative models; (IV) the impact of the self-attention mechanism.

The performances of the sliding window and padding methods are not only influenced by the size of the training data, but also the zero-padded values. The sliding window method does not depend on the sequence length, which makes it more flexible in dealing with various sequence lengths. Hence, the number of training samples was not compromised for the experiments. On the other hand, the padding method requires a pre-defined fixed sequence length, and is unable to deal with longer sequences. Hence, the number of training samples was compromised by excluding longer sequences. The impact of the training data size has been shown by the performance difference across the KoW and NGY datasets. The number of the training samples of KoW for the sliding window method is similar to the one for the padding method (Table 5.2), and their performances for interaction detection are comparable to each other (Table 5.5). In contrast, the number of training samples of NGY for the sliding window method is much larger than that for the padding method (Table 5.2). The prediction using the sliding window method is more accurate than the one obtained using the padding method (Table 5.6). In addition, the shorter sequences were padded with zeros. This poses a problem for the information extracted by optical flow, because the zero values in the optical flow feature vector represent the background of the intersection or static road users. Even though a padding mask is incorporated into the sequence for indicating the actual sequence length, the negative impact cannot be fully remedied due to the complex learning process in training. The negative impact of padded zeros from the padding method has been uncovered by the impaired performance of the ablative model *[CVAE+op+att]* compared to the one using the sliding window method.

In general, the proposed CVAE model *[CVAE+ob+op+att]* outperforms the sequence-to-sequence encoder–decoder baseline model [*S2S+ob+op+att*] quantitatively (Table 5.5 and 5.6) and qualitatively (Fig. 5.10 and 5.11). In the CVAE model, the latent variables $\mathbf{z}$ are trained to capture the stochastic attributes of road users' behavior in various traffic situations, and these variables are regularized by the Kullback-Leibler divergence loss against a Gaussian prior. In addition to the Kullback-Leibler divergence loss, the reconstruction loss is trained by minimizing the cross-entropy loss between ground truth and prediction. Optimizing these two losses together enables the CVAE model to generate divergent predictions. With the multi-sampling process of the latent variables, the predicted probabilities of interaction at each frame vary, especially when the probabilities change over time, as shown in Fig. 5.10 and 5.11; the variance of the probabilities indicates the uncertainty of the predictions. By contrast, the baseline model is trained only by optimizing the reconstruction loss. It tends to learn the "average" behavior of road users. Predictions by the baseline model are rather deterministic and there is no mechanism to interpret the uncertainty of the predictions.

The combined information of object detection and optical flow shows a stable performance for the interaction detection task. The performance of interaction detection highly depends on the quality of the input information extracted from videos, which is often impaired for many reasons. A single type of information may not be sufficient for this task. As indicated by the limited performance of the ablative models that only use optical flow information on the KoW dataset, without object information the noisy optical flow information from the through lane or padded zeros may impact detection performance negatively. Similarly, the ablative models that only use object information on the NGY dataset also achieved limited performance. The distorted object

information, especially for the road users close to the camera at the NGY dataset, could lead to wrong interaction detection. Combining the extraction techniques increases the possibility of maintaining a good quality of input information, thereby allowing the model to achieve a stable performance of interaction detection.

The self-attention mechanism does not show a consistent benefit across the datasets. Regardless of the self-attention mechanism, the CVAE-based models yield very similar results for interaction detection using both the sliding window and padding methods on the KoW dataset, and using the padding method on the NGY dataset. An improvement through the self-attention mechanism can be found for the sliding window method on the NGY dataset. Firstly, the self-attention layer is followed by an LSTM (Fig. 5.5), which may be redundant for learning the interconnections along the time axis. Secondly, the self-attention layer is likely under-trained due to the small dataset size or redundant layers, whereas the LSTM is already sufficient for learning the temporal patterns of the sequence data from the KoW intersection. On the other hand, the sequence data from the NGY intersection is more complex, e. g., featuring longer and more varying sequence lengths (see Fig. 5.2), and high traffic density. On top of the LSTM, the self-attention mechanism has turned out to be beneficial for further learning the temporal patterns.

In summary, the sliding window method is more flexible than the padding method in dealing with various sequence lengths. The CVAE-based models using the combined information of both object detection and optical flow achieve a more stable performance compared to using a single type of information. The multi-sampling process enables the CVAE-based models to mimic the uncertainty of road users' behavior, while the self-attention mechanism is only beneficial for learning temporal patterns from complex data. Overall the proposed model *[CVAE+ob+op+att]* using the sliding window method achieves a more desirable performance across the datasets.

5.4 Discussion

This section discusses the wrongly detected scenarios by the proposed CVAE model using the sliding window method, and the challenges of smoothly transferring the trained model from one intersection to the other.

5.4.1 Failed Detection

Various reasons can lead to a wrong interaction classification. Table 5.7 categorizes the wrongly detected scenarios (false negative and false positive) tested on both the KoW and NGY datasets. The false negative examples are visualized in Fig. 5.12 for the KoW intersection and in Fig. 5.13 for the NGY intersection, and the false positive examples are visualized in Fig. 5.14 for the KoW intersection and in Fig. 5.15 for the NGY intersection

The false negative scenarios are associated with VRUs entering (FN-I and FN-III) or leaving (FN-II) the intersection space. Due to their relatively long distance to the target vehicle but fast travel speed, they are erroneously classified as non-interaction.

Table 5.7: Categories of the wrongly detected scenarios by [CVAE+ob+op+att] using the sliding window method.

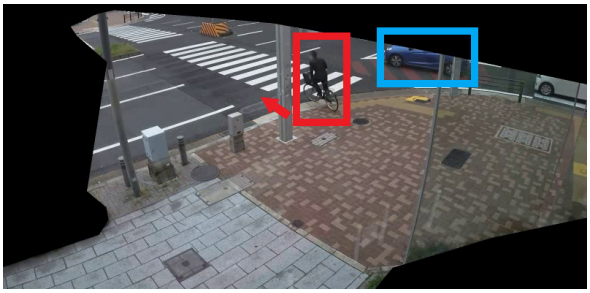
Category	Scenario description	Category	KoW	NGY
FN	pedestrian entering the intersection	(FN-I)	8	7
	pedestrian leaving the intersection	(FN-II)	1	4
	cyclist entering the intersection	(FN-III)	-	2
FP	car following	(FP-I)	4	17
	pedestrian standing on the sidewalk close to the intersection	(FP-II)	-	3
	pedestrian approaching from the sidewalk	(FP-III)	-	1
	pedestrian finishing crossing	(FP-IV)	1	1
Total*			14	35

*The total number of wrongly detected scenarios listed here is slightly different as that shown in the above confusion matrices due to the multi-sampling of the CVAE model.

FN-I FN-II

Figure 5.12: Examples of false negative detection on the KoW dataset. The right-turning target vehicles are denoted by the blue bounding boxes and the involved VRUs are denoted by the red bounding boxes.

FN-I FN-II



FN-III

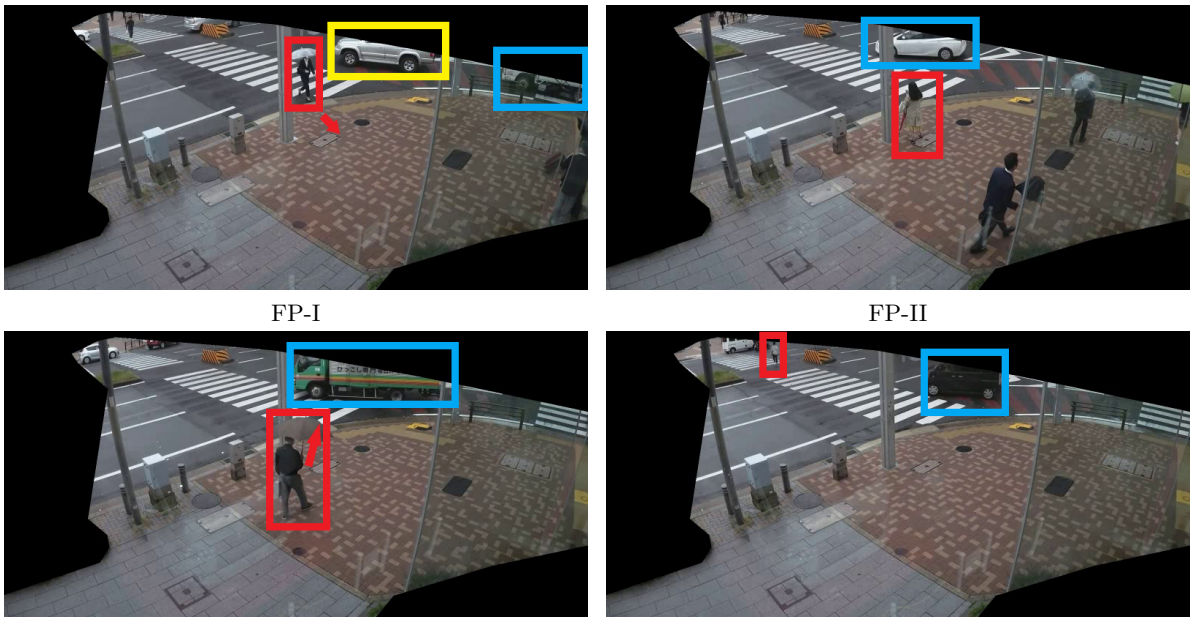
Figure 5.13: Examples of false negative detection on the NGY dataset. The left-turning target vehicles are denoted by the blue bounding boxes and the involved VRUs are denoted by the red bounding boxes.

Most of the false positive scenarios are associated with the target vehicle following a leading vehicle. As exemplified by FP-I in Fig. 5.14 and 5.15, only the leading vehicle (in the yellow bounding box) required direct interaction with the involved VRUs. After the leading vehicle finished turning, the pedestrian (in the red bounding box) also completed crossing. Afterwards, there was no interaction required from the target vehicle (in the blue bounding box) with the VRUs. However, the CVAE model has limited performance for handling this type of situation. This is because the current model does not have explicit information to differentiate the leading and target vehicles, and the model is not specifically trained for car-following situations.

In addition, a short distance from the VRUs to the intersection, e. g., standing on the sidewalk close to the intersection (FP-II Fig. 5.15) or just finishing crossing (FP-IV, Fig. 5.14 and 5.15), can also lead to a false positive case. The distortion of distance may lead to a false positive case as well. For example, in FP-III in Fig. 5.15, even though the pedestrian on the sidewalk was relatively far from the turning vehicle, it was classified as an interaction by the model due to the distorted distance at the NGY intersection. However, the camera at the KoW intersection was installed at a higher elevation than the camera at the NGY intersection. The distortion is thus less harmful for the horizontal distance. Among other reasons, this might have contributed to the better performance of the model when tested on the KoW dataset instead of on the NGY dataset.

FP-I FP-IV

Figure 5.14: Examples of false positive detection on the KoW dataset. The right-turning target vehicles are denoted by the blue bounding boxes, the leading, but not target vehicle, is denoted by the yellow bounding box, and the involved VRUs are denoted by the red bounding boxes.



FP-I FP-II

FP-III FP-IV

Figure 5.15: Examples of false positive detection on the NGY dataset. The left-turning target vehicles are denoted by the blue bounding boxes, the leading, but not target vehicle, is denoted by the yellow bounding box, and the involved VRUs are denoted by the red bounding boxes.

Based on the discussion of failed detection scenarios, the limitations of the proposed model are summarized as follows:

1) The definition of interaction only considers the relationship between the target turning vehicle and the involved VRUs. The car-following relationship is not included. Not considering this relationship often leads to false positive detection between the following car and the VRUs.

2) The crossing directions of VRUs are not used as a factor to differentiate interaction types. For example, interactions between a turning vehicle and VRUs that are approaching or leaving the crossing area from near side and far side are labeled as the same interaction type. However, the discussion above indicates that the moving directions of VRUs are important for estimating the interactions between the turning vehicle and VRUs, especially when the VRUs are leaving the intersection.

3) The exact distances between a turning vehicle and the involved VRUs are not measured, thus the change of distances between them cannot be correctly quantified. Without the measurement of distance, it is difficult for a model to distinguish the subtle difference between interaction and non-interaction. Especially, when the image distance is distorted by the camera's perspective or when an occlusion happens, the model's performance will be impaired.

5.4.2 Challenges of Cross-Dataset Generalization

The models proposed in this chapter are adapted for interaction detection at different intersections, in order to analyze the generalizability of the above models. In the previous experiment settings, all the models were trained and tested using the dataset from the same intersections. In this section, the models trained using the KoW dataset were tested on the NGY dataset, and vice versa. Frames from the test set were resized into the same size and mirrored into the same direction as the training set, so that the trained models could be tested on both datasets without changing the setting of the input size.

Table 5.8: Performance of cross-dataset validation for interaction detection on the KoW and NGY datasets.

Model	Input form	Accuracy	Precision	Recall	F1-score
Trained on the NGY dataset and tested on the KoW dataset					
[S2S+ob+op+att]	sliding win.	0.490	0.430	0.490	0.350
[CVAE+ob+op+att]	sliding win.	0.490±0.003	0.495±0.001	0.934±0.005	0.647±0.002
[S2S+ob+op+att]	padding	0.473	0.450	0.470	0.400
[CVAE+ob+op+att]	padding	0.485±0.002	0.491±0.001	0.804±0.004	0.609±0.001
Trained on the KoW dataset and tested on the NGY dataset					
[S2S+ob+op+att]	sliding win.	0.535	0.526	0.704	0.602
[CVAE+ob+op+att]	sliding win.	0.541±0.005	0.540±0.005	0.557±0.007	0.548±0.006
[S2S+ob+op+att]	padding	0.464	0.420	0.460	0.380
[CVAE+ob+op+att]	padding	0.490±0.002	0.475±0.052	0.174±0.010	0.255±0.003

Table 5.8 lists the results for the cross-dataset validation. It shows that both the CVAE-based and sequence-to-sequence encoder–decoder models do not achieve good performance either using the sliding window or padding method. This could be because these two datasets (Sec. 5.1) are very different in terms of, e. g., vehicle travel direction, camera perspective, frame size and rate, sequence length, intersection layout, traffic density, and cultural factors (Germany vs. Japan). It should be noted that because the camera parameters and reference coordinates were not available from the datasets, in this thesis projective transformation is not applied to transform the perspective to a

bird's-eye view. Under the cross-dataset validation setting, the resized frames distort the motion and position of the dynamic objects and confuse the models for predicting interactions between vehicles and VRUs. However, this leads to a very interesting future research question—how to generalize the models for different intersections and traffic, and even different cultures?

5.5 Summary

In this chapter, an end-to-end generative model based on CVAE has been proposed to automatically detect interactions between vehicles and VRUs in temporarily shared spaces of intersections using video data. All the road users that appear during a vehicle's turning time are detected by recent state-of-the-art deep learning object detectors. Simultaneously, road users' motion information is captured by optical flow. Even without tracking, object detection and optical flow together provide rich information for interaction detection. The sequence-to-sequence CVAE model using the sliding window or padding method is able to learn dynamic patterns from sequences of varying length and predicts fine-grained interaction class labels at each frame of less than 0.1 seconds. The frame-wise predictions provide a clue as to how intensity of interaction evolves as time unfolds, which in turn represents the dynamics of interaction between a turning vehicle and VRUs. The average voting scheme summarizes frame-wise predictions so as to accurately get a class label for the overall sequence. Besides, the multi-sampling process generates divergent predictions and the variance of multiple predictions reflects the uncertainty of the model's prediction. The Kernel Density Estimation function is used to quantitatively measure the uncertainty level.

The efficacy of the model was validated in different environments, a right-turn intersection in Germany and a left-turn intersection in Japan. It achieved an F1-score above 0.96 at the right-turn intersection and 0.89 at the left-turn intersection, and outperformed a sequence-to-sequence encoder–decoder model quantitatively and qualitatively. In contrast, the encoder–decoder model only outputs deterministic predictions and there is no way to interpret the uncertainty of the predictions.

A series of ablation studies investigated the effectiveness of the combined information from object detection and optical flow, and the self-attention mechanism for learning temporal patterns from complex sequences. The comparison between the sliding window and padding methods shows that the former method is more flexible in coping with sequences of varying sequence length—the number of samples is not restricted to the maximum sequence length that a model can handle, which stands in contrast to the padding method. The self-attention mechanism has only shown a clear positive effect for interaction detection on the complex NGY dataset.

To summarize, the proposed model can be applied for automatically learning interactions between vehicles and VRUs at a fine-grained time interval in temporarily shared spaces of intersections with busy traffic.

In future work, several improvements can be made to reduce the limitations of this detection model. First, the dichotomous classification of interaction should be extended to multi-class classification, e. g., taking the confrontation direction and car-following relationship into consideration. Second, the accuracy of feature extraction can be enhanced by using multiple cameras or even tracking. Third, projective transformation techniques or data recorded by drones with a bird's-eye view can be explored to reduce the distortion caused by the camera's perspective and filter the noisy optical flow information captured from the through lane next to the turning lane. Last but not least, the generalizability of the model for interaction detection at different intersections needs to be studied further.

6 Trajectory Prediction

This chapter proposes three frameworks based on Conditional Variational Auto-Encoder (CVAE) (Chapter 4) for multi-path trajectory prediction conditioned on relevant internal and external stimuli that influence an agent's motion behavior. As discussed in Sec. 4.2, the internal stimulus mainly focuses on physical motion of the target agent and the external stimulus considers two aspects: agent-to-agent interaction and scene contexts for agent-to-environment interaction.

The first framework (Sec. 6.1), named Multi-Context Encoder Network (MCENet), focuses on investigating multiple contexts, i. e., pedestrian grouping and three types of scene context for trajectory prediction in share spaces. Given the fact that scene context is space dependent, the framework is mainly evaluated using the test sets that have the same scene context as the training sets. Its limited generalizability in new scene contexts using the so-called leave-one-out cross validation is discussed as well in Sec. 6.1.4.

The second framework (Sec. 6.2), named Attentive Maps Encoder Network (AMENet), investigates a novel method with a self-attention mechanism for modeling agent-to-agent interaction and focuses on predicting trajectories in various unseen shared spaces that the model has not been trained on. Compared to MCENet, besides considering the motion of the target agent and agent-to-agent interaction with grouping context as the standard setting, this framework does not use any scene context information in order to maintain a good performance with respect to generalizability.

The last framework (Sec. 6.3), named Dynamic Context Encoder Network (DCENet) extends the self-attention mechanism to model agent-to-agent interaction. Regarding the input information, it is closely related to the second framework, AMENet, i. e., both AMENet and DCENet focus on exploring dynamic information (motion and agent-to-agent interaction), but scene context information is not considered. DCENet is designed to improve the performance of AMENet in shared spaces. Its generalizability is further evaluated in intersections of mixed traffic.

Table 6.1 provides a summary of the datasets and input information of the three frameworks. More details are given in the following sections for each framework.

Table 6.1: Summary of the datasets and input information of the trajectory prediction frameworks.

Dataset	**Gates3**	**HBS**	**HC**	**TrajNet***	**inD***	**Input information**		
& **Input**	shared s.	shared s.	shared s.	shared s.	intersection	target-motion	agent-to-agent	scene-context
MCENet	√	√	√			√	√	√
AMENet				√		√	√	
DCENet				√	√	√	√	

*benchmarks that contain multiple datasets

6.1 Multi-Context Encoder Network

The results presented in this Sec. 6.1 are adapted from (Cheng et al., 2020b), published in the proceedings of the 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC).

6.1.1 Framework

The *Multi-Context Encoder Network* (MCENet) predicts multi-path trajectories of heterogeneous agents by introducing grouping and scene contexts. Agent-to-agent interaction is modeled by an occupancy grid map (Sec. 4.2.3.2). Fig. 6.1 presents the pipeline for multi-context trajectory prediction.

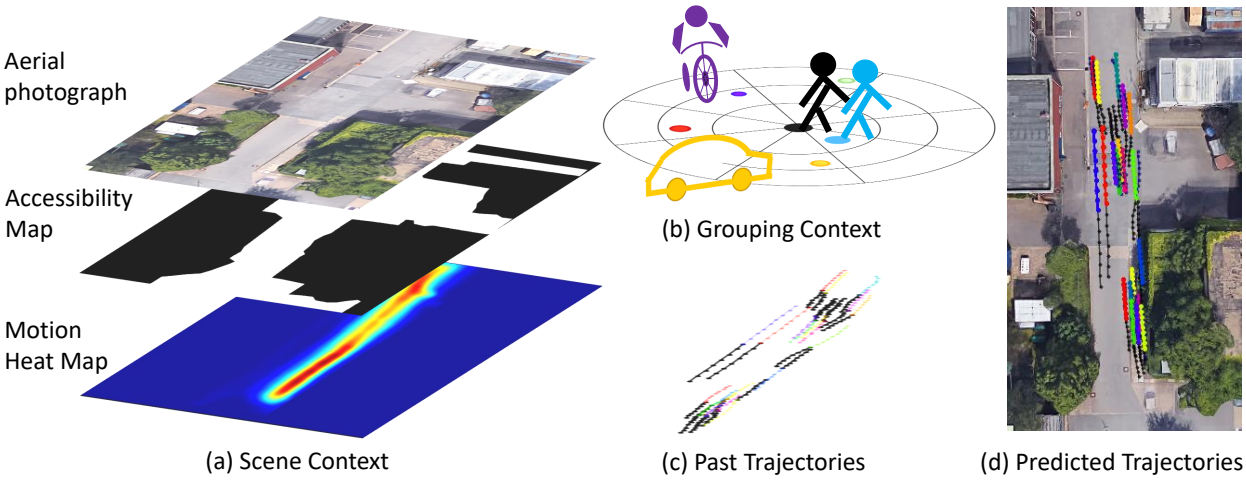


Figure 6.1: The pipeline for multi-context trajectory prediction. Scene context (a), grouping context (b), and past trajectories (c) are considered jointly to predict future trajectories for each agent in a shared space. The aerial photograph is taken from Imagery ©2020 Google.

The following aspects are investigated by MCENet for trajectory prediction in shared spaces:

1) **Grouping context.** Each agent's behavior is affected by neighboring agents. A target agent has social connections to the agents in the same group, but it has to avoid collisions with other agents. Distinguishing the group and non-group agents of agent-to-agent interaction for the target agent is useful for analyzing its motion (more details presented in Sec. 4.2.3.2).

2) **Scene context.** Agents' behavior is constrained by the scene context of the environment, such as space layout and arrangement of buildings, especially in a shared space. To explore the effect of the scene, three types of scene context are studied in this framework: *motion heat maps* describe the prior of how different agents move; an *aerial photograph image* provides global visual information of the scene; and *accessibility maps* define the accessible areas respective to the agents' transport mode (more details presented in Sec. 4.2.3.3).

3) **Multi-path trajectories.** Given a segment of a past trajectory, there is more than one plausible future trajectory. This framework predicts multiple plausible and socially-acceptable trajectories.

4) **Heterogeneous road agents.** Pedestrians, cyclists, and vehicles are taken into account together, rather than focusing on a specific type of agent (pedestrians or cars).

An overview of the framework is given in Fig. 6.2. MCENet consists of two encoders and a decoder: one encoder is trained to encode the past information from observation that includes the motion, interaction, and context information of a target agent; analogously, another encoder is trained to encode the future information from the ground truth. Then, the two types of encoded information are fused and forwarded to learn a latent space that describes the distribution of future trajectories. The decoder is trained to predict multi-path trajectories of a target agent depending on its past information and a set of stochastic latent variables that are sampled from the learned latent space. The following explains each module presented in Fig. 6.2 in detail.

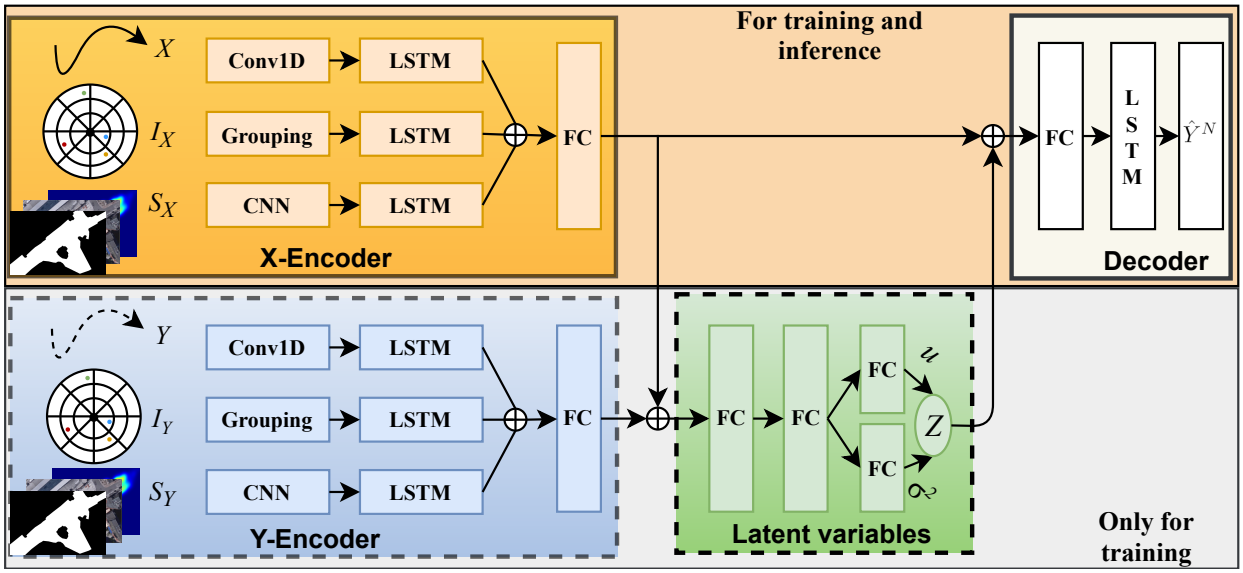


Figure 6.2: The framework of the multi-context encoder network. Conv1D stands for 1D convolutional layer, CNN stands for convolutional neural network, LSTM stands for long short-term memory, and FC stands for fully connected layer. X and Y denote the respective motion input from observation and ground truth. I and S denote respective interaction information and scene context information with the subscript associated with observation (X) or ground truth (Y).

The X-Encoder and Y-Encoder encode the observed and the ground truth information in parallel, in order to consider the motion, interaction, and environment contexts along the complete time horizon. The X-Encoder is used to encode the observed information. First, a 1D convolutional layer (Conv1D) is used to learn motion features along the time axis of the trajectory input, which is characterized by a sequence of offsets and the one-hot encoding of the target agent's transport mode (Sec. 4.2.3.1). The grouping algorithm described in Sec. 4.2.3.2 is used to differentiate group and non-group neighboring agents, and only the non-group neighboring agents are stored in the occupancy map for learning the agent-to-agent interaction. A three-layer CNN is used to extract scene contextual features from one of the three types of scene context mentioned above. Alternatively, rather than training the CNN from scratch, the feature extractor can also be substituted by a pre-trained network, e. g., MobileNet (Howard et al., 2017). Then, the features of these three branches are fed to different Long Short-Term Memories (LSTMs) for learning the hidden information at each step. The outputs of the LSTMs are concatenated and passed through a fully connected (FC) layer followed by the ReLU activation for feature fusion. The Y-Encoder is used to encode the ground truth information and works in the same way as the X-Encoder.

During training, the encoded observed and ground truth representations are concatenated and forwarded to two FC layers followed by the ReLU activation. The following two FC layers are trained to learn the mean and variance of the latent variables $\mathbf{z}$. In the end, the output of the X-Encoder and the latent variables $\mathbf{z}$ are concatenated and fed to the Decoder for reconstructing

Y. The Decoder consists of an FC layer for fusion and dimension reduction and one LSTM for sequentially predicting future trajectories.

During inference, the Y-Encoder is removed (no ground truth information) and the observed information is encoded and fused by the trained X-Encoder in the same way as in the training phase. To generate a future prediction sample, a latent variable z is sampled from the Gaussian prior $\mathcal{N}(0, \mathbf{I})$ and concatenated with the output of the X-Encoder as input of the Decoder. This step is repeated N times to generate N samples of future trajectories.

6.1.2 Experiments

6.1.2.1 Datasets

The model is validated on three shared space datasets, i. e., Gates3 (Robicquet et al., 2016), HBS (Pascucci et al., 2017) and HC (Cheng et al., 2019). All these datasets have a very distinct space layout and scene context. Gates3 is one of the most challenging subsets of the Stanford Drone Dataset (Robicquet et al., 2016). It was captured at a very busy roundabout shared by pedestrians and cyclists on the Stanford University campus. HBS was recorded near a busy train station in the German city of Hamburg with pedestrians crossing among vehicles and cyclists. HC was acquired over a street with buildings and trees on both sides on the Leibniz University Hannover campus. Table 6.2 lists the statistics of the datasets.

Each dataset was split into training (last 70 % of the total steps) and test (first 30 %) subsets. Conventionally, eight steps of past trajectories are taken as observation and the following eight steps are predicted. Longer term prediction is possible, but 2.4 seconds are sufficient for most humans to respond to an emergency braking (Taoka, 1989). Hence, only the performance for predictions of the next four seconds is reported.

Table 6.2: The datasets for testing MCENet.

Name	#ped.	#cyc.	#veh.	#frames.	Frame rate	Description
Gates3 (Robicquet et al., 2016)*	159	223	0	9.9k	2 fps	a shared roundabout
HBS (Rinke et al., 2017)	115	22	338	3.6k	2 fps	a shared area
HC (Cheng et al., 2019)	384	42	13	3.5k	2 fps	a shared street

*a few unfeasible trajectories measured with unlikely acceleration were discarded

6.1.2.2 Evaluation Metrics

Average displacement error (ADE) and final displacement error (FDE) are the most commonly applied metrics to measure the performance of trajectory prediction (Alahi et al., 2016; Gupta et al., 2018; Sadeghian et al., 2019). The mean values of ADE and FDE over all the predicted trajectories are reported as the overall errors of prediction.

ADE is the aligned Euclidean distance from the predicted trajectory $\hat{Y}$ to its corresponding ground truth trajectory Y averaged over all the steps, denoted by Eq. (6.1).

$$\mathrm{ADE} = \frac{1}{T'} \sum_{t'=1}^{T'} \sqrt{\left(\hat{Y}_{t'} - Y_{t'}\right)^2}, \tag{6.1}$$

where $t' \leq T'$ and T' is the last step of the prediction.

FDE is the Euclidean distance of the last position from Y to the corresponding $\hat{Y}$, denoted by Eq. (6.2). It measures a model's ability to predict the destination of a trajectory and is more challenging as errors accumulate over time.

$$\text{FDE} = \sqrt{\left(\hat{Y}_{T'} - Y_{T'}\right)^2}. \tag{6.2}$$

The mostlikely prediction and the best prediction @*top*10 are employed to evaluate the performance of multi-path trajectory prediction.

- The mostlikely prediction is selected by the trajectory ranking method using a bivariate Gaussian distribution as described in detail in Sec 4.2.2.
- The @*top*10 prediction is the best one out of ten predicted trajectories that has the smallest ADE and FDE compared to the ground truth.

6.1.2.3 Compared Models and Ablation Studies

To quantify the efficacy of MCENet, it is compared with three state-of-the-art[1] deep learning models.

- S-LSTM (Alahi et al., 2016) proposes a social pooling layer for agent-to-agent interaction modeling, in which a rectangle occupancy map is used to pool the existence of near neighboring agents at each step.
- S-GAN (Gupta et al., 2018) applies GAN (Goodfellow et al., 2014) for multiple future trajectories generation. It models agent-to-agent interaction via the hidden states of all the neighboring agents.
- SS-LSTM (Xue et al., 2018) employs an encoder–decoder structure. The encoder uses different LSTMs to encode the motion input, agent-to-agent interaction input, and agent-to-environment interaction input.

A series of ablation studies is designed to analyze the impact of each context, i. e., grouping context and three types of scene context. The ablative models of MCENet that use some or all of the above contexts are listed in Table 6.3. The ablative model that only takes motion input and interaction input with no grouping context is treated as the baseline model. In order to guarantee a fair comparison, all the models are trained using the same data and tested in the same real-world scenarios.

Table 6.3: The ablative models of MCENet with different input combinations.

Model name	**Motion input**	**Interaction input**		**Scene input**		
		interaction	grouping	heat map	aerial photograph	accessibility map
Baseline	√	√				
MCENet+gp	√	√	√			
MCENet+hm	√	√		√		
MCENet+hm+gp	√	√	√	√		
MCENet+ap+gp	√	√	√		√	
MCENet+am+gp	√	√	√			√

[1] at the time of submission of (Cheng et al., 2020b) to the ITSC conference.

The detailed settings of the experiments can be found at the open-source project repository: `https://github.com/haohao11/MCENET`.

6.1.3 Results

6.1.3.1 Quantitative Results

Table 6.4 lists the results measured by ADE/FDE of the MCENet models and the three compared state-of-the-art models tested on the aforementioned datasets. It is shown that in most cases the MCENet models (with different scene contexts plus grouping context) outperform the other methods with respect to predicting the mostlikely trajectories. Only on Gates3, SS-LSTM performs slightly better when measured by FDE. The better overall results reported by the MCENet models prove that grouping context and scene context are beneficial for trajectory prediction. The MCENet models are able to learn useful information from them effectively. In addition, the MCENet models report superior results with respect to the best predictions (@*top*10). This demonstrates that predicting multiple plausible trajectories is necessary and helpful to analyze how agents behave in the future. It is worth noting that the baseline model reports comparable results with the state-of-the-art methods. This indicates that the baseline model is able to predict accurate future trajectories even though based only on the past motion information and interaction information without considering either grouping or scene context.

Table 6.4: The results of the MCENet models and the compared models tested on the shared space datasets. ADE/FDE errors are denoted in meters and best values are highlighted in boldface.

Data	HBS	HC	Gates3	Avg.
S-LSTM	1.67/3.03	1.11/1.98	3.63/6.56	2.14/3.86
S-GAN	1.45/2.86	0.97/1.67	2.98/5.42	1.80/3.32
SS-LSTM	0.82/1.75	0.49/0.79	1.61/**2.89**	0.97/1.81
mostlikely predictions				
Baseline	0.77/1.80	0.49/0.80	1.54/3.04	0.93/1.88
MCENet+gp	0.77/1.80	**0.47**/0.77	1.50/2.98	0.91/1.85
MCENet+hm	0.76/1.77	**0.47**/0.77	1.52/2.99	0.92/1.84
MCENet+hm+gp	**0.71**/**1.68**	0.48/0.79	1.50/3.01	**0.89**/1.83
MCENet+ap+gp	0.74/1.76	**0.47**/0.76	**1.48**/2.91	0.90/**1.81**
MCENet+am+gp	0.80/1.86	**0.47**/**0.75**	**1.48**/2.98	0.92/1.86
Best predictions (@*top*10)				
Baseline	0.57/1.28	0.47/0.77	1.19/2.26	0.74/1.44
MCENet+gp	0.56/1.26	0.45/0.72	1.17/2.34	0.73/1.44
MCENet+hm	0.57/1.26	0.45/0.73	1.20/2.24	**0.72**/1.41
MCENet+hm+gp	**0.55**/**1.20**	0.46/0.73	1.18/2.26	0.73/**1.40**
MCENet+ap+gp	**0.55**/1.25	0.45/0.72	1.21/2.30	0.74/1.42
MCENet+am+gp	0.60/1.32	**0.44**/**0.70**	**1.15**/**2.22**	0.73/1.41

The results of the ablation studies are given in the lower part of Table 6.4. It is shown that the models utilizing grouping and scene contexts simultaneously achieve better overall performance than the models that partially consider grouping (MCE+gp) or scene context (MCE+hm). In terms of different scene contexts, the models using heat maps and accessibility maps achieve comparable or higher performance than the model using an aerial photograph. This indicates that the motion prior of road agents represented by the heat maps is useful for predicting the agents'

future movements, and the accessibility maps provide explicit information about accessible areas with respect to the agents' transport mode.

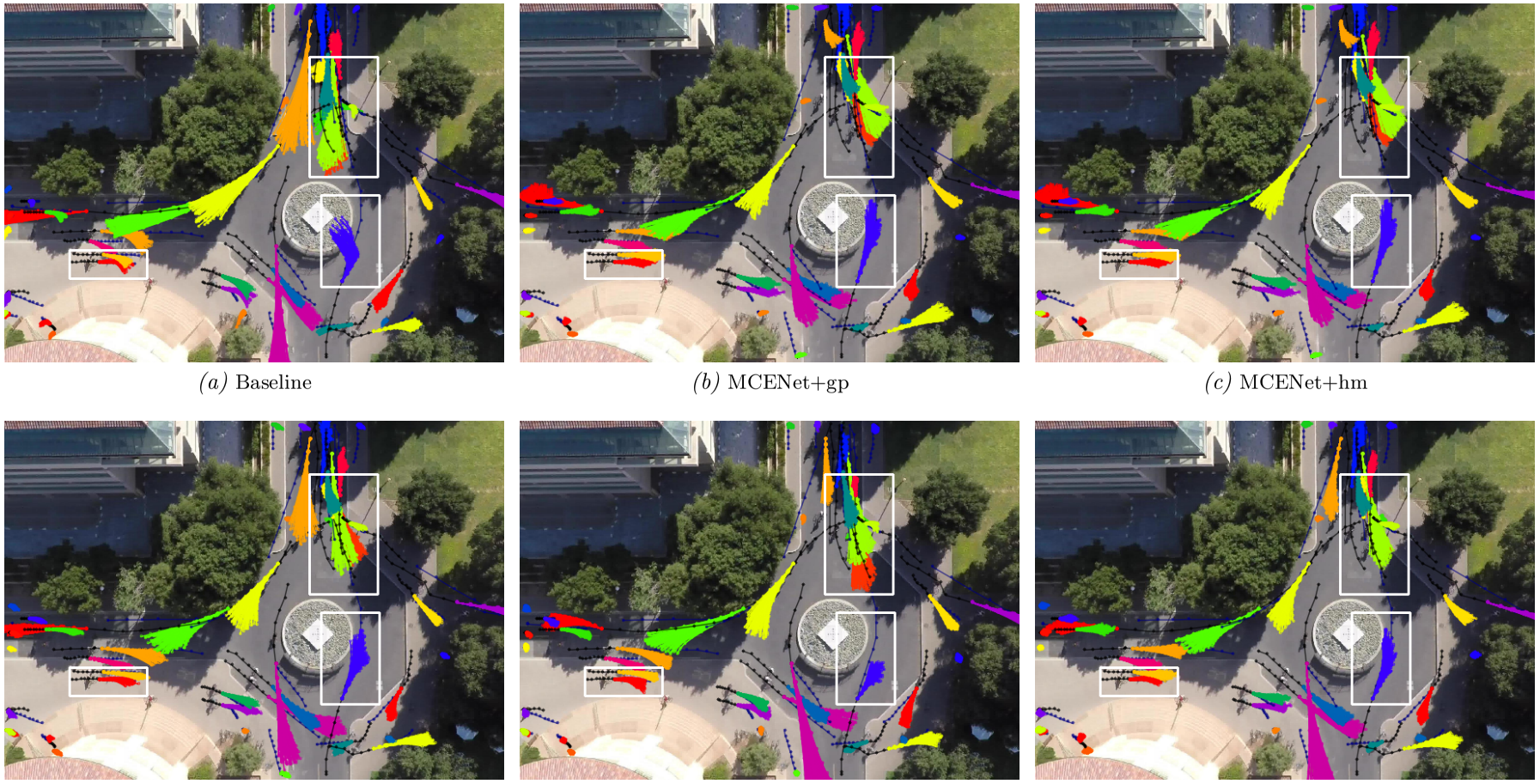
6.1.3.2 Qualitative Results

Fig. 6.3 and 6.4 show the qualitative results output by the MCENet models for multi-path trajectory predictions in the three shared spaces.

The impact of different contexts on predicting trajectories in the roundabout Gates3 is visualized in Fig. 6.3. Each sub-figure represents the utility of different contexts and the salient differences are highlighted by the white boxes. It is shown that using grouping context (Fig. 6.3b) is helpful for converging the predicted multiple trajectories into a smaller intended area (the "fan" shaped area) compared to the baseline model (Fig. 6.3a). Using heat maps, the MCENet model generates predictions that are more accurately aligned with the ground truth. For instance, the trajectories of the blue agent (in the middle box) are incorrectly predicted towards the center of the roundabout by the baseline model (Fig. 6.3a). When utilizing heat maps (Fig. 6.3c), however, the predicted trajectories follow along the road and fit the ground truth. This indicates that the prior information captured by the heat maps is important for predicting trajectories, especially in the scene with complex interactions. When the scene context of heat maps is combined with grouping context (Fig. 6.3d) the prediction results are improved further.

The second row of Fig. 6.3 shows the impact of different types of scene context on the MCENet models. Fig. 6.3e shows the prediction by MCENet+ap+gp considering the scene context from an aerial photograph. In comparison to the model not using this information (Fig. 6.3b), the aerial photograph provides a global visual context of the scene and improves the prediction. This indicates that MCENet+ap+gp is able to extract useful information directly from the RGB image to help predict trajectories. On the other hand, compared to the predictions using the scene contexts provided by respective heat maps (Fig. 6.3d) and accessibility maps (Fig. 6.3f), the improvement due to the aerial photograph is smaller, which is also in line with the quantitative results reported in Table 6.4. This is because the scene context in an RGB image is implicit while heat maps and accessibility maps provide explicit scene contexts. However, from the perspective of data acquisition, an RGB image is easier to be acquired than heat maps and accessibility maps, especially in complex scenes. Furthermore, it is apparent that predicted trajectories generated with the use of accessibility maps (Fig. 6.3f) have less divergence than the ones with heat maps (Fig. 6.3d). This is because the accessibility maps have strong constraints on how an agent should behave in a given scene, while the heat maps only provide statistical prior information about how previous agents have interacted with the environment.

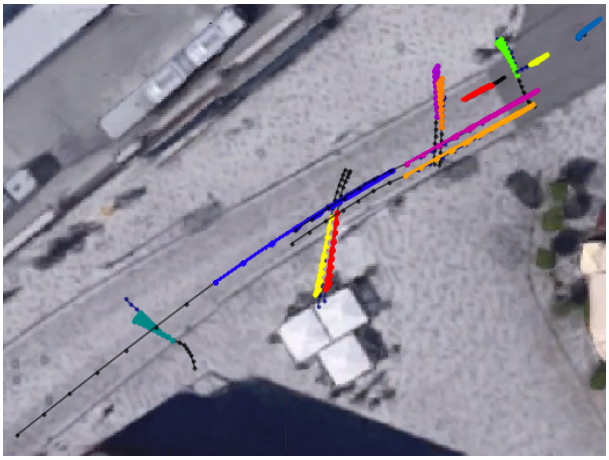
Fig. 6.4 demonstrates MCENet+am+gp that considers the contexts of accessibility maps and grouping across different spaces. It is shown that the model is able to precisely predict the future trajectories of different agents (denoted by different colors) by observing their past trajectories (in black). In environments with clearly defined orientation of the road geometry, i. e., HBS and HC, the multiple predictions of each trajectory deviate at a very small scale, i. e., most predictions are very close to the ground truth trajectories. In Gates3, instead of straight road geometry, the roundabout environment leads to more complicated agent-to-agent and agent-to-environment interactions. Even though MCENet+am+gp is able to correctly predict the trajectory for each agent, the multiple predictions of each trajectory deviate more widely, especially with respect to entering or leaving the roundabout. This implies that the ability of predicting multiple plausible trajectories is important for this task, because the uncertainty of an agent's movement increases over time in a complex environment.



(a) Baseline *(b)* MCENet+gp *(c)* MCENet+hm

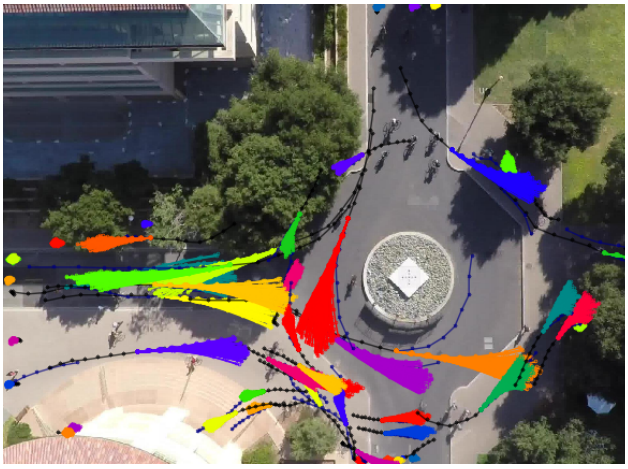
(d) MCENet+hm+gp *(e)* MCENet+ap+gp *(f)* MCENet+am+gp

Figure 6.3: The qualitative results of MCENet tested on the Gates3 dataset. Past trajectories are denoted in black and ground truth trajectories in purple. Different agents are denoted in different colors. Important differences are highlighted in the white boxes. Background images are retrieved from (Robicquet et al., 2016).



(a) HBS

(b) HC



(c) Gates3

Figure 6.4: The qualitative results of MCENet considering the context information of grouping and accessibility maps across different shared spaces. Past trajectories are denoted in black and ground truth trajectories in purple. The background images of (a) and (b) are taken from Imagery ©2020 Google, and (c) is retrieved from (Robicquet et al., 2016).

6.1.4 Discussion

The above experiments and analyses for trajectory prediction are limited to the same individual shared spaces, i. e., all the MCENet models are trained and tested in the same environment. Therefore, the models' generalizability still has to be discussed. The scene contexts in HC, HBS, and Gates3 are totally different from each other for heterogeneous road users (as shown in Fig. 6.4). Can the MCENet models being trained by, e. g., the HC and HBS datasets be generalized to the Gates3 dataset?

To answer the question above, the so-called leave-one-out cross validation is applied: the MCENet models were trained using two of the above datasets and tested on the third unseen dataset. This operation was repeated for each dataset and the average performance is reported below. Furthermore, the trained models were fine-tuned via retraining using a gradually increased amount of the data taken from the tested space. The amount of data for fine-tuning is quantified by *visibility*. If no data from the tested space is leveraged, the visibility rate is zero. Up to 30 % (0.3 visibility rate) of the data from the tested space was used for fine-tuning and the remaining 70 % of the data was set as the independent test set.

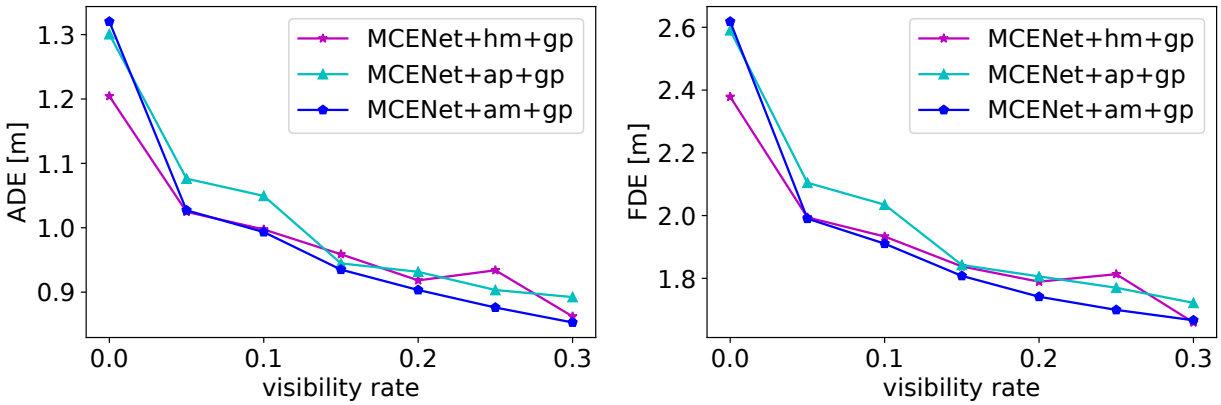


Figure 6.5: The performance of the MCENet models measured by leave-one-out cross validation.

Fig. 6.5 shows the results for the MCENet models that use different scene contexts. It is shown that with zero visibility of the tested space, the performances of all the MCENet models drop considerably compared to what is shown in Table 6.4. This is a reasonable phenomenon because different spaces have different scenarios. Scene information is an important factor for the MCENet models. The models trained in the other spaces have no knowledge about the scene information of the tested space without fine-tuning. Therefore, the learned scene information does not match the one of the test set.

Fig. 6.5 also demonstrates that with an increasing visibility rate for fine-tuning, the models' performances improve significantly. The improved performances indicate that the MCENet models can be easily transferred to a new space through fine-tuning. The fine-tuning method, to some extent, remedies the problem of overfitting the MCENet models to particular contexts and transfer the models to "new" contexts. Nevertheless, this method is unpractical if the data from the tested space is not available, which is often the case in reality. It remains an open question how to improve the models' generalizability without fine-tuning. This question will be further investigated by the other two frameworks in the following sections.

6.1.5 Summary

In this section, a novel framework MCENet for multi-path trajectory prediction of heterogeneous road user agents in shared spaces has been proposed. The framework incorporates scene context, interaction context, and motion information to capture the variations of future trajectories by learning a set of stochastic latent variables. Multi-path trajectories are predicted depending on the past information of the target agents by introducing the stochastic latent variables. The efficacy of the framework was investigated in several complex real-world scenarios that showed a clear improvement as measured by the average and final displacement errors over three recent state-of-the-art models.

The ablation studies helped to analyze the impact of grouping and scene contexts for trajectory prediction. The studies suggest that the MCENet model that considers the grouping context predicts more accurate trajectories, and multiple predictions for each trajectory converge into a smaller intended area than the predictions by the model without the grouping context.

Furthermore, the impact of the three types of scene context, i. e., heat map, aerial photograph, and accessibility map, has been measured. Comparison of the three scene types suggests that it is more difficult to learn scene context from an aerial photograph than from heat maps and accessibility maps. An aerial photograph provides implicit scene information by an RGB image for all agents without the differentiation of transport mode, whereas heat maps and accessibility maps provide more explicit information, i. e., the distributions of the past trajectories and the accessibilities of the road surface according to the agents' transport mode.

In the end, the generalizability of the framework was discussed. On the one hand, the leverage of the scene contexts improves the model's performance of trajectory prediction in the same scene for which the model was trained. On the other hand, the specific scene context impairs the model's generalizability for predicting trajectories in a new scene, on which the model was not trained.

6.2 Attentive Maps Encoder Network

The results presented in this Sec. 6.2 are adapted from (Cheng et al., 2021c), published in the ISPRS Journal of Photogrammetry and Remote Sensing.

On the basis of the previous framework MCENet (Sec. 6.1.1), the second framework is introduced in this section, with the aim to investigate the generalizability with respect to predicting trajectories in "new" scenes.

6.2.1 Framework

The *Attentive Maps Encoder Network* (AMENet) predicts multi-path trajectories for heterogeneous agents by introducing dynamic maps and incorporating the self-attention mechanism (Vaswani et al., 2017) for modeling agent-to-agent interaction. To gain more detailed insight, the following aspects are further explored by AMENet for trajectory prediction in shared spaces:

1) **Attentive dynamic maps.** This interaction module learns spatio–temporal interconnections between agents. It encodes the information extracted from the neighboring agents' orientation, speed, and position in relation to the target agent at each step. The self-attention mechanism enables the module to automatically focus on the salient features extracted over different steps.

2) **Generalizability.** AMENet is designed to predict heterogeneous road users' trajectories in various unseen environments in real-world mixed traffic, which addresses the open question of generalizability of MCENet (as discussed in Sec. 6.1.4).

Fig. 6.6 gives an overview of the framework. Two encoders learn the representations of an agent's behavior into a latent space: the X-Encoder learns the information from the observed trajectories, while the Y-Encoder learns the information from the future trajectories of the ground truth and is removed in the inference phase. The Decoder is trained to predict the future trajectories conditioned on the information learned by the X-Encoder and the representations sampled from the latent space.

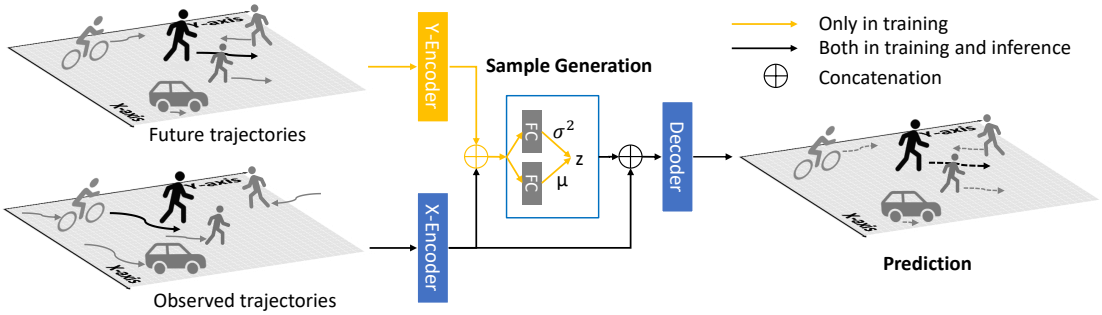


Figure 6.6: The overview of the framework for the attentive maps encoder network. FC stands for fully connected layer. The specific structure of the X-Encoder/Y-Encoder is given by Fig. 6.7.

Fig. 6.7 presents the detailed process of encoding. The X-Encoder is used to encode past information. It has two branches in parallel to process the motion information (upper branch) and dynamic maps information for interaction (lower branch). The upper branch takes the offsets $(\Delta x_i{}^t, \Delta y_i{}^t)_{t=1}^{T}$ for each target agent over the observed steps[2]. The motion information initially is

[2]Unlike in MCENet, here only the offsets are leveraged as the motion input, but the one-hot encoding of the agent's transport mode is not used since this information is not provided by the experimental datasets.

passed to a 1D convolutional layer (Conv1D) with a one-step stride along the time axis to learn motion features one step after another. Then, the output is sequentially passed to an FC layer and an LSTM module for encoding the temporal features into a hidden state, which contains all the motion information of the target agent. The lower branch takes the dynamic maps $(\text{Map}_i^t)_{t=1}^T$ as input. The interaction information at each step is passed through a 2D convolutional layer (Conv2D) with the ReLU activation and a Max Pooling layer (MaxPool) for learning the spatial features among all the agents. The output of the MaxPool at each step is flattened and concatenated along the time axis to form a timely distributed feature vector. Then, the feature vector is fed forward to the attention layer for learning interaction information. Output of the attention layer is passed to an LSTM used to encode the dynamic interconnection in the sequence. Both the hidden states (the last output) from the motion LSTM and the interaction LSTM are concatenated and passed to an FC layer for feature fusion, forming the complete output of the X-Encoder, denoted as Φ_x.

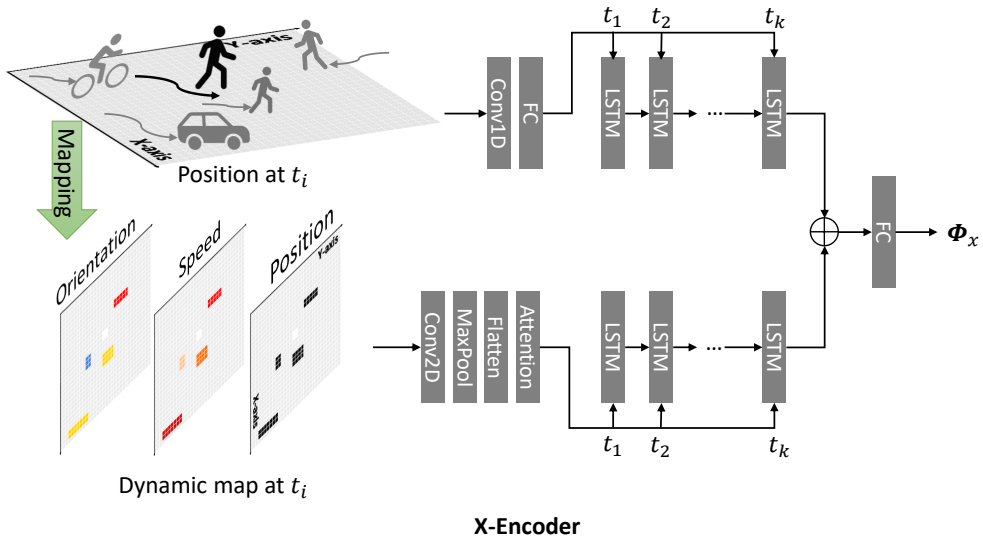


Figure 6.7: The structure of the X-Encoder of AMENet. The upper branch extracts motion information of target agents and the lower one attentively learns the interaction information between neighboring agents from the dynamic maps over time. Motion information and interaction information are encoded by their respective LSTMs sequentially. The last outputs of both LSTMs are concatenated and forwarded to a fully connected (FC) layer to get the final output of the X-Encoder. The Y-Encoder has the same structure as the X-Encoder.

The Y-Encoder has the same structure as the X-Encoder, which is used to encode both the target agent's motion and interaction information from the ground truth during training time. Dynamic maps are also leveraged in the Y-Encoder, although they are not reconstructed from the Decoder (only the future trajectories are reconstructed). This extended structure distinguishes AMENet from conventional CVAE structure (Kingma and Welling, 2014; Kingma et al., 2014; Sohn et al., 2015) and the work by Lee et al. (2017), in which only ground truth trajectories are inserted for training the recognition model.

6.2.2 Experiments

6.2.2.1 Datasets

The performance of the proposed framework has been validated in the *TrajNet challenge* (Sadeghian et al., 2018a), which is one of the most popular multi-scenario forecasting benchmarks. In this challenge, eight consecutive ground-truth steps (3.2 seconds) of each trajectory are for observation

and the following twelve steps (4.8 seconds) are required to forecast. TrajNet is a super-set of diverse popular benchmark datasets: BIWI (Pellegrini et al., 2009), Crowds (Lerner et al., 2007), MOT (Ferryman and Shahrokni, 2009), and SDD (Stanford Drone Dataset) (Robicquet et al., 2016). There is a total of 11,448 trajectories from these four subsets covering 38 scenes for training. The test data is from another 20 scenes without ground truth. Each scene presents various traffic densities in different space layouts, which makes the prediction task challenging and requires a model to generalize, in order to adapt to the various complex scenes. With the aim to guarantee a fair comparison, the TrajNet challenge provides a specific server for online evaluation. Table 6.5 lists the benchmark's statistics.

Table 6.5: The datasets hosted by the TrajNet benchmark (Sadeghian et al., 2018a) for testing AMENet.

Dataset name	**Agent type**	**Training**			**Online test**		
		#scenes	#trajs	#frames	#scenes	#trajs	#frames
BIWI (Pellegrini et al., 2009)	ped.	1	145	2.9k	1	51	1.0k
Crowds (Lerner et al., 2007)	ped.	5	2211	44.2k	2	268	5.4k
MOT (Ferryman and Shahrokni, 2009)	ped.	1	107	2.1k	0	0	0
SDD (Robicquet et al., 2016)	mixed*	31	8985	179.7k	17	2829	56.6k
Total	mixed	38	11448	228.9k	20	3148	63.0k

*pedestrians, bikers, skateboarders, cars, buses, and golf cars

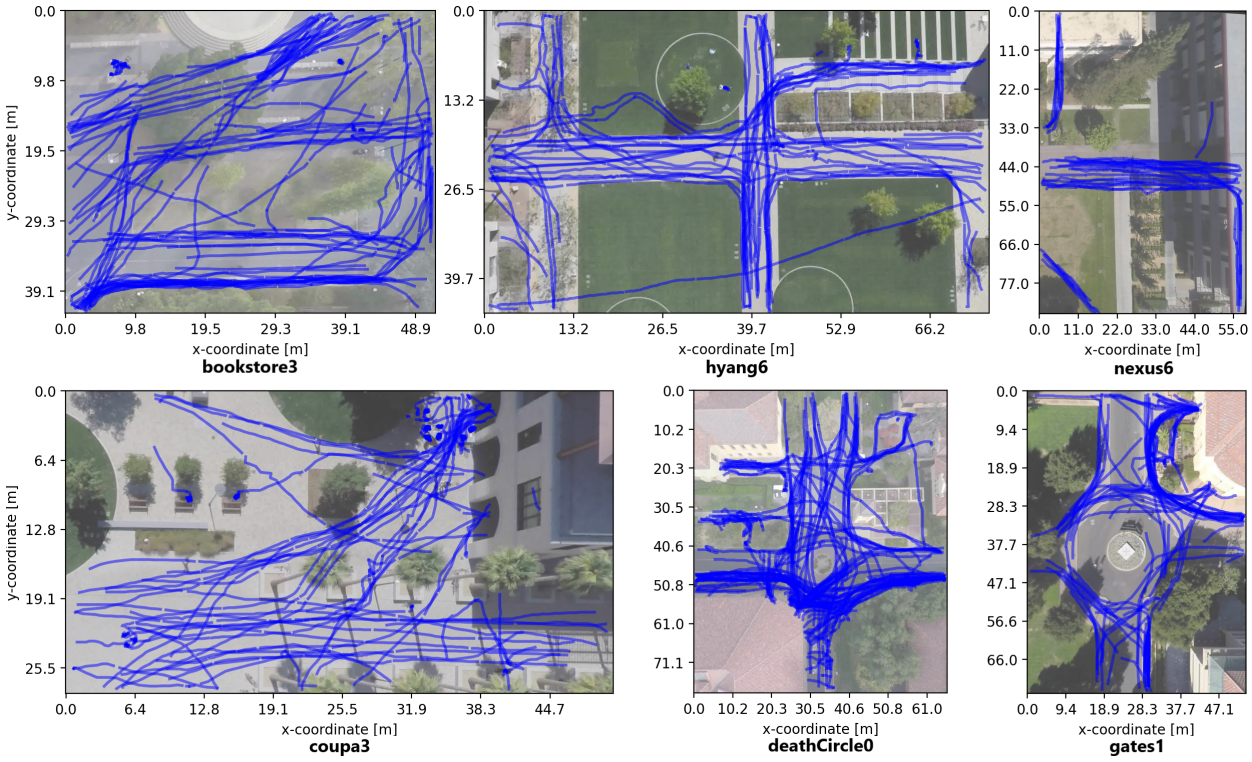


Figure 6.8: Visualization of each scene of the offline test set. The background images are retrieved from (Sadeghian et al., 2018a).

In order to train and evaluate the proposed method, as well as the ablation studies, six of the total 38 scenes in the training set are selected as the offline test set. Namely, they are *bookstore3*, *coupa3*, *deathCircle0*, *gates1*, *hyang6*, and *nexus0*. The selection of these scenes is based on the space layout, data density, and percentage of non-linear trajectories, as shown in Table 6.8. Fig. 6.8 visualizes the trajectories in each scene.

6.2.2.2 Evaluation Metrics

In addition to ADE/FDE, mostlikely and @*top*10 predictions (Sec. 6.1.2.2), all failed predictions that involve collisions and trajectories' linearity are employed to measure the performance of the proposed framework.

- A collision refers to two predicted trajectories intersecting each other with a distance of less than 0.1 meters at the same time. Considering the interval (0.4 seconds) between two consecutive steps is relatively large, the granularity of the current interval may not correctly capture how two trajectories intersect. Hence, following (Sadeghian et al., 2019) linear interpolation is applied to insert an intermediate step in order to reduce the interval to 0.2 seconds for collision analysis. In this study, all pairwise collisions are considered. As long as a collision is detected for a predicted trajectory, the prediction is considered to have failed.
- The linearity of a trajectory not only depends on the continuity of the travel direction, but also on the speed. A two-degree polynomial fitting, the same scheme provided by (Gupta et al., 2018), is used to categorize the linearity of trajectories. It compares the sum of the squared residuals over the fitting with the leastsquares error. A trajectory is categorized as linear if the sum of the squared residuals is smaller than a predefined error threshold.

6.2.2.3 Compared Models and Ablation Studies

The performance of AMENet is compared with the most influential previous models and the recent state-of-the-art models published on the TrajNet challenge leader board.

- *Linear (off)*: a simple temporal linear regressor using the offset of the consecutive positions of a trajectory as input;
- *Social Force* (Helbing and Molnar, 1995) is a rule-based model with the repelling force for collision avoidance and the attractive force for social connections;
- *S-LSTM* (Alahi et al., 2016) proposes social pooling with a rectangular occupancy grid for close neighboring agents;
- *SR-LSTM* (Zhang et al., 2019) uses a states refinement module for extracting social effects between the target agent and its neighboring agents;
- *RED* (Becker et al., 2018) uses an RNN-based Encoder with Multilayer Perceptron (MLP) for trajectory prediction;
- *MX-LSTM* (Hasan et al., 2018) exploits the head pose information of agents to help analyze their moving intention;
- *S-GAN* (Gupta et al., 2018) proposes to utilize GAN (Goodfellow et al., 2014) for multi-path trajectory prediction;
- *Ind-TF* (Giuliari et al., 2021) proposes a novel idea that utilizes the Transformer network (Vaswani et al., 2017) for sequence prediction. No social interactions between agents are considered by this model.

In order to analyze the impact of each module in the proposed framework, i. e., dynamic maps, self-attention, and the extended structure of the CVAE, several ablative models are investigated, as listed in Table 6.6.

- ENet: (E)ncoder (Net)work, which is conditioned only on the motion information. The interaction information is not leveraged. This model is treated as the baseline model.

– OENet: (O)ccupancy+ENet, where interactions are modeled by the occupancy grid in both the X-Encoder and the Y-Encoder.

– AOENet: (A)ttention+OENet, where the self-attention mechanism is added.

– MENet: (M)aps+ENet, where interactions are modeled by the proposed dynamic maps in both the X-Encoder and the Y-Encoder.

– ACVAE: (A)ttention+CVAE, where the dynamic maps are added only in the X-Encoder. It is equivalent to a CVAE model with the self-attention mechanism.

– AMENet: (A)ttention+MENet, where the self-attention mechanism is added. It is the full model of the proposed framework.

Table 6.6: The ablative models of AMENet.

Model	**X-Encoder**				**Y-Encoder**			
name	motion	occupancy	dynamic maps	attention	motion	occupancy	dynamic maps	attention
ENet*	√				√			
OENet	√	√			√	√		
AOENet	√	√		√	√	√		√
MENet	√		√		√		√	
ACVAE	√		√	√	√			
AMENet	√		√	√	√		√	√

*the baseline model

The detailed settings of the experiments can be found at the open-source project repository: `https://github.com/haohao11/AMENet`.

6.2.3 Results

6.2.3.1 Results for TrajNet Challenge

The performances of single trajectory prediction from different methods on the TrajNet benchmark are given in Table 6.7. The results were originally reported on the leader board[3] up to the date of 14 June 2020.

AMENet outperformed the other models and won the first place measured by the aforementioned metrics. From the table, it is apparent that AMENet reduces the prediction errors by a large margin in comparison with the most cited and pioneering deep learning model S-LSTM, as well as the generative model S-GAN that is based on GAN. Meanwhile, AMENet produces more accurate predictions than the linear model and the rule-based model, as well as three other recent deep learning models, i. e., MX-LSTM, SR-LSTM, and RED. Compared with the most recent model Ind-TF, AMENet achieves comparable performance in terms of ADE and slightly better performance in terms of FDE (from 1.197 to 1.183 m). The superior performance given by AMENet here also validates the efficacy of the ranking method to select the mostlikely prediction from the multiple predicted trajectories, as introduced in Sec. 4.2.2. It should be noted that the online server does not provide an evaluation for collision and linearity analyses. Hence, they are not reported in Table 6.7.

[3] `http://trajnet.stanford.edu/result.php?cid=1`, accessed on 14 June 2020.

Table 6.7: The results of AMENet tested on TrajNet. Smaller values indicate a better performance and best values are highlighted in boldface.

Model	Avg. [m]↓	FDE [m]↓	ADE [m]↓
S-LSTM (Alahi et al., 2016)	1.3865	3.098	0.675
S-GAN (Gupta et al., 2018)	1.3340	2.107	0.561
MX-LSTM (Hasan et al., 2018)	0.8865	1.374	0.399
Linear (off)	0.8185	1.266	0.371
Social Force (Helbing and Molnar, 1995)	0.8185	1.266	0.371
SR-LSTM (Zhang et al., 2019)	0.8155	1.261	0.370
RED (Becker et al., 2018)	0.7800	1.201	0.359
Ind-TF (Giuliari et al., 2021)	0.7765	1.197	**0.356**
AMENet*	**0.7695**	**1.183**	**0.356**

*named *ikg_tnt* on the leader board of TrajNet challenge

6.2.3.2 Results for Multi-Path Prediction

The performance for multi-path prediction is investigated quantitatively and qualitatively. The offline test set is used for this purpose so that the evaluation can be measured in detail with the information of ground truth. Table 6.8 shows the quantitative results. Compared to the mostlikely prediction, as expected the @top10 prediction yields similar but slightly better performance. This indicates that (1) the generated multiple trajectories increase the chance to narrow down the errors; (2) the ranking method is effective for ordering the multiple predictions and proposing a good one, which is especially important for tasks where prior knowledge of the ground truth is not available, e. g., path planning for automated agents.

The performance of AMENet is also measured by the number of failed trajectories according to collision analysis. It is clear that across the datasets, in most cases AMENet successfully predicts trajectories without collisions in bookstore3, hyang6, and nexus6 in terms of the @*top*10 and the mostlikely predictions. But it generates six and two failed trajectories in deathCircle0 and gates1, respectively, in terms of the @*top*10 prediction. Similarly, it outputs six and two failed trajectories in coupa3 and deathCircle0, respectively, in terms of the mostlikely prediction. In total, the average failure rate is 0.3 %, i. e., on average 1.3 out of 407 trajectories predicted with collisions. This failure rate is very low. Nevertheless, out of the concern for traffic safety this number should be reduced to zero.

Table 6.8: The results of AMENet tested on the TrajNet offline test set. Evaluation results are measured by ADE/FDE/failed predictions for multi-path trajectory prediction.

Dataset	Layout	#Trajs	Non-linear traj rate	@*top*10	mostlikely
bookstore3	parking	429	0.71	0.477/0.961/0	0.486/0.979/0
coupa3	corridor	639	0.31	0.221/0.432/0	0.226/0.442/6
deathCircle0	roundabout	648	0.89	0.650/1.280/6	0.659/1.297/2
gates1	roundabout	268	0.87	0.784/1.663/2	0.797/1.692/0
hyang6	intersection	327	0.79	0.534/1.076/0	0.542/1.094/0
nexus6	corridor	131	0.88	0.542/1.073/0	0.559/1.109/0
Avg.	-	407	0.74	0.535/1.081/1.3	0.545/1.102/1.3

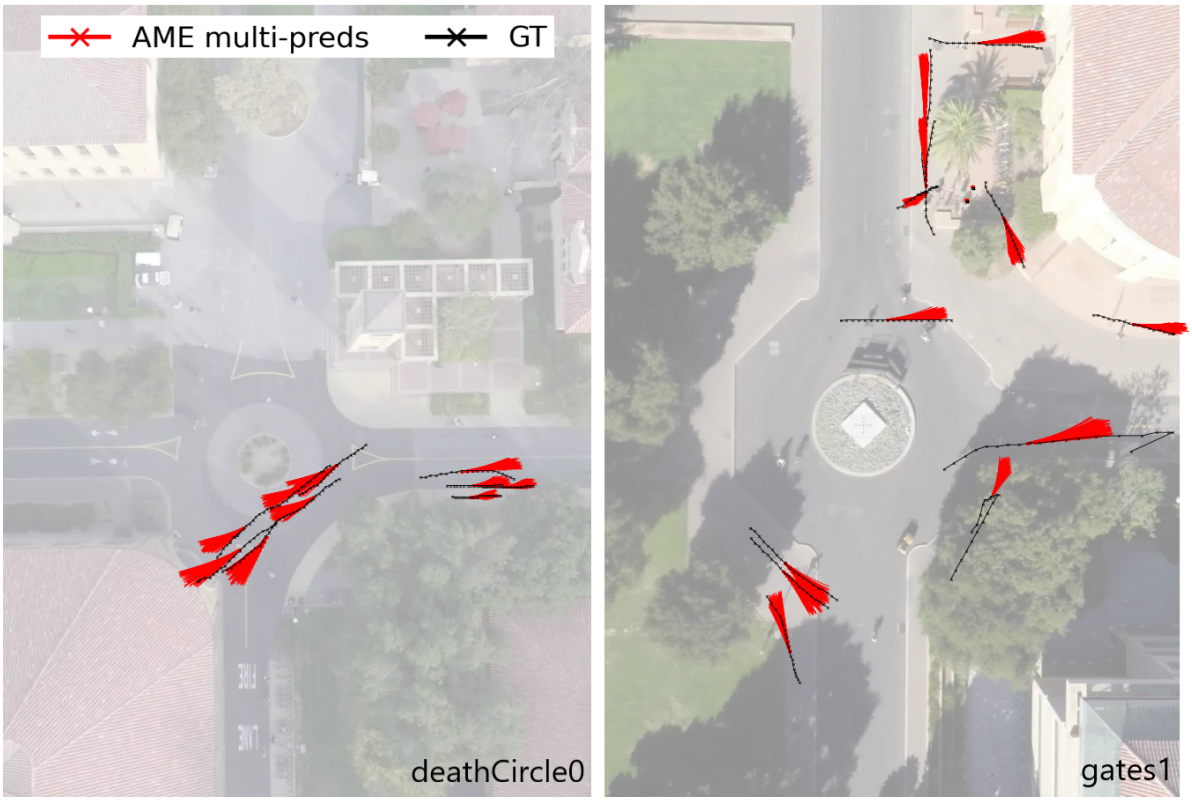


Figure 6.9: The results of AMENet for multi-path prediction. The background images are retrieved from (Sadeghian et al., 2018a).

Fig. 6.9 showcases some qualitative examples of the multi-path trajectory prediction by AMENet. As shown in the roundabout deathCircle0 and gates1, each moving agent has more than one possibility (different speed and orientation) to choose a future path. The predicted trajectories diverge more widely in further steps as uncertainty about an agent's intention increases with time. In comparison to single-path trajectory prediction, predicting multiple plausible trajectories indicates a larger intended area and raises the chance to cover the correct path an agent might choose in the future. Also, the "fan" of possible trajectories can be interpreted as reflecting uncertainty of prediction. Conversely, a single prediction provides limited information for inference and is likely to lead to a false conclusion if the prediction is not correct/precise in the early steps. On the other hand, agents that stand still are correctly predicted by AMENet with high certainty, as shown by two agents in gates1 in the upper right area. As designed by the model, only interactions between agents lead to adaptions in the predicted path and deviation from linear paths; the scene context, e. g., road geometry, is not modeled which does not affect prediction.

6.2.3.3 Results for Ablation Studies

Table 6.9 shows the quantitative results for the ablation studies. Errors are measured by ADE/FDE for the mostlikely prediction. The comparison between OENet and the baseline model ENet shows that extracting interaction information from the occupancy grid does not lead to a better performance. Even though the self-attention mechanism is added to the occupancy grid (denoted by AOENet), the slightly enhanced performance still falls behind the baseline model. The comparison indicates that interactions are not effectively learned from the occupancy grid regardless of the self-attention mechanism across the datasets. Comparison between MENet and ENet shows a similar pattern. Performance is slightly less inferior using the dynamic maps than it is using the

occupancy grid (MENet vs. OENet) in comparison to the baseline model. However, profound improvements become apparent after employing the self-attention mechanism. First, the comparison between ACVAE and ENet shows that even without the extended structure in the Y-Encoder, the dynamic maps with the self-attention mechanism in the X-Encoder are very beneficial for modeling interactions. On average, performance is improved by 4.0 % and 4.5 % as measured by ADE and FDE, respectively. Second, comparison between the proposed model AMENet and ENet shows that after extending the dynamic maps to the Y-Encoder, errors, especially the absolute values of FDE, further decrease across all the datasets; ADE is reduced by 9.5 % and FDE is reduced by 10.0 %. This improvement has also been confirmed by the benchmark challenge (as shown in Table 6.7).

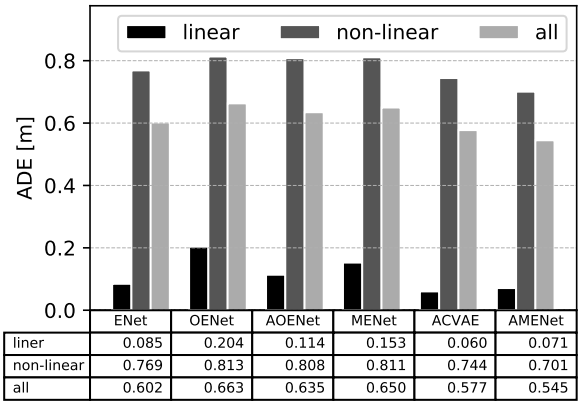
Table 6.9: The results of AMENet for ablation studies. Evaluation results are measured by ADE/FDE for the mostlikely prediction of the ablative models and the proposed model AMENet. Best values are highlighted in boldface.

Scene	ENet	OENet	AOENet	MENet	ACVAE	AMENet
bookstore3	0.532/1.080	0.601/1.166	0.574/1.144	0.576/1.139	0.509/1.030	**0.486/0.979**
coupa3	0.241/0.474	0.342/0.656	0.260/0.509	0.294/0.572	0.237/0.464	**0.226/0.442**
deathCircle0	0.681/1.353	0.741/1.429	0.726/1.437	0.725/1.419	0.698/1.378	**0.659/1.297**
gates1	0.876/1.848	0.938/1.921	0.878/1.819	0.941/1.928	0.861/1.823	**0.797/1.692**
hyang6	0.598/1.202	0.661/1.296	0.619/1.244	0.657/1.292	0.566/1.140	**0.542/1.094**
nexus6	0.684/1.387	0.695/1.314	0.752/1.489	0.705/1.346	0.595/1.181	**0.559/1.109**
Avg.	0.602/1.224	0.663/1.297	0.635/1.274	0.650/1.283	0.577/1.170	**0.545/1.102**

Furthermore, the evaluation is decomposed for non-linear and linear trajectories across all of the above models. Fig. 6.10 visualizes the values of (a) ADE and (b) FDE averaged over the six scenes in the offline test set. Across the models, the performance for predicting non-linear trajectories demonstrates a similar pattern compared to predicting all the trajectories (linear + non-linear) and AMENet outperforms the other models measured by both metrics. Obviously, predicting the linear trajectories is easier than the non-linear ones. In this regard, all the models perform very well (ADE ≤ 0.2 m and FDE ≤ 0.4 m), especially the AMENet and ACVAE models. This observation indicates that if there are other agents interacting with each other, the continuity of their motion is likely to be interrupted, i. e., deviating from the free-flow trajectories (Rinke et al., 2017). The model has to adapt to this deviation to achieve a good performance. On the other hand, if there is no such reason to disrupt the linearity of motion, then the model does not generate deviated trajectories.

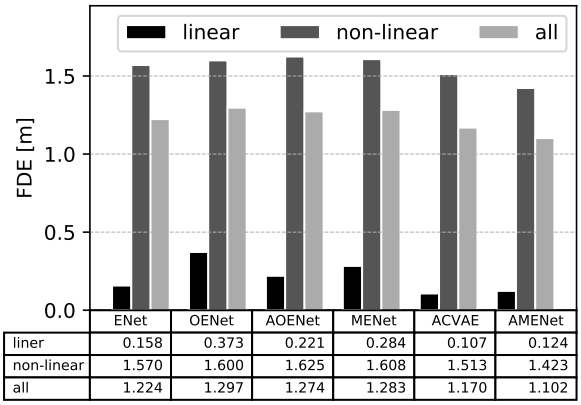
6.2.3.4 Qualitative Results

Fig. 6.11 showcases some qualitative results by the proposed AMENet model in comparison to the ablative models. In general, AMENet generates accurate predictions and outperforms the other models in all the scenes, which is especially visible in coupa3 (a) and bookstore3. All the models predict plausible trajectories for two agents walking in parallel in coupa3 (b) (denoted by the black box), except the baseline model ENet. Without modeling interactions, the ENet model generates two trajectories that intersected. In hyang6 limited performance was shown by ENet, AOENet, and MENet regarding travel speed and by OENet and ACVAE regarding destination for the fast-moving agent. In contrast, AMENet still provides a good prediction. In nexus6 (a) and (b), only two agents were present, and all models perform well. More agents were involved in the roundabout scenes, in which the prediction task is more challenging. AMENet generates accurate predictions for most of the agents. However, its performance is limited by the agents that changed speed or direction rapidly from their initial movement. There is also the particular



	ENet	OENet	AOENet	MENet	ACVAE	AMENet
liner	0.085	0.204	0.114	0.153	0.060	0.071
non-linear	0.769	0.813	0.808	0.811	0.744	0.701
all	0.602	0.663	0.635	0.650	0.577	0.545

(a) ADE



	ENet	OENet	AOENet	MENet	ACVAE	AMENet
liner	0.158	0.373	0.221	0.284	0.107	0.124
non-linear	1.570	1.600	1.625	1.608	1.513	1.423
all	1.224	1.297	1.274	1.283	1.170	1.102

(b) FDE

Figure 6.10: The results of AMENet for predicting linear and non-linear trajectories. The prediction errors for linear, non-linear and all trajectories are measured by (a) ADE and (b) FDE for all ablative models, as well as the proposed model AMENet.

interesting scenario of the two agents that walked towards each other in deathCircle0 (denoted by the black box). In reality, when the right agent changed its heading towards the left agent, the left agent had to decelerate strongly to yield. Regarding the interaction and compared with the other models, AMENet generates non-conflict trajectories.

6.2.4 Discussion

Based on the extensive studies and results, the advantages and limitations of the AMENet model are discussed below.

AMENet demonstrated superior performance in different shared spaces. Firstly, the proposed model was able to achieve a state-of-the-art performance on the TrajNet benchmark challenge datasets, which contain various scenes. Secondly, the results of the ablation studies demonstrated that information of interactions between agents is beneficial for trajectory prediction. However, the performance highly depends on how such information is leveraged. It was difficult for the occupancy grid, which is only based on the positions of the neighboring users, to extract useful information for interaction modeling, because positions change from one step to the next and from one scene to another. The speed and orientation information is not considered by the occupancy grid, which may explain why the occupancy grid performed worse than the dynamic maps in the same settings. Thirdly, as interactions change over time, the self-attention mechanism automatically extracts the salient features from the dynamic maps over different steps.

However, several limitations of the model have been uncovered throughout the experiments. First, the resolution of the map was approximated according to the experimental data and the size of the neighboring agents was not yet considered. This may limit the model when dealing with big-sized agents, such as buses or trucks. Second, from the qualitative results it is clear that the model showed limited performance for predicting the behavior of the agents that drastically changed direction and speed, which is in general a very challenging task without extra information from the agents, such as body posture or line of gaze. Last but not least, in this study, scene context information was not included. The lack of this information may lead to a wrong prediction, e. g., trajectories leading into obstacles or inaccessible areas. Scene context can have a positive effect on trajectory prediction, e. g., a trajectory follows a (curved) path, whereas a strong constraint

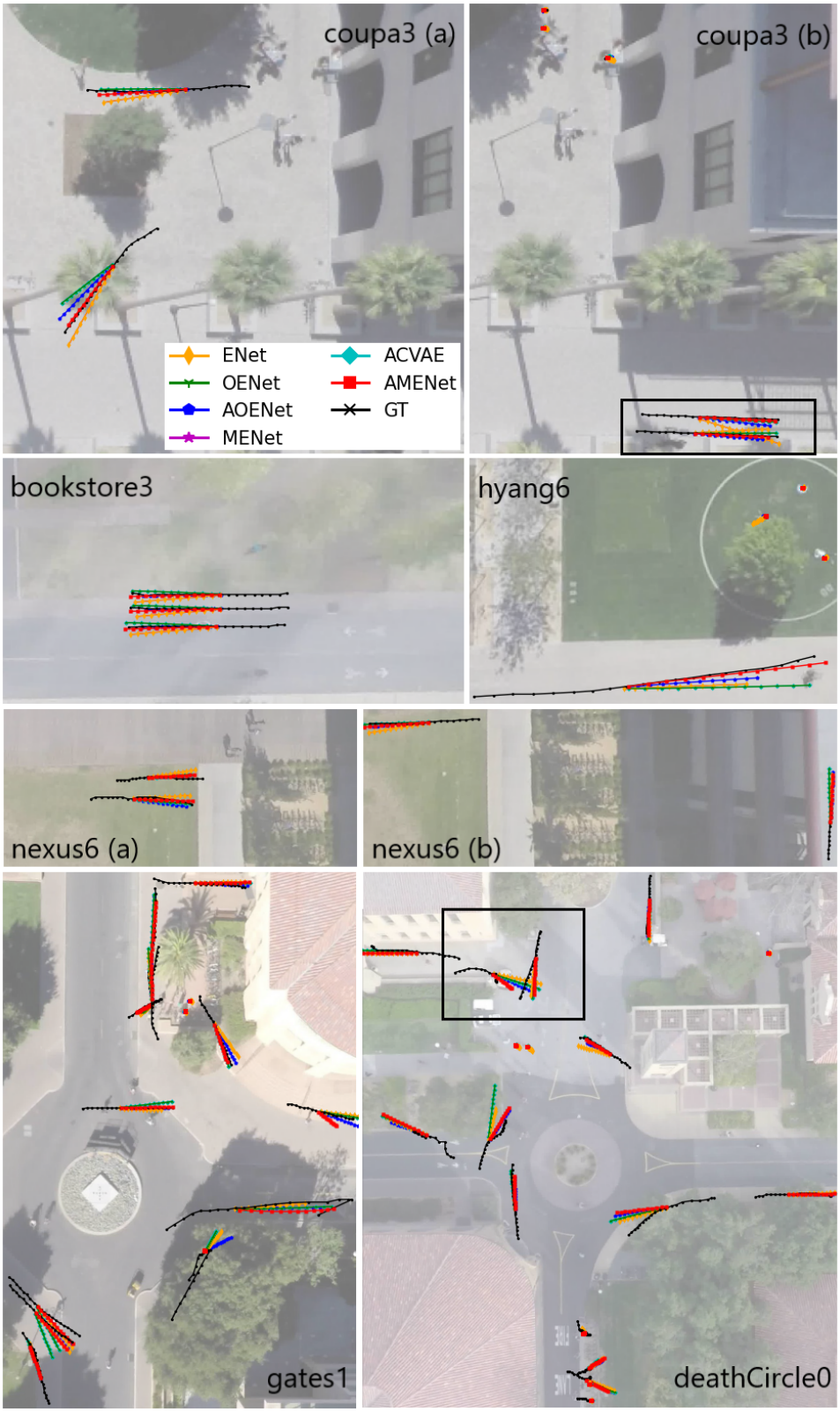


Figure 6.11: The qualitative results of AMENet and ablative models. Trajectories are predicted by ENet, OENet, AOENet, MENet, ACVAE, and AMENet in comparison with the ground truth (GT) trajectories on TrajNet. The background images are retrieved from (Sadeghian et al., 2018a).

from the scene context can easily overfit a model for some particular scene layouts (as discussed in Sec. 6.1.4). Hence, a good mechanism for parsing the scene information is needed to balance the trade-off, especially for a model trained on one scene and applied to another.

6.2.5 Summary

In this section, the Attentive Maps Encoder Network (AMENet) has been proposed to use motion and interaction information for multi-path trajectory prediction of heterogeneous road user agents in various real-world environments. The latent space learned by the X-Encoder and Y-Encoder for both sources of information enables the model to capture the stochastic properties of motion behavior for predicting multiple plausible trajectories after a short observation time. The efficacy and the generalizability of the model were validated on one of the the most challenging benchmarks, TrajNet, which contains various datasets in different unseen real-world environments. This framework not only achieved state-of-the-art performance, but also won the first place on the leader board[4] for predicting 12 time-step positions of 4.8 seconds. Meanwhile, each component of AMENet was analyzed via a series of ablation studies.

The following summarizes improvements made from the previous framework MCENet to the framework AMENet introduced in this section.

Compared to MCENet, AMENet proposes a novel way—attentive dynamic maps—to extract the social effects between agents during interactions. The dynamic maps capture accurate interaction information by encoding not only the neighboring agents' relative position but also orientation and travel speed in relation to the target agent, while the self-attention mechanism enables the model to learn the global dependency of interaction over different steps.

In contrast to MCENet, AMENet is designed in a way that scene context information is not considered and only the interactions between agents will lead to adaption of agents' movement. This design remedies the problem of overfitting a model for a specific scene such that the model cannot be easily generalized for new scenes, where the scene context is totally different from the one the model has been trained on.

However, how to effectively use the scene context information and avoid overfitting still needs to be further explored in future work.

[4]This evaluation performance was measured at the time of its submission to the online server.

6.3 Dynamic Context Encoder Network

The results presented in this Sec. 6.3 are adapted from (Cheng et al., 2021b), pending to be published in the proceedings of the 2021 IEEE International Conference on Robotics and Automation at the time of completion of this thesis.

In this section, the third framework is introduced, with the aim to improve performance of the previous framework AMENet (Sec. 6.2.1) for trajectory prediction.

6.3.1 Framework

The *Dynamic Context Encoder Network* (DCENet) predicts multi-path trajectories of heterogeneous agents by enhancing the self-attention architectures (Vaswani et al., 2017). Fig. 6.12 depicts the general structure of the self-attention architecture.

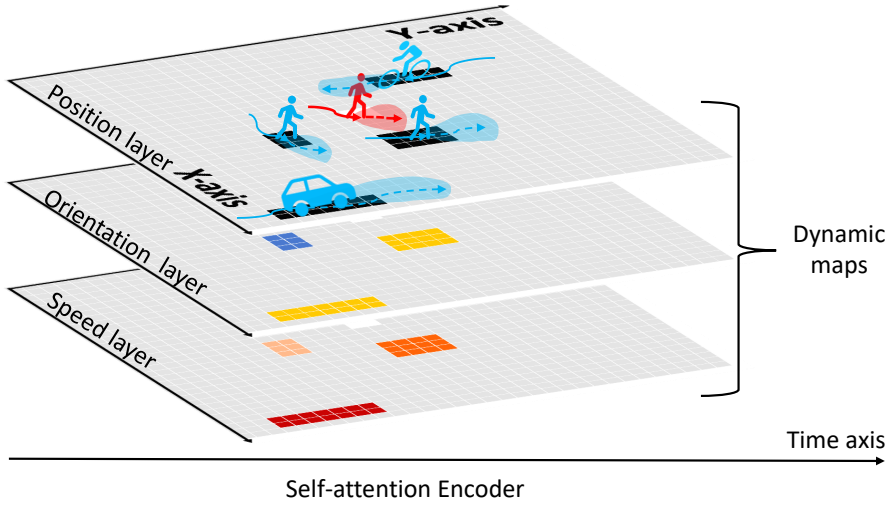


Figure 6.12: The pipeline of dynamic context encoder network. Multiple future trajectories (the mostlikely one indicated by dash lines with the shaded area of possible trajectories) of a target agent (in red) are predicted conditioned on its observed movement (solid line) with consideration of its interactions between neighboring agents (in blue) in mixed traffic. Interaction is learned through the dynamic maps with each layer dedicated to capturing position, orientation, and speed information (indicated by color-coded rectangles) using self-attention structures.

Built upon the previous framework AMENet (Sec. 6.2.1), the following aspects are further investigated by DCENet for trajectory prediction.

1) Dedicated self-attention structures are designed for learning motion information and agent-to-agent interaction information, respectively.

2) The model is further generalized to predict accurate trajectories not only in unseen shared spaces, but also at intersections in mixed traffic.

Fig. 6.13 depicts the detailed framework of DCENet. It has four modules—Encoder X, Encoder Y, latent space, and Decoder. It is shown that DCENet inherits a similar framework from AMENet. The major difference is that DCENet adopts a two-stream architecture (Simonyan and Zisserman, 2014a) with dedicated self-attention mechanisms. One stream is dedicated to learning the spatial context and another stream to learning temporal context explicitly. These streams are later fused.

In this way, exploiting the complex dynamic spatial–temporal context is decomposed into a) learning the temporal dependencies between steps by using the self-attention and a following global average pooling and b) learning spatial context at a step among agents by using the self-attention and an LSTM. The following explains the learning process in detail.

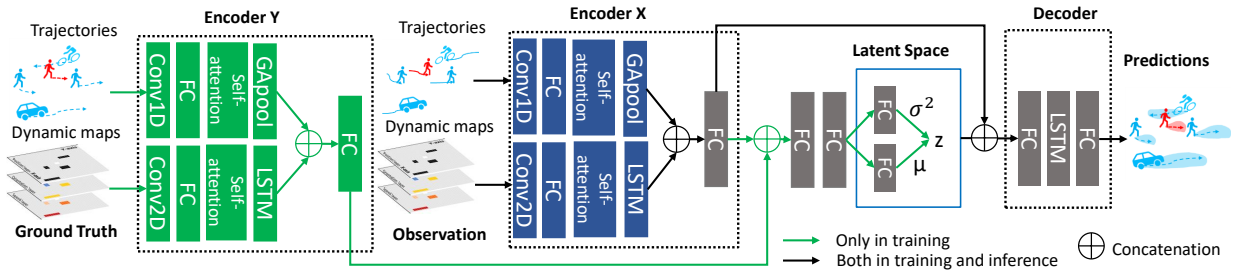


Figure 6.13: The framework of DCENet. Encoder X and Encoder Y are identical in structure. Conv1D stands for 1D convolutional layer and Conv2D for 2D convolutional layer, FC stands for fully connected layer, GApool stands for global average pooling layer, and LSTM stands for Long Short-Term Memory.

Encoder X and Encoder Y are identical in structure and work analogously for encoding the observed and future (ground truth) information, respectively. As shown in Fig. 6.13, there are two branches in both encoders dedicated to learning motion and agent-to-agent interaction information. Without loss of generality, the encoding process is explained below by taking Encoder X as an example.

- The upper branch of Encoder X is constructed in the order of a 1D convolutional (Conv1D) layer, a fully connected layer (FC), and a self-attention layer followed by a global average pooling layer. The Conv1D layer convolutionally learns the motion information one step after another. The FC layer is used for connecting all the steps. The self-attention layer with the global average pooling block attentively learns an agent's motion pattern over time, which is equivalent to the general structure of the Transformer encoder (Sec. 2.5). Hence, this block is also called Transformer encoder in this framework. In comparison, AMENet employs an LSTM for learning the motion information.
- In parallel to the upper branch, the lower branch is constructed in the order of a 2D convolutional (Conv2D) layer, an FC layer, a self-attention layer, and an LSTM. The Conv2D layer extracts spatial information from the dynamic map at each step. The following FC and self-attention layers work in a similar way as the previous branch. In the end, instead of using global average pooling, an LSTM is used to encode the interaction information. It should be noted that the structure of the lower branch is identical to the corresponding structure in the AMENet framework.
- The encodings of the two branches are concatenated and passed to an FC layer. The output of the FC layer is the final encoding of Encoder X.

During the training phase, both the observed trajectory and dynamic maps, as well as the future trajectory and dynamic maps are encoded by the above encoders. Then, their encodings are concatenated and passed through two FC layers for fusion. After that, two side-by-side FC layers are used to estimate the mean and the variance of the latent variables. A future trajectory is reconstructed by the LSTM Decoder step by step conditioned on the encodings of observation and the latent variables.

In the inference phase, the ground truth of the future trajectory is no longer available and its pathway is removed (color coded in green in Fig. 6.13). A latent variable is sampled from the prior Gaussian distribution and concatenated with the observation encodings, which together serve as

the condition for the following trained Decoder for predicting a future trajectory. The sampling and decoding process is repeated multiple times to predict multiple trajectories.

6.3.2 Experiments

6.3.2.1 Datasets

DCENet is tested on both the TrajNet benchmark (Sec. 6.2.2.1) and the newly published large-scale inD[5] benchmark. inD consists of 33 subsets and was collected using drones from four very busy intersections in Germany by Bock et al. (2019). In contrast to TrajNet, in which most of the environments (i. e., shared spaces) are pedestrian friendly, the intersections in inD are dominated by vehicles. This makes the prediction task more challenging due to the very different travel speeds between pedestrians and vehicles, as well as their direct interactions. The inD benchmark was processed by following the same format as the TrajNet benchmark: Each trajectory contains 20 consecutive steps, the first eight steps for observation and the following twelve steps for prediction. The interval between two consecutive steps lasts 0.4 seconds. Table 6.10 lists the statistics of the inD benchmark after processing.

Table 6.10: The inD benchmark for testing DCENet. It was acquired from four different intersections in Germany.

Intersection type	**Agent type**	**Training**			**Test**		
		#subsets	#trajs	#frames	#subsets	#trajs	#frames
Intersection-(A)	mixed	5	525	10.5k	2	199	4.0k
Intersection-(B)	mixed	7	815	16.3k	4	646	12.9k
Intersection-(C)	mixed	8	2809	56.2k	4	1807	36.1k
Intersection-(D)	mixed	2	122	2.4k	1	87	1.7k
Total	mixed	22	4271	85.4k	11	2739	54.7k

6.3.2.2 Compared Models and Ablation Studies

Performance of DCENet is compared with the most influential previous models and the recent state-of-the-art models published on the TrajNet challenge. It is also compared with the three most representative models—S-LSTM and S-GAN, as well as AMENet (Sec. 6.2.2) on the inD benchmark. All models were trained and tested by using the same data in order to guarantee a fair comparison.

- S-LSTM (Alahi et al., 2016) is the first deep learning method that uses occupancy grid for modeling interactions between agents. In comparison, DCENet employs attentive dynamic maps for agent-to-agent interaction (Sec. 4.2.3.2).
- S-GAN (Gupta et al., 2018) adopts GAN (Goodfellow et al., 2014) for multi-path trajectory prediction. In comparison, DCENet adopts CVAE (Sohn et al., 2015) as the generative module for multi-path trajectory prediction.
- AMENet (Sec. 6.2.2) only employs the self-attention mechanism (Vaswani et al., 2017) for learning agent-to-agent interaction. In comparison, DCENet adopts a two-stream architecture (Simonyan and Zisserman, 2014a) of attention modules, with respective streams dedicated to learning the spatial and temporal contexts explicitly.

[5] `https://www.ind-dataset.com/`, accessed on 31 October 2020.

A series of ablation studies is designed to analyze the impact of each proposed module, i. e., dynamic maps, the Transformer encoder, and LSTM encoder–decoder:

- *Baseline:* an LSTM encoder–decoder that only uses the observed trajectory as input;
- *DCENet w/o DMs:* the branch of encoding dynamic maps (DMs) is removed from the final DCENet model;
- *Trans. En&De*: the LSTM encoder–decoder is substituted by the Transformer encoder (the self-attention layer + global average pooling) in the DCENet framework

Table 6.11: The ablative models of DCENet.

Model name	**Motion encoder**		**Interaction encoder**		**Decoder**	
	LSTM	att+GAp[2]	att+LSTM [1]	att+GAp[2]	LSTM	att+GAp[2]
Baseline	√				√	
AMENet	√		√		√	
DCENet w/o DMs		√			√	
Trans. En&De:		√		√		√
DCENet		√	√		√	

[1]a self-attention layer plus an LSTM
[2]a self-attention layer plus an average global pooling, which is equivalent to the Transformer encoder (more detail is shown in Sec. 2.5)

The detailed settings of the experiments can be found at the open-source project repository: `https://github.com/haohao11/DCENet`.

6.3.3 Results

6.3.3.1 Quantitative Results

Experimental results of different methods including the ablative models of DCENet reported on the TrajNet leader board are listed in Table 6.12. The results shown were taken on 31 October 2020. It is clear that DCENet reports new state-of-the-art performance and the ablative models also demonstrate performances comparable to previous work.

First, in comparison to the baseline model, both DCENet w/o DMs and Ind-TF achieve better performance. DCENet w/o DMs slightly outperforms Ind-TF in the average score (0.7760 m vs. 0.7765 m) and FDE (1.195 m vs. 1.197 m), but falls behind it in ADE (0.357 m vs. 0.356 m). Given the fact that both DCENet w/o DMs and Ind-TF only use observed trajectories as input, the proposed framework (self-attention + LSTM encoder–decoder) is effective for exploring spatial–temporal context in a comparable level as the Transformer-based network. Furthermore, Ind-TF utilizes BERT, a heavily stacked Transformer structure and must be pre-trained on an external large-scale dataset, while DCENet does not require it.

Second, from comparing the baseline model to S-LSTM, it is evident that the former generates significantly smaller errors. The main difference between them is that the baseline model is CVAE-based and generates multiple trajectories. It indicates that the future motion of humans is of high uncertainty and predicting a set of possible trajectories is more practical than only predicting a single one. Meanwhile, the baseline model also significantly outperformed S-GAN, which is also a generative model for predicting multiple trajectories.

Third, Trans. En&De that adopts the Transformer encoder for both encoding and decoding, interestingly, does not achieve improved performance compared to DCENet. This phenomenon implies

Table 6.12: The results of DECNet tested on the TrajNet. Models are categorized into deterministic and stochastic depending on whether they incorporate a generative module.

Model	Category	Avg. [m]↓	FDE [m]↓	ADE [m]↓
S-LSTM (Alahi et al., 2016)	deterministic	1.3865	3.098	0.675
S-GAN (Gupta et al., 2018)	stochastic	1.334	2.107	0.561
MX-LSTM (Hasan et al., 2018)	deterministic	0.8865	1.374	0.399
Linear (off)	deterministic	0.8185	1.266	0.371
Social Force (Helbing and Molnar, 1995)	deterministic	0.8185	1.266	0.371
SR-LSTM (Zhang et al., 2019)	deterministic	0.8155	1.261	0.370
RED (Becker et al., 2018)	deterministic	0.7800	1.201	0.359
Ind-TF (Giuliari et al., 2021)	deterministic	0.7765	1.197	0.356
AMENet (Sec. 6.2)	stochastic	0.7695	1.183	0.356
Baseline	stochastic	0.8045	1.239	0.370
DCENet w/o DMs	stochastic	0.7760	1.195	0.357
Trans. En&De	stochastic	0.7780	1.196	0.360
DCENet	stochastic	**0.7660**	**1.179**	**0.353**

that the self-attention + LSTM encoder–decoder structure is more sophisticated for exploring dynamic context between agents than Trans. En&De for trajectory prediction. Meanwhile, Trans. En&De slightly falls behind AMENet which uses the LSTM network as decoder. These observations indicate that the Transformer encoder structure is not as efficient as a simple LSTM network for the decoding process.

Lastly, DCENet and AMENet outperform DCENet w/o DMs. The improved performances suggest that the dynamic maps are beneficial for modeling interactions between agents for trajectory prediction.

Furthermore, DCENet is tested on inD to investigate its performance and generalizability. Table 6.13 lists the quantitative results measured by ADE/FDE. DCENet achieves superior performance with respect to the @*top*10 prediction across all intersections and reduces the errors by a large margin. It also outperforms the other models with respect to the mostlikely prediction at three out of four intersections. DCENet only slightly falls behind the AMENet model at intersection-(C). As anticipated, the performance of the mostlikely prediction falls behind the performance of the @*top*10 prediction. However, the ranking method is still effective in recommending a reliable candidate when compared to the other models.

Table 6.13: The results of DECNet tested on inD that are measured by ADE/FDE.

Model	S-LSTM	S-GAN	AMENet	DCENet
inD		*@top 10*		
Intersection-(A)	2.04/4.61	2.84/4.91	0.95/1.94	**0.72/1.50**
Intersection-(B)	1.21/2.99	1.47/3.04	0.59/1.29	**0.50/1.07**
Intersection-(C)	1.66/3.89	2.05/4.04	0.74/1.64	**0.66/1.40**
Intersection-(D)	2.04/4.80	2.52/5.15	0.28/0.60	**0.20/0.45**
Avg.	1.74/4.07	2.22/4.29	0.64/1.37	**0.52/1.23**
inD		*mostlikely*		
Intersection-(A)	2.29/5.33	3.02/5.30	1.07/2.22	**0.96/2.12**
Intersection-(B)	1.28/3.19	1.55/3.23	0.65/1.46	**0.64/1.41**
Intersection-(C)	1.78/4.24	2.22/4.45	**0.83/1.87**	0.86/1.93
Intersection-(D)	2.17/5.11	2.71/5.64	0.37/0.80	**0.28/0.62**
Avg.	1.88/4.47	2.38/4.66	0.73/1.59	**0.69/1.52**

6.3.3.2 Qualitative Results

The qualitative results are shown in Fig. 6.14. The first two rows showcase the scenarios of the TrajNet benchmark. It should be noted that the qualitative analysis on TrajNet was carried out on the validation set (an independent subset of the training set) for comparison with the ground truth. DCENet accurately predicts two pedestrians walking towards each other at bookstore3. The shaded areas represent multiple possible trajectories. It also correctly predicts the static pedestrians in coupa3, as well as the pedestrians walking in parallel. In deathCircle0, DCENet predicts different possible turning angles for the cyclist in the roundabout. In hyang6, two pedestrians walking closely to each other are predicted correctly.

The last two rows showcase the scenarios of the inD benchmark. DCENet predicts a fast driving vehicle with a slightly different predicted speed at Intersection-(A). It predicts that a left-turning vehicle might turn at intersection-(B) at varying speed and turning angle. Meanwhile, it correctly predicts the vehicle in the neighborhood to stand still, a static pedestrian on the sidewalk, and another pedestrian could move at slightly different speed and angle. The model successfully predicts the interaction at the zebra crossing at intersection-(C), where the vehicle stopped to yield the way to the pedestrian. Similar predictions are made for the walking and static pedestrians, as well as the vehicle waiting at the entrance of intersection-(D). Overall, it is apparent that the recommended single path is very close to the corresponding ground truth for each agent.

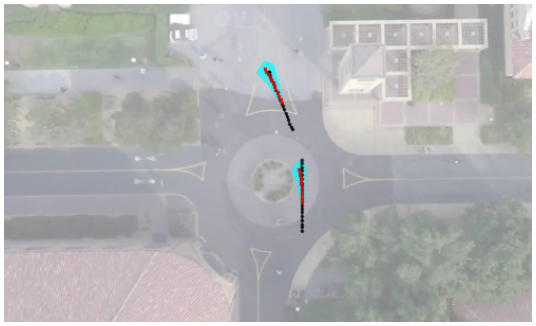
6.3.4 Discussion

The experiments conducted on the TrajNet benchmark have shown the efficacy of the DCENet framework. (1) DCENet was effective for predicting accurate trajectories for heterogeneous agents in various real-world traffic scenes. (2) Compared to the baseline model, DCENet learned interaction via dynamic maps with the two-stream self-attention structures effectively and achieved improved performance. (3) Both LSTM and the Transformer encoder were capable of learning complex sequential patterns, and their combination further enhanced the performance for trajectory prediction. (4) The experiments conducted on the inD benchmark further demonstrated the generalizability of the DCENet framework. It was able to generalize to different datasets both of shared spaces and intersections and to maintain a superior performance.

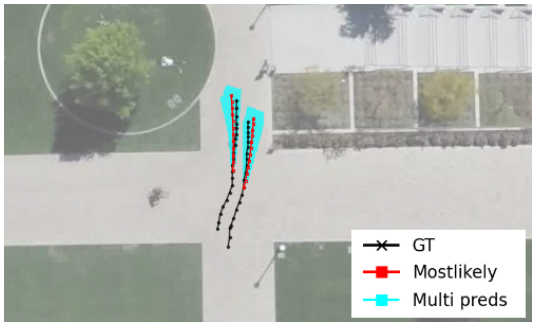
However, even though DCENet improves the performance compared to the previous framework AMENet (Sec. 6.2.2), it does not completely address the inherited limitations. Both frameworks

(a) TrajNet bookstore-3

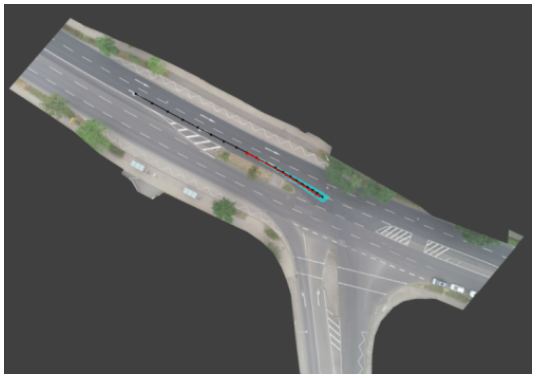
(b) TrajNet coupa-3



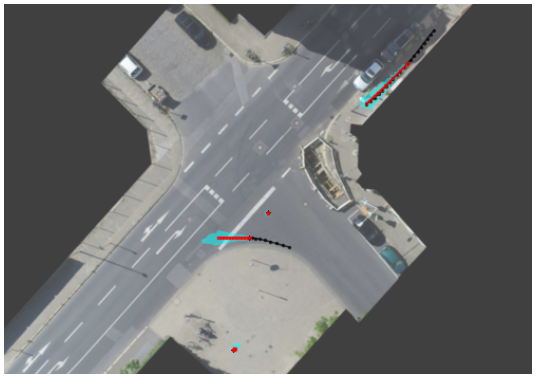
(c) TrajNet deathCircle-0



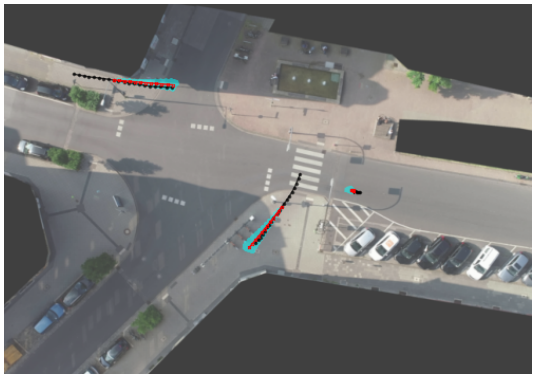
(d) TrajNet hyang-6



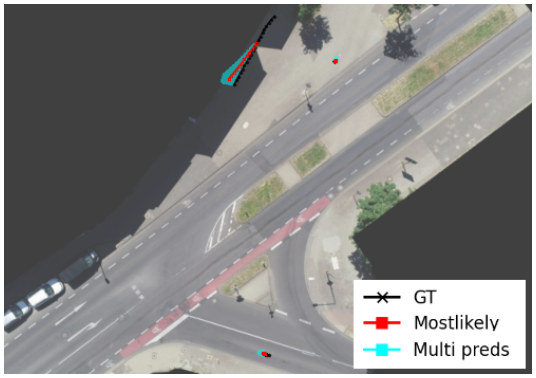
(e) inD Intersection-(A)



(f) inD Intersection-(B)



(g) inD Intersection-(C)



(h) inD Intersection-(D)

Figure 6.14: The qualitative results of DCENet. Multi-path trajectory predictions are generated by DCENet in the shared spaces of the TrajNet benchmark and at different intersections of the inD benchmark. The background images of (a)-(d) are retrieved from (Sadeghian et al., 2018a) and (e)-(h) are retrieved from (Bock et al., 2019).

focus on exploring dynamic context (motion and agent-to-agent interaction) for learning road users' behavior. On the one hand, as discussed earlier in the framework of MCENet (Sec. 6.1.4), considering static scene context of agent-to-environment interaction to some extent may lead to the models being overfitted to a particular space and impair the models' generalizability for predicting trajectories in new spaces. On the other hand, how to effectively incorporate static context and maintain generalizability still remains an unsolved problem. Indeed, this will be a very interesting direction for future research work.

6.3.5 Summary

In this section, a novel framework Dynamic Context Encoder Network (DCENet) has been proposed for multi-path trajectory prediction for heterogeneous agents in various real-world traffic scenarios. Learning of dynamic spatial–temporal context is decomposed into learning temporal context between steps using the Transformer encoder and exploiting the dynamic spatial context between agents using the the self-attention architectures with the following LSTM encoder. The spatial–temporal context is encoded into a latent space using a CVAE module. A set of future trajectories for each agent is predicted conditioned on the spatial–temporal context using the trained CVAE module. DCENet was evaluated on the TrajNet benchmark and achieved a new state-of-the-art performance on the leader board at the time of its submission. Its superior performance on the inD benchmark further validated its efficacy and generalizability. Meanwhile, ablation studies investigated the impact of each module in DCENet.

The following summarizes improvements made from the previous AMENet model to the DCENet model introduced in this section. Methodologically, on the basis of AMENet, DCENet inherits the self-attention mechanism with the LSTM encoder from AMENet for exploiting dynamic context of agent-to-agent interaction, but further extends the self-attention architecture for exploiting temporal context, i. e., the motion of the target agent. Combining these two streams contributes a clear improvement of performance tested on the TrajNet benchmark of shared spaces. Experimentally, DCENet was further evaluated on predicting trajectories in intersections of the inD benchmark and achieved superior performance.

However, similar to AMENet, scene context information is not considered in DCENet. In future work, this framework should be extended to effectively explore scene context without sacrificing generalizability, an unsolved problem left by AMENet.

7 Conclusion and Outlook

7.1 Conclusion

In this thesis, road users' behavior in shared spaces has been studied using deep learning methods. The temporarily shared spaces of intersections and shared spaces as a traffic design are two types of traffic environment commonly seen in urban areas in many countries. The former is proposed to allow vehicles to turn and directly interact with other crossing road users while the latter aims to reduce the dominance of vehicles and improve pedestrian movement and comfort.

The uncertainty of road users' behavior is increased in such shared, yet ambiguous environments, e. g., where road users have to negotiate with each other. Understanding how road users behave in such environments is essential, not only for safety analysis of road users' behavior, but also because of the foreseeable advent of autonomous driving in urban areas as well as for the creation of intelligent systems for traffic management. With the aim of automatically learning road users' behavior, this thesis has proposed the use of conditional generative models for automated interaction detection in the temporarily shared spaces of intersections and trajectory prediction in shared spaces as a traffic design. The uncertainty of road users' behavior is encoded into the so-called latent space with a set of stochastic variables, which in turn can be employed to map one deterministic input (i. e., an observation of a road user's past motion) to many possible future behavior patterns.

Video data is used as input to train a sequence-to-sequence model based on a Conditional Variational Auto-Encoder (CVAE) for interaction detection between vehicles and VRUs at intersections. The object information (i. e., road user's location and type) is detected using state-of-the-art deep learning methods and the motion information is captured by optical flow. Vehicle turning sequences of varying length are accurately classified—depending on whether or not interaction is required between a turning vehicle and any involved vulnerable road users (VRUs)—conditioned on the above information extracted from video data. The model also provides a frame-wise probability representing how the intensity of interaction between a turning vehicle and VRUs evolves over time. In addition, with the multi-sampling process from the latent space trained for encoding various behavior patterns, the model generates divergent predictions at each video frame and the variance of the predictions is used to quantify the uncertainty level of the model's output. This process provides a clue in what way the output of the model can be trusted, especially when the model is uncertain facing a highly non-deterministic situation.

The model was validated using real-world traffic data acquired from different intersections and was proven to be effective. It achieved an F1-score above 0.96 at a busy right-turn intersection in Germany and 0.89 at an extremely busy left-turn intersection in Japan. Its efficacy implies that object and optical flow information directly extracted from video data can be used to learn how road users interact with each other. Unlike many other approaches based on trajectory analysis, the proposed model provides an alternative solution for automated interaction detection.

Various factors and state-of-the-art deep learning architectures are investigated for trajectory prediction. In this thesis, three frameworks based on CVAE are proposed for accurate multi-path trajectory prediction of heterogeneous road user agents in shared spaces. Similar to the interaction detection model described above, the latent space of the CVAE is trained for encoding various behavior patterns, and the multi-sampling process from the trained latent space enables

the frameworks to generate not only one deterministic future trajectory, but multiple possible future trajectories for each agent instead. In comparison to single-path trajectory prediction, multi-path trajectory prediction increases the possibility of correctly detecting an agent's intent, i. e., the multiple predictions form into an area indicating the potential intent of the agent, and the size of the area reflects the uncertainty of the agent's intent. Meanwhile, a ranking method based on a bivariate Gaussian distribution is proposed to select the mostlikely prediction out of the multiple predictions.

The first framework, named Multi-Context Encoder Network (MCENet), focuses on studying multiple contexts for trajectory prediction. MCENet incorporates scene context, interaction context with pedestrian grouping, and motion information to mimic how an agent adapts its movement accordingly. Particularly, the impact of three types of scene context, i. e., heat map, aerial photograph, and accessibility map, are studied. On the one hand, this approach indicates that rich scene context information improves the accuracy of trajectory prediction measured by the average and final displacement errors in comparison to the ground truth. On the other hand, however, the scene context information is rather space-dependent. Consequently, the model's generalizability is limited, i. e., MCENet trained using the data from one shared space does not generalize well in another shared space.

The second framework, named Attentive Maps Encoder Network (AMENet), focuses on improving the generalizability for predicting accurate trajectories in various unseen shared spaces. First, AMENet does not explore any scene context information in order to keep the model from overfitting to a particular scene. Second, it introduces a novel module, i. e., attentive dynamic maps for agent-to-agent interaction modeling. The interaction module learns spatio–temporal interconnections between agents considering their orientation, speed, and position in relation to the target agent at each step. Meanwhile, the self-attention mechanism (Vaswani et al., 2017) enables the module to automatically focus on the salient features extracted over different steps. The efficacy of AMENet was proven by the superior performance tested on the Trajenet benchmark (Sadeghian et al., 2018a) for predicting trajectories in 20 distinct unseen shared spaces. It not only achieved state-of-the-art performance, but also won the first place on the leader board of the open challenge at the time of its submission.

The third framework, named Dynamic Context Encoder (DCENet), focuses on improving the accuracy for predicting trajectories in unseen shared spaces, as well as at intersections of mixed traffic. DCENet is an advanced version of AMENet and uses dedicated self-attention structures to explore dynamic context, i. e., motion and agent-to-agent interaction. AMENet uses Long Short-Term Memory network for learning temporal context from motion information, whereas DCENet adopts a two-stream architecture with one stream dedicated to learning spatial context from dynamic maps, and another stream dedicated to learning temporal context from motion information. The superior performance of DCENet was proven on two large-scale benchmarks. DCENet outperformed AMENet and two other very recent models in the TrajNet open challenge. Moreover, it achieved superior performance tested on the inD intersection benchmark (Bock et al., 2019) of mixed traffic, surpassing AMENet as well as several highly cited models.

7.2 Outlook

Considering the implications of intelligent technology for the future of traffic management, the proposed methods for learning road users' behavior in shared spaces provide a starting point for further study of both the efficacy and possible applications of the models presented in this thesis. There are—at the very least—two potential directions for this future research which can be outlined as follows: (1) improvement of the interaction detection and trajectory prediction methods, and (2) joint applications of interaction detection and trajectory prediction for safety analysis.

Interaction detection using the current model introduced in this thesis is limited with respect to domain adaptation. For example, the model trained on the dataset of the left-turn intersection in Japan cannot be smoothly transferred to the dataset of the right-turn intersection in Germany. This is due to the fact that these two datasets are very different in terms of, such as vehicle travel direction, camera perspective, frame size and rate, sequence length, intersection layout, traffic density, or cultural factors (Germany vs. Japan). Projective transformation, to some extent, might be a solution to the problems caused by the position of the camera. But many other factors, e. g., driving behavior and traffic rules, have to be taken into consideration as well.

The trajectory prediction frameworks introduced in this thesis come with the following limitations. First, in order to prevent overfitting, scene context information is not fully used for trajectory prediction by AMENet and DCENet. However, scene context provides important information about road users' behavior, i. e., agents adapt their movement to avoid collisions with obstacles. The challenge lies in the question of how to guide an autonomous agent through a specific scene and then teach it to adapt its movement in different scenes. Second, collision avoidance is automatically learned from the dynamic maps based on trajectory data. The empirical results have shown that the proposed model is able to learn interactions between agents, and that only very few predictions (0.3 %) lead to collisions. Nonetheless, from the perspective of traffic safety, collisions should be completely avoided. Taking this into account, however, the question of how to "hard-code" rules of collision avoidance for agent-to-environment and agent-to-agent conflicts into a deep learning model remains unanswered. In this regard, Reinforcement Learning might be an alternative solution to automatically embed these rules into a model for predicting collision-free trajectories, or hybrid models with manually designed rules, such as combining deep learning and Social Force models for collision avoidance (Johora et al., 2020).

The interaction detection and trajectory prediction methods introduced in this thesis can be jointly applied for safety analysis, recognizing the road users' past behavior and forecasting their near future behavior. Fig. 7.1 provides a conceptual pipeline for the potential combination. It contains four components: interaction detection, trajectory tracking, trajectory prediction, and safety analysis.

The interaction detection model can be extended to automatically classify levels of conflict severity (Sayed and Zein, 1999). The detected severity levels, in turn, serve as criteria for estimating whether a prompt precocious action, e. g., via a warning sound or a sign, should be taken to guide a road user's behavior.

Since the severity level does not always remain static as the situation evolves, safety analysis does not only have to correctly understand the current situation based on observation, it also has to predict what will happen next. Trajectory prediction provides an insight into the development of the traffic situation. Based on the road users' past behavior, it foresees the intent of the involved road users in the near future. The trajectory prediction frameworks can be extended for safety analysis, e. g., by using the predicted trajectories to calculate time-to-collision (Perkins and Harris,

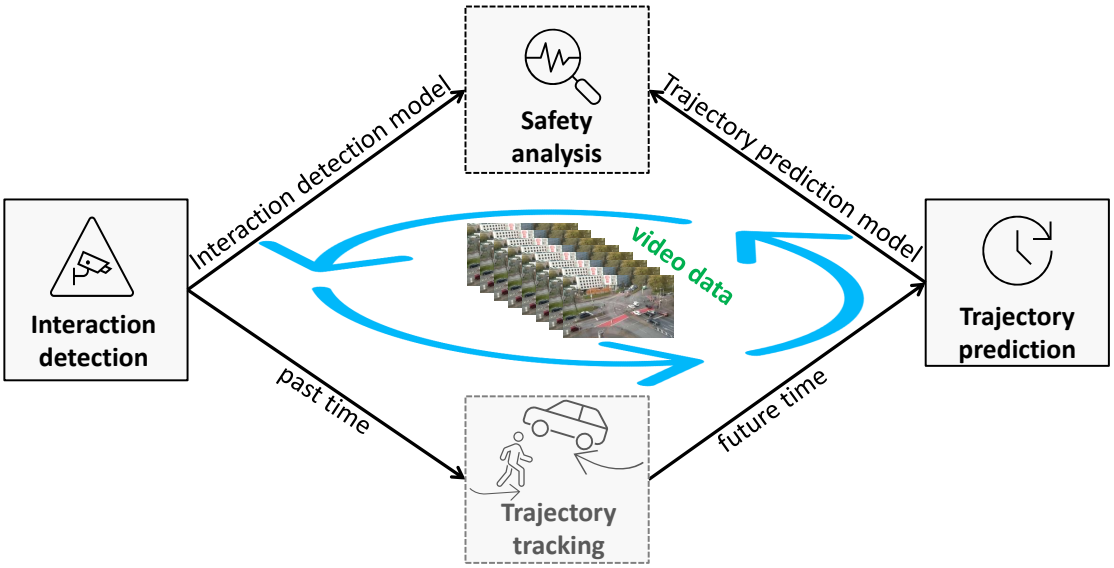


Figure 7.1: The conceptual pipeline for safety analysis considering both the road users' past and their future behavior.

1968) and detecting abnormal trajectories by comparing the anticipated/predicted trajectories with the actual ones.

Furthermore, safety analysis should be done in an iterative way, i. e., estimating road users' behavior based observation of their past behavior, predicting their future behavior in order to detect potential conflicts, and continuously updating this process whenever new information becomes available as time unfolds.

However, an intermediate component, trajectory tracking, needs to be added to connect the interaction detection and trajectory prediction methods. The interaction detection model proposed in this thesis learns road users' behavior directly from video data without any need for trajectories. But all the trajectory prediction frameworks proposed in this thesis predict future trajectories based on the observed past trajectories. Therefore, automatically tracking road users from video data and extracting their trajectories is an important step to achieve safety analysis by considering road users' past behavior as well as their possible future behavior in varying traffic situations.

List of Figures

List of Tables

List of Acronyms

TOR	Turn-on-Red
VRUs	Vulnerable Road Users
ITS	Intelligent Transportation Systems
GANs	Generative Adversarial Networks
VAE	Variational Auto-Encoder
CVAE	Conditional Variational Auto-Encoder
IOU	Intersection of Union
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks
LSTMs	Long Short-Term Memories
GRU	Gated Recurrent Unit
KDE	Kernel Density Estimation
TTC	Time-to-Collision
PET	Post-Encroachment Time
DST	Deceleration-to-Safety Time
GT	Gap Time
CA	Cellular Automata
SFM	Social Force Model

Nomenclature

Conditional Generative Model

$\boldsymbol{X}$	vector of input data
$\boldsymbol{Y}$	vector of output
$\mathbb{E}$	expectation
$\mathbb{N}_0$	non-negative natural number
$\mathbf{I}$	identity matrix
$\mathbf{X}$	set of input data
$\mathbf{Y}$	set of output data
$\mathbf{z}$	set of latent variables
$\mathcal{L}$	reconstruction loss
$\mathcal{N}$	normal distribution
μ	mean value
ϕ	variational parameters
σ	standard deviation
θ	generative parameters
D_{KL}	Kullback-Leibler divergence loss
f	function
$g_\phi(\,.\,)$	re-parameterization function with the variation parameter ϕ
i	index of a road user or an agent
J	number of dimensionality
L	total number of samples
l	index of a sample
N	positive integer number
$p(\boldsymbol{X})$	marginal probability of $\boldsymbol{X}$
$p(\mathbf{z})$	prior distribution of $\mathbf{z}$
T	observed time steps or sequence length
t	observed time step or frame index

T' predicted time steps or sequence length

t' predicted time step or frame index

T^* fixed sequence length for observed time steps

z latent variable

DBSCAN

ϵ upper threshold of distance

$\mathbf{N}_\epsilon$ ϵ-neighborhood

MinPts the minimum number of points

p point p

q point q

Interaction Detection

$\boldsymbol{X}$ input feature vector

$\boldsymbol{Y}$ ground truth label vector

Γ_i degree of uncertainty for the prediction of sequence i

$\lambda(\,.\,)$ weighting function

$\mathbb{R}^d$ dimensionality of an input feature vector

$\mathcal{H}$ cross entropy

$\mathcal{L}$ loss function

w size of sliding window

ω normalization parameter

h smoothing parameter, or bandwidth

K Gaussian kernel function

k number of sub-sequences

X^t input feature at time step t

Y^t ground truth label at time step t

Deep Learning

$\boldsymbol{x}$ input of a neural network

$\boldsymbol{y}$ output of a neural network

$\mathbf{b}$ bias of a layer

$\mathbf{U}$ weights of a layer

$\mathbf{W}$ weights of a layer

$\mathbf{z}$	intermediate output of a linear transformation
σ	activation function
$\tilde{c}$	memory cell of a Long Short-Term Memory layer before input gate
c	memory cell of a Long Short-Term Memory layer
$C(\,.\,)$	cost or loss function
E	error between a neural network's output and ground truth
f	forget gate of a Long Short-Term Memory layer
h	output of a hidden layer
i	input gate of a Long Short-Term Memory layer
l	index of the l-th layer
n	index of a data point
o	output gate of a Long Short-Term Memory layer
t	index of a time step

Optical Flow

(x, y)	coordinates of a point
$\mathbf{\Delta x}$	neighborhood of $\mathbf{x}$
$\mathbf{A}$	a symmetric matrix
$\mathbf{b}$	a vector
$\mathbf{d}$	a global displacement
$\mathbf{T}$	transpose operation
$\mathbf{x}$	a 2D vector in a local coordinate system
c	a scalar
$e(\cdot)$	error function
$f(\cdot)$	brightness function
I	a neighborhood for a point
r_i	the i-th coefficient of a polynomial
t	time step
u	velocity in x-axis
v	velocity in y-axis
$w(\cdot)$	weighting function

Trajectory Prediction

α	relative angle between target and neighboring agents
$\boldsymbol{I}$	notation for agent-to-agent interaction
$\boldsymbol{S}$	notation for agent-to-environment interaction
$\mathsf{G}(i)$	a set of group members for the target agent i
$\mathsf{N}(i)$	a set of neighboring agents for the target agent i
$\odot$	element-wise multiplication
$\pi(\,.\,)$	mapping function for interactions
ζ	coexisting time ratio in the same cluster
I	notation for interaction of a given agent
O	notation for the orientation layer of a dynamic map
P	notation for the position layer of a dynamic map
R	radius for an occupancy grid
r	relative distance between target and neighboring agents
S	notation for the speed layer of a dynamic map
$W\,, H$	width and height of a dynamic map
$w\,, h$	relative position regarding the width and height of a dynamic map
$x\,, y$	local Cartesian coordinates

Transformer Network

$\boldsymbol{h}_t$	representation or hidden state at time step t
$\boldsymbol{x}_t$	input at time step t
$\boldsymbol{y}_t$	output at time step t
$\mathbf{p}^t$	position encoding at time step t
$\mathbf{T}$	transpose operation
$\mathbf{X}$	input vector
d, D	d-th dimension of the total dimensionality of D
d_K	dimensionality of the key vector
$d_{\mathbf{X}}$	dimensionality of the input vector
h	total number of heads
i	the i-th head
K	key vector
Q	query vector

V value vector

W_Q, W_K, W_V trainable parameters respective to the Q, K and V vectors

$f(\cdot)$, $g(\cdot)$ element-wise non-liner transformation functions

YOLO

IOU Intersection of Union

$P(\cdot)$ probability function

Bibliography

Al-Molegi, A., Jabreel, M., Martinez-Balleste, A., 2018. Move, attend and predict: An attention-based neural model for people's movement prediction. Pattern Recognition Letters 112, 34–40.

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

Alhajyaseen, W.K., Asano, M., Nakamura, H., 2012. Estimation of left-turning vehicle maneuvers for the assessment of pedestrian safety at intersections. IATSS research 36, 66–74.

Allen, B.L., Shin, B.T., Cooper, P.J., 1978. Analysis of traffic conflicts and collision. Transportation Research Record 677, 67–74.

Amirian, J., Hayet, J.B., Pettré, J., 2019. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE. pp. 2964–2972.

Anvari, B., Bell, M.G., Sivakumar, A., Ochieng, W.Y., 2015. Modelling shared space users via rule-based social force model. Transportation Research Part C: Emerging Technologies 51, 83–103.

Archer, J., 2004. Methods for the assessment and prediction of traffic safety at urban intersections and their application in micro-simulation modelling. Royal institute of technology .

Asano, M., Iryo, T., Kuwahara, M., 2010. Microscopic pedestrian simulation model combined with a tactical model for route choice behaviour. Transportation Research Part C: Emerging Technologies 18, 842–855.

Aveni, A.F., 1977. The not-so-lonely crowd: Friendship groups in collective behavior. Sociometry , 96–99.

Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer normalization. stat 1050, 21.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015.

Bandini, S., Crociani, L., Feliciani, C., Gorrini, A., Vizzari, G., 2017a. Collision avoidance dynamics among heterogeneous agents: The case of pedestrian/vehicle interactions, in: Conference of the Italian Association for Artificial Intelligence, Springer. pp. 44–57.

Bandini, S., Crociani, L., Vizzari, G., 2017b. An approach for managing heterogeneous speed profiles in cellular automata pedestrian models. Journal of Cellular Automata 12.

Bartoli, F., Lisanti, G., Ballan, L., Del Bimbo, A., 2018. Context-aware trajectory prediction, in: 2018 24th International Conference on Pattern Recognition (ICPR), pp. 1941–1946.

Becker, S., Hug, R., Hübner, W., Arens, M., 2018. An evaluation of trajectory prediction approaches and notes on the trajnet benchmark. arXiv preprint arXiv:1805.07663 .

Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5, 157–166.

Bisagno, N., Zhang, B., Conci, N., 2018. Group lstm: Group trajectory prediction in crowded scenarios, in: European Conference on Computer Vision (ECCV), Springer. pp. 213–225.

Bjørnskau, T., 2017. The zebra crossing game–using game theory to explain a discrepancy between road user behaviour and traffic rules. Safety science 92, 298–301.

Bock, J., Krajewski, R., Moers, T., Runde, S., Vater, L., Eckstein, L., 2019. The inD dataset: A drone dataset of naturalistic road user trajectories at German intersections, in: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE. pp. 1929–1934.

Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010. Springer, pp. 177–186.

Burstedde, C., Klauck, K., Schadschneider, A., Zittartz, J., 2001. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. Physica A: Statistical Mechanics and its Applications 295, 507–525.

Campbell, D.T., 1958. Common fate, similarity, and other indices of the status of aggregates of persons as social entities. Behavioral science 3, 14–25.

Chai, C., Wong, Y.D., 2015. Fuzzy cellular automata model for signalized intersections. Computer-Aided Civil and Infrastructure Engineering 30, 951–964.

Cheng, H., Feng, L., Liu, H., Hirayama, T., Murase, H., Sester, M., 2021a. Interaction detection between vehicles and vulnerable road users: A deep generative approach with attention. arXiv preprint arXiv:2105.03891 .

Cheng, H., Johora, F.T., Sester, M., Müller, J.P., 2020a. Trajectory modelling in shared spaces: Expert-based vs. deep learning approach?, in: Swarup, S., Savarimuthu, B.T.R. (Eds.), Multi-Agent-Based Simulation XXI - 21st International Workshop, MABS 2020, Auckland, New Zealand, May 10, 2020, Revised Selected Papers, Springer. pp. 13–27.

Cheng, H., Li, Y., Sester, M., 2019. Pedestrian group detection in shared space, in: Proceedings of the 30th IEEE Intelligent Vehicles Symposium (IV), IEEE. pp. 1707–1714.

Cheng, H., Liao, W., Tang, X., Yang, M.Y., Sester, M., Rosenhahn, B., 2021b. Exploring dynamic context for multi-path trajectory prediction, in: 2021 IEEE international Conference on Robotics and Automation (ICRA), IEEE.

Cheng, H., Liao, W., Yang, M.Y., Rosenhahn, B., Sester, M., 2021c. Amenet: Attentive maps encoder network for trajectory prediction. ISPRS Journal of Photogrammetry and Remote Sensing 172, 253–266.

Cheng, H., Liao, W., Yang, M.Y., Sester, M., Rosenhahn, B., 2020b. MCENET: Multi-context encoder network for homogeneous agent trajectory prediction in mixed traffic, in: 2020 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 1–8.

Cheng, H., Liu, H., Shinmura, F., Akai, N., Murase, H., Hirayama, T., 2020c. Automatic interaction detection between vehicles and vulnerable road users during turning at an intersection, in: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 912–918.

Cheng, H., Sester, M., 2018a. Mixed traffic trajectory prediction using lstm–based models in shared space, in: The Annual International Conference on Geographic Information Science, Springer. pp. 309–325.

Cheng, H., Sester, M., 2018b. Modeling mixed traffic in shared space using lstm with probability density mapping, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 3898–3904.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734.

Choi, E.H., 2010. Crash factors in intersection-related crashes: An on-scene perspective. Technical Report HS-811 366.

Clarke, E., 2006. Shared space-: the alternative approach to calming traffic. Traffic Engineering & Control 47, 290–292.

Compton, R., Milton, E., 1994. Safety impact of permitting right-turn-on-red. National Highway Traffic Safety Administration, Report No. DOT-HS-808 .

Cooper, P., 1984. Experience with traffic conflicts in canada with emphasis on "post encroachment time" techniques, in: International calibration study of traffic conflict techniques. Springer, pp. 75–96.

Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise., in: KDD, pp. 226–231.

Farnebäck, G., 2003. Two-frame motion estimation based on polynomial expansion, in: Scandinavian conference on Image analysis, Springer. pp. 363–370.

Ferryman, J., Shahrokni, A., 2009. Pets2009: Dataset and challenge, in: International workshop on performance evaluation of tracking and surveillance, pp. 1–6.

Fleck, J.L., Yee, B.M., 2002. Safety evaluation of right turn on red. ITE journal 72, 46–48.

Franke, U., Gavrila, D., Görzig, S., Lindner, F., Paetzold, F., Wöhler, C., 1998. Autonomous driving goes downtown. Intelligent Systems , 40–48.

Gallagher, H.L., Frith, C.D., 2003. Functional imaging of 'theory of mind'. Trends in cognitive sciences 7, 77–83.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 32, 1231–1237.

Gérin-Lajoie, Martand Richards, C.L., McFadyen, B.J., 2005. The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space. Motor control 9, 242–269.

Gerlach, J., Methorst, R., Boenke, D., Leven, J., 2009. Sense and nonsense about shared space-for an objective view of a popular planning concept. Luettavissa .

Gindele, T., Brechtel, S., Dillmann, R., 2010. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments, in: 13th International IEEE Conference on Intelligent Transportation Systems, IEEE. pp. 1625–1631.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 580–587.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2015. Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence 38, 142–158.

Giuliari, F., Hasan, I., Cristani, M., Galasso, F., 2021. Transformer networks for trajectory forecasting, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE. pp. 10335–10342.

Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep learning. volume 1. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: In Proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 2672–2680.

Graves, A., 2013. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 .

Graves, A., Fernández, S., Gomez, F., Schmidhuber, J., 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, in: Proceedings of the 23rd international conference on Machine learning, pp. 369–376.

Gupta, A., Johnson, J., Li, F.F., Savarese, S., Alahi, A., 2018. Social gan: Socially acceptable trajectories with generative adversarial networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 2255–2264.

Habibovic, A., Davidsson, J., 2011. Requirements of a system to reduce car-to-vulnerable road user crashes in urban intersections. Accident Analysis & Prevention 43, 1570–1580.

Hamilton-Baillie, B., 2004. Urban design: Why don't we do it in the road? Modifying traffic behavior through legible urban design. Journal of Urban Technology 11, 43–62.

Hamilton-Baillie, B., 2008. Shared space: Reconciling people, places and traffic. Built environment 34, 161–181.

Hamilton-Baillie, B., Jones, P., 2005. Improving traffic behaviour and safety through urban design, in: Proceedings of the Institution of Civil Engineers-Civil Engineering, Thomas Telford Ltd. pp. 39–47.

Harvey, A.C., et al., 1990. Forecasting, structural time series models and the kalman filter. Cambridge Books .

Hasan, I., Setti, F., Tsesmelis, T., Del Bue, A., Galasso, F., Cristani, M., 2018. Mx-lstm: mixing tracklets and vislets to jointly forecast trajectories and head poses, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 6067–6076.

Hayward, J.C., 1972. Near-miss determination through use of a scale of danger. Highway Research Record , 24–34.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 770–778.

Helbing, D., Molnar, P., 1995. Social force model for pedestrian dynamics. Physical review E 51, 4282.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

Horn, B.K., Schunck, B.G., 1981. Determining optical flow. Artificial intelligence 17, 185–203.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 .

Hoy, M., Tu, Z., Dang, K., Dauwels, J., 2018. Learning to predict pedestrian intention via variational tracking networks, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 3132–3137.

Hupfer, C., 1997. Deceleration to safety time (dst)-a useful figure to evaluate traffic safety, in: ICTCT Conference Proceedings of Seminar, pp. 5–7.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning PMLR, pp. 448–456.

Ismail, K., Sayed, T., Saunier, N., 2010. Automated analysis of pedestrian–vehicle conflicts: Context for before-and-after studies. Transportation research record 2198, 52–64.

Ismail, K., Sayed, T., Saunier, N., Lim, C., 2009. Automated analysis of pedestrian–vehicle conflicts using video data. Transportation research record 2140, 44–54.

Jackson, S., Miranda-Moreno, L.F., St-Aubin, P., Saunier, N., 2013. Flexible, mobile video camera system and open source video analysis software for road safety and behavioral analysis. Transportation research record 2365, 90–98.

Johora, F.T., Cheng, H., Müller, J.P., Sester, M., 2020. An agent-based model for trajectory modelling in shared spaces: A combination of expert-based and deep learning approaches, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1878–1880.

Johora, F.T., Müller, J.P., 2018. Modeling interactions of multimodal road users in shared spaces, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 3568–3574.

Kaparias, I., Bell, M.G., Dong, W., Sastrawinata, A., Singh, A., Wang, X., Mount, B., 2013. Analysis of pedestrian–vehicle traffic conflicts in street designs with elements of shared space. Transportation research record 2393, 21–30.

Kaparias, I., Bell, M.G., Greensted, J., Cheng, S., Miri, A., Taylor, C., Mount, B., 2010. Development and implementation of a vehicle–pedestrian conflict analysis method: adaptation of a vehicle–vehicle technique. Transportation research record 2198, 75–82.

Kingma, D.P., Mohamed, S., Rezende, D.J., Welling, M., 2014. Semi-supervised learning with deep generative models, in: In Proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 3581–3589.

Kingma, D.P., Welling, M., 2014. Auto-encoding variational bayes, in: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada.

Kitani, K.M., Ziebart, B.D., Bagnell, J.A., Hebert, M., 2012. Activity forecasting, in: European Conference on Computer Vision (ECCV), pp. 201–214.

Klinger, T., Rottensteiner, F., Heipke, C., 2017. Probabilistic multi-person localisation and tracking in image sequences. ISPRS Journal of Photogrammetry and Remote Sensing 127, 73–88.

Koetsier, C., Busch, S., Sester, M., 2019. Trajectory extraction for analysis of unsafe driving behaviour. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 42 (2019), Nr. 2/W13 42, 1573–1578.

Laube, P., Imfeld, S., Weibel, R., 2005. Discovering relative motion patterns in groups of moving point objects. International Journal of Geographical Information Science 19, 639–668.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation 1, 541–551.

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

Lerner, A., Chrysanthou, Y., Lischinski, D., 2007. Crowds by example, in: Computer Graphics Forum, Wiley Online Library. pp. 655–664.

Loftsgaarden, D.O., Quesenberry, C.P., et al., 1965. A nonparametric estimate of a multivariate density function. The Annals of Mathematical Statistics 36, 1049–1051.

Luong, M.T., Pham, H., Manning, C.D., 2015. Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421.

Massaad, A., Massaad, Z., 2020. Right-turn-on-red impact assessment and volume estimation model for critical intersections. International Journal of Transportation Science and Technology .

McGee, H., Warren, D., 1976. Right turn on red. Public Roads 40.

Methorst, R., Gerlach, J., Boenke, D., Leven, J., 2007. Shared space: safe or dangerous. A contribution to objectification of a popular design philosophy 3.

Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 .

Mohajerin, N., Rohani, M., 2019. Multi-step prediction of occupancy grid maps with recurrent neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 10600–10608.

Mohanan, M., Salgoankar, A., 2018. A survey of robotic motion planning in dynamic environments. Robotics and Autonomous Systems 100, 171–185.

Morris, B.T., Trivedi, M.M., 2008. A survey of vision-based trajectory learning and analysis for surveillance. Transactions on circuits and systems for video technology 18, 1114–1127.

Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., Theraulaz, G., 2010. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. PloS one 5, e10047.

Nagel, K., Schreckenberg, M., 1992. A cellular automaton model for freeway traffic. Journal de physique I 2, 2221–2229.

Ni, Y., Wang, M., Sun, J., Li, K., 2016. Evaluation of pedestrian safety at intersections: A theoretical framework based on pedestrian-vehicle interaction patterns. Accident Analysis & Prevention 96, 118–129.

Osborne, M.J., Rubinstein, A., 1994. A course in game theory. MIT press.

Parker Jr, M., Zegeer, C.V., 1989. Traffic conflict techniques for safety and operations: Observers manual. Technical Report. United States. Federal Highway Administration.

Parzen, E., 1962. On estimation of a probability density function and mode. The annals of mathematical statistics 33, 1065–1076.

Pascucci, F., Rinke, N., Schiermeyer, C., Berkhahn, V., Friedrich, B., 2017. A discrete choice model for solving conflict situations between pedestrians and vehicles in shared space. arXiv preprint arXiv:1709.09412 .

Pellegrini, S., Ess, A., Schindler, K., Van Gool, L., 2009. You'll never walk alone: Modeling social behavior for multi-target tracking, in: International Conference on Computer Vision (ICCV), pp. 261–268.

Perkins, S.R., Harris, J.L., 1968. Traffic conflict characteristics-accident potential at intersections. Highway Research Record .

Preusser, D.F., Leaf, W.A., DeBartolo, K.B., Blomberg, R.D., Levy, M.M., 1982. The effect of right-turn-on-red on pedestrian and bicyclist accidents. Journal of safety research 13, 45–55.

Rasouli, A., Kotseruba, I., Tsotsos, J.K., 2017. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 206–213.

Rasouli, A., Tsotsos, J.K., 2019. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. IEEE Transactions on Intelligent Transportation Systems 21, 900–918.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 779–788.

Redmon, J., Farhadi, A., 2017. Yolo9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 7263–7271.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 .

Reid, S., 2009. DfT Shared Space Project Stage 1: Appraisal of Shared Space. MVA Consultancy.

Rezende, D.J., Mohamed, S., Wierstra, D., 2014. Stochastic backpropagation and approximate inference in deep generative models, in: Proceedings of the 31st International Conference on Machine Learning (ICML), JMLR.org. pp. 1278–1286.

Rinke, N., Schiermeyer, C., Pascucci, F., Berkhahn, V., Friedrich, B., 2017. A multi-layer social force approach to model interactions in shared spaces using collision prediction. Transportation Research Procedia 25, 1249–1267.

Robicquet, A., Sadeghian, A., Alahi, A., Savarese, S., 2016. Learning social etiquette: Human trajectory understanding in crowded scenes, in: European Conference on Computer Vision (ECCV), pp. 549–565.

Rudenko, A., Palmieri, L., Herman, M., Kitani, K.M., Gavrila, D.M., Arras, K.O., 2020. Human motion trajectory prediction: A survey. The International Journal of Robotics Research 39, 895–935.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. nature 323, 533–536.

Sadeghian, A., Kosaraju, V., Gupta, A., Savarese, S., Alahi, A., 2018a. Trajnet: Towards a benchmark for human trajectory prediction. arXiv preprint .

Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., Savarese, S., 2019. Sophie: An attentive gan for predicting paths compliant to social and physical constraints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1349–1358.

Sadeghian, A., Legros, F., Voisin, M., Vesel, R., Alahi, A., Savarese, S., 2018b. Car-net: Clairvoyant attentive recurrent network, in: European Conference on Computer Vision (ECCV), pp. 151–167.

Salamati, K., Schroeder, B., Rouphail, N.M., Cunningham, C., Long, R., Barlow, J., 2011. Development and implementation of conflict-based assessment of pedestrian safety to evaluate accessibility of complex intersections. Transportation research record 2264, 148–155.

Saunier, N., Sayed, T., 2006. A feature-based tracking algorithm for vehicles in intersections, in: The 3rd Canadian Conference on Computer and Robot Vision (CRV'06), IEEE. pp. 59–59.

Saunier, N., Sayed, T., 2008. Probabilistic framework for automated analysis of exposure to road collisions. Transportation research record 2083, 96–104.

Sayed, T., Zaki, M.H., Autey, J., 2013. Automated safety diagnosis of vehicle–bicycle interactions using computer vision analysis. Safety science 59, 163–172.

Sayed, T., Zein, S., 1999. Traffic conflict standards for intersections. Transportation Planning and Technology 22, 309–323.

Schindler, K., Ess, A., Leibe, B., Van Gool, L., 2010. Automatic detection and tracking of pedestrians from a moving stereo rig. ISPRS Journal of Photogrammetry and Remote Sensing 65, 523–537.

Schönauer, R., 2017. A Microscopic Traffic Flow Model for Shared Space. Ph.D. thesis. Graz University of Technology.

Sester, M., Feuerhake, U., Kuntzsch, C., Zhang, L., 2012. Revealing underlying structure and behaviour from movement data. KI-Künstliche Intelligenz 26, 223–231.

Shirazi, M.S., Morris, B.T., 2016. Looking at intersections: A survey of intersection monitoring, behavior and safety analysis of recent studies. IEEE Transactions on Intelligent Transportation Systems 18, 4–24.

Simonyan, K., Zisserman, A., 2014a. Two-stream convolutional networks for action recognition in videos, in: Advances in neural information processing systems, pp. 568–576.

Simonyan, K., Zisserman, A., 2014b. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

Sohn, K., Lee, H., Yan, X., 2015. Learning structured output representation using deep conditional generative models, in: In Proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 3483–3491.

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, pp. 3104–3112.

Suzuki, K., Nakamura, H., 2006. Trafficanalyzer-the integrated video image processing system for traffic flow analysis, in: Proceedings of the 13th ITS World Congress, London, 8-12 October 2006.

Svensson, Å., Hydén, C., 2006. Estimating the severity of safety related behaviour. Accident Analysis & Prevention 38, 379–385.

Tageldin, A., Sayed, T., 2016. Developing evasive action-based indicators for identifying pedestrian conflicts in less organized traffic environments. Journal of Advanced Transportation 50, 1193–1208.

Taoka, G.T., 1989. Brake reaction times of unalerted drivers. ITE Journal 59, 19–21.

Tarko, A.P., 2001. Predicting right turns on red. Transportation research record 1776, 138–142.

Tay, M.K.C., Laugier, C., 2008. Modelling smooth paths using gaussian processes, in: Field and Service Robotics, pp. 381–390.

Tomasi, C., Kanade, T., 1991. Detection and tracking of point features. Technical Report. School of Computer Science, Carnegie Mellon Univ. Pittsburgh.

Varshneya, D., Srinivasaraghavan, G., 2017. Human trajectory prediction using spatially aware deep attention models. arXiv preprint arXiv:1705.09436 .

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: In Proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 5998–6008.

Vemula, A., Muelling, K., Oh, J., 2018. Social attention: Modeling attention in human crowds, in: In Proceedings of International Conference on Robotics and Automation (ICRA), IEEE. pp. 1–7.

Wang, X., Girshick, R., Gupta, A., He, K., 2018. Non-local neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 7794–7803.

Wong, Y.D., Ho, J., Foo, H., 2004. Left-turn-on-red traffic scheme in Singapore. Institute of Transportation Engineers. ITE Journal 74, 24.

Xu, Kelvand Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention, in: Proceedings of the 32ed International Conference on Machine Learning (ICML), pp. 2048–2057.

Xue, H., Huynh, D.Q., Reynolds, M., 2018. Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction, in: In Proceedings of Winter Conference on Applications of Computer Vision (WACV), pp. 1186–1194.

Yamaguchi, K., Berg, A.C., Ortiz, L.E., Berg, T.L., 2011. Who are you with and where are you going?, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 1345–1352.

Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K., 2018. Convolutional neural networks: an overview and application in radiology. Insights into imaging 9, 611–629.

Yang, D., Özgüner, Ü., Redmill, K., 2018. Social force based microscopic modeling of vehicle-crowd interaction, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE. pp. 1537–1542.

Yi, Q., Chen, X., Li, D., et al., 2012. Evaluating Safety Performance and Developing Guidelines for the Use of Right Turn on Red (RTOR). Technical Report. Southwest Region University Transportation Center (US).

Yilmaz, A., Javed, O., Shah, M., 2006. Object tracking: A survey. Acm computing surveys (CSUR) 38, 13.

Zeng, W., Nakamura, H., Chen, P., 2014. A modified social force model for pedestrian behavior simulation at signalized crosswalks. Procedia-Social and Behavioral Sciences 138, 521–530.

Zhang, P., Ouyang, W., Zhang, P., Xue, J., Zheng, N., 2019. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 12085–12094.

Zhang, Y., Duan, H., 2007. Modeling mixed traffic flow at crosswalks in micro-simulations using cellular automata. Tsinghua Science & Technology 12, 214–222.

Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H., 2019a. M2det: A single-shot object detector based on multi-level feature pyramid network, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 9259–9266.

Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., Wu, Y.N., 2019b. Multi-agent tensor fusion for contextual trajectory prediction, in: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 12126–12134.

Acknowledgements

I would like to give my utmost thanks to all the people who have helped me during my Ph. D. studies.

First, I would like to express my appreciation to my supervisor Prof. Monika Sester. She has given me the encouragement, inspiration, guidance, and support throughout my research. It has been an honor for me to work with her in the past four years. Furthermore, I am very grateful to the funding provider DFG (Deutsche Forschungsgemeinschaft) for GRK 1931 that made my research financially possible. I sincerely thank Prof. Christian Heipke for his comments and corrections of my thesis.

Second, I would like to thank all the professors and colleagues from the Institute of Cartography and Geoinformatics at Leibniz University Hannover and from the Research Training Group SocialCars. I had many opportunities and wonderful experiences to learn from and exchange ideas with them. I would also like to thank the people whom I have been collaborating with. Namely, they are Prof. Jörg P. Müller and Fatema T. Johora from TU-Clausthal, Prof. Bodo Rosenhahn and Wentong Liao from the Institute of Information Processing at Leibniz University Hannover, Prof. Michael Ying Yang from University of Twente, Prof. Mark Vollrath and Dr. Frederik Schewe from TU-Braunschweig, and Prof. Hiroshi Murase, Prof. Takatsugu Hirayama, Dr. Hailong Liu and the other colleagues from the Murase Lab at Nagoya University.

Last but not least, I would like to thank my family and my friends for their constant support not only for my research, but also many other aspects.

Hao Cheng

Curriculum Vitae

Personal Data

Place and Date of Birth: Hubei, China — 11 October 1987

Education

04.2017 – 03.2021 Ph.D. student
Leibniz Universität Hannover, Germany
Institute: Institute of Cartography and Geoinformatics
Project: SocialCars, cooperative (de-)centralized traffic management
Thesis Topic: "Deep Learning of User Behavior in Shared Spaces"
Advisor: Prof. Dr.-Ing. habil. Monika Sester

08.2019 – 10.2019 Special research student
Nagoya University, Japan
Institute: Graduate School of Informatics, Murase Lab
Project: "Interaction Detection between Turning Vehicles and Vulnerable Road Users at an Intersection"
Advisor: Prof. Dr. Hiroshi Murase and Prof. Dr. Takatsugu Hirayama

10.2014 – 03.2017 Master of Science in Internet Technologies and Information Systems (ITIS)
TU Braunschweig, **Leibniz Universität Hannover**, **TU Clausthal**, **Georg-August-Universität Göttingen**, Germany
Thesis: "Food Recommender Systems"
Research: "Pattern Recognition in Online Recipe Data"
Advisor: Prof. Dr. Eelco Herder

09.2007 – 06.2011 Bachelor of Administration in Information Management and Information System
Huazhong Agricultural University, China
Thesis: "Warehouse Management Information System—Based on Chongqing Xinlai Electronic Technology Co., Ltd."
Advisor: Dr. Hui Huang

Work Experience

06.2012 – 06.2014 Project Manager
China Mobile Group Hubei Co., Ltd, China

07.2011 – 06.2012 Sourcing Engineer
Yulong Computer Telecommunication Scientific Co., Ltd, China