Interaction-aware Pedestrian Trajectory Prediction using Monocular Video in Automated Driving

Master Thesis

by

S.H.J. Tak

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 16, 2022 at 13:00.

Student number:	4975154	
Thesis committee:	Prof. Dr. Dariu M. Gavrila	TU Delft, Academic supervisor
	M.Sc. Phillip Czech,	Mercedes-Benz AG, Daily supervisor
	Dr. Julian F.P. Kooij	TU Delft, Committee member
Date:	June 4, 2022	

TU Delft

Preface

This master thesis was written at Mercedes-Benz in Stuttgart, as the final part of the Master of Science Mechanical Engineering degree at Delft University of Technology. I would like to thank M.Sc. Phillip Czech, Dr. Ir. Fabian Flohr and M.Sc. Markus Braun for the many interesting discussions, as well as their supervision in general, throughout my thesis. Furthermore, I want to thank Prof. Dr. Dariu M. Gavrila for providing me with the academic supervision, more specifically, with feedback on my final draft and green light presentation.

S.H.J. Tak
Stuttgart, June 4, 2022

Abstract

Pedestrian trajectory prediction is essential for developing safe autonomous driving systems. Such trajectories depend on various contextual cues, among which surrounding objects.

This work proposes the first pedestrian trajectory prediction method in the 2D on-board domain that models interactions between the pedestrian and surrounding static- and dynamic- contextual objects using a graph-based approach. Our two-stream model separately encodes past motion history and interactions. The encoded information from both streams is fused and decoded to generate future pedestrian trajectories. The interactions are modelled using spatial graphs, which are temporally connected using a Gated Recurrent Unit. The graph nodes represent the pedestrian and contextual objects, and the edges represent the interaction importance between nodes.

In experiments on the *PIE* and $JAAD_{full}$ dataset, it is shown that our graph-based interaction-aware trajectory prediction method outperforms all considered baselines on nearly all metrics. Moreover, the performance gain on $JAAD_{full}$ is most significant for the close-by pedestrians. Finally, modeling the interactions with all considered contextual objects, i.e. vehicles, crosswalks, and traffic lights, improves trajectory prediction performance most compared to only using a subset of these objects.

Contents

1 Introduction

Vehicles are one of the most widely used types of transportation around the globe. However, this widespread use does come with safety concerns, as an estimated 1.35 million people die in vehicle-related accidents every year. More than half of these deaths involve *Vulnerable Road Users (VRUs)*, such as pedestrians and cyclists [99]. A 2008 research in the united states into car crashes indicated that in over 94% of cases, the critical reason, i.e. the last event in the crash causal chain, can be attributed to the driver [63]. Therefore, taking the human out of the equation by the development of *reliable autonomous driving systems (ADS)* has the potential to reduce this number drastically. In order to safely manoeuvre in a scene with other traffic participants, it is paramount that an ADS understands how this scene will develop over time.

1.1. The trajectory prediction problem

In order to gain insights into scene development, researchers have been focusing on *future behaviour prediction* of VRUs. These VRUs are particularly challenging traffic participants to deal with due to their ability to quickly switch between motion modes, e.g., standing still to walking [67]. The type of VRU behaviour prediction that can strongly influence safe ADS navigation is *trajectory prediction*, i.e. predicting the future path of the VRU. The trajectory prediction problem can be divided into three sub-problems when viewed on the highest level of abstraction [80], as is illustrated in Figure 1.1.

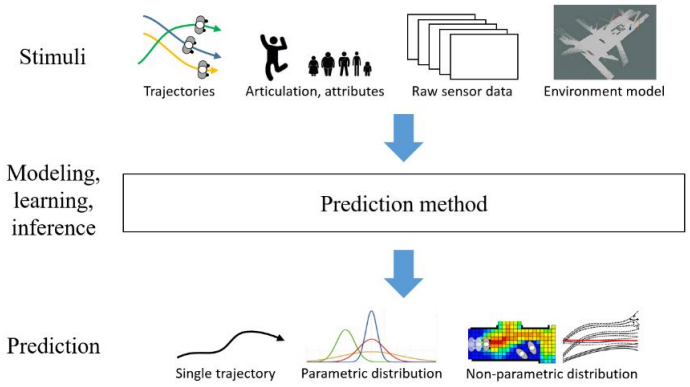


Figure 1.1: Overview of the trajectory prediction problem. Certain stimuli serve as input for a prediction method, which transforms the input into a trajectory prediction. Figure adopted from [80]

Thus, for a certain scene, the trajectory prediction problem takes certain *stimuli* as input and processes them to make predictions. The scene contains a certain number of dynamic objects, i.e. objects that have the potential to move by themselves, such as pedestrians, cyclists, and vehicles. These objects are called *agents*. The agent of interest, i.e. for which the trajectory is predicted, is denoted with the term *target agent*. The environment and objects surrounding the target agent, as well as attributes of the target agent itself, are potential *contextual cues*. These contextual cues present both internal- and external stimuli providing relevant information for the motion behaviour of the target agent.

1.1.1. Domains

Research into the field of trajectory prediction is performed in three application domains, namely service robots, intelligent vehicles, and advanced surveillance systems [80]. Methods developed for these applications are distinguished based on the type of used sensor data, as well as in which coordinate system and perspective they operate. The following distinct domains are defined as the reference for the remainder of this thesis.

- **The on-board domain**: observes the environment from a moving ground-level perspective, i.e. the sensors move along with the coordinate system of an autonomous system. Safe navigation of the autonomous system through an environment is essential for enabling the use of this technology in society. Safe navigation requires the autonomous system to anticipate the future behaviour of humans or other road users and react accordingly. The first application belonging to this domain is *self-driving vehicles*, in which case the autonomous system is a vehicle, and the environment is some drive-able infrastructure within the operational design domain. The second application that belongs to the on-board domain is *mobile service robots*, in which case the autonomous system is a service robot, and the environment is an open-ended domestic, urban, or industrial space where there is interaction with humans. A further distinction is made below, based on the type of used sensor data.
  - **The 2D on-board domain**: solely uses a monocular camera for perception. This means that the environment is perceived as projected on the 2D image plane, and thus, the system only uses data in the image, or pixel, coordinate system. Most importantly, there *is no reliable information present* in the sensor data about the actual distance in meters between the sensor and any arbitrary agent or object in the scene.
  - **The 3D on-board domain**: utilises any perception sensor where the data is provided in a three-dimensional coordinate system, such as the car/sensor or world coordinate system. Most importantly, there *is reliable information present* in the sensor data about the actual distance in meters between the sensor and any arbitrary agent or object in the scene.
- **The top-down domain**: observes the environment from a fixed, usually top-down perspective. In other words, the sensors are fixed cameras, i.e. surveillance cameras, generally used to analyse human crowds and vehicular traffic scenes. Trajectory prediction in this domain can be used to support many tasks, such as traffic monitoring and crowd management. The *advanced surveillance systems* application belongs to this domain.

1.1.2. Challenges

Predicting the future motion of other road users is a complex problem in general. However, VRUs are particularly challenging road users to deal with due to their ability to quickly switch between motion modes, e.g., standing still to walking [67]. Predicting the correct future motion is problematic because it can depend on various stimuli. Additionally, the domain-specific challenges are discussed below.

- **The on-board domain**: has two main challenges that are both present for the 2D- and 3D sub-domains. First, the intelligent vehicle must deal with *partial visibility* [52]. That means the environment can be partially occluded and only becomes visible in the next frame when the car has moved. Second, the ego-motion of an autonomous system can *influence the behaviour* of other agents in the scene [74].
  - **The 2D on-board domain**: specifically introduces two more challenges [72, 53], both related to the limitations introduced by using only a monocamera. First, the ego-motion does not only affect the motion of other agents in a scene, but it also directly influences the predicted trajectories, as these are also projected on the image plane. In other words, *disentangling* the ego-motion from the trajectory prediction is hardly possible. Second, the *relative positioning* of agents and objects in a scene is harder to predict due to the absence of reliable depth information, which leads to a more noisy environment perception.
  - **The 3D on-board domain**: enables reliable depth information from 3D sensor modalities or prior knowledge, e.g., LiDAR or Birds-Eye-View maps. This opens up the possibility to more accurately determine *relative positions* of agents and objects, reducing the noise on the environment perception. Contrary to the 2D domain, the 3D on-board domain can *disentangle* the trajectory prediction from the ego-motion, although only if a mapping to world coordinates is available.

- **The top-down domain**: generally works with stationary cameras that have a clear top-down overview of the entire scene. Therefore, it does not have to deal with any of the on-board challenges. However, due to the camera positioning, it is not able to work with more detailed *agent- or scene attributes* that can only be captured with a ground-level camera.

1.1.3. Contextual cues

Contextual cues are the internal or external stimuli that provide relevant information for the motion behaviour of agents in a scene. These contextual cues can originate from two main sources. First, from the target agent itself, i.e. *target agent context cues* and second, from anything that surrounds the target agent. Regarding the latter, we can further differentiate between static surroundings, i.e. *static context cues*, and dynamic surroundings, i.e. *dynamic context cues*. Each type of contextual cue is further elaborated on below.

- **Target agent context cues** refer to the target agent semantic attributes, articulated pose or kinematics. *Semantic attributes* include demographic information, agent appearance, and latent features such as awareness of the approaching vehicle. *Pose* includes visual features with regards to the agent's body, such as body pose, head- and body orientation, or hand gestures. *Kinematics* of the agent refers to the position and velocity of the pedestrian.

- **Static context cues** refer to *surrounding objects- or agents that remain stationary* with which the target agent can potentially interact. The static contextual objects include the infrastructural environment, such as pedestrian crossings, stoplights and street layout, as well as non-moving surrounding agents, such as parked vehicles.

- **Dynamic context cues** refer to the *surrounding moving agents* with whom the target agent can potentially interact. These other agents include, amongst others, vehicles, riders, and pedestrians. Moreover, in the on-board domain, the ego-vehicle is considered a dynamic context cue.

1.1.4. Pedestrian crossing action prediction

Pedestrian crossing prediction can be considered a reduced, simplified task compared to the trajectory predicting problem. Where pedestrian trajectory prediction is a *regression problem* predicting a future path, crossing prediction is a more simple *classification problem* where the goal is to predict whether the pedestrian is going to cross the street in the near future. The task of pedestrian crossing prediction uses similar stimuli and model architectures as trajectory prediction methods, and hence, the associated literature could provide interesting insights.

Note that even though the task of crossing prediction is often called *intention prediction* in literature, there is a distinction. Where crossing prediction is about the *action of crossing*, intention prediction refers to predicting the pedestrian's *principal goal* to cross the road, i.e. the true latent intention. For example, a pedestrian waiting at a bus stop can walk onto the road for a brief moment to check if the bus is coming. In this case, the pedestrian is performing the action of crossing, but its principal goal is not to cross the road.

Intention prediction has not been explored much [70, 89, 15], whereas pedestrian crossing prediction has received more attention [106, 50, 72, 76, 12, 60, 93, 71].

1.2. Focus of the thesis

The focus of this thesis has been decided based on two particular aspects of the trajectory prediction problem. The first aspect involves *the application domain* of choice, and the second is regarding the *complex motion behaviour of VRUs* and how to adequately model this. Both are discussed below.

1. Regarding *the application domain*: the domain of main interest is the *2D on-board domain*. The reasons for choosing the 2D on-board domain over the 3D on-board domain are threefold.

    - *Improve 2D image-based reasoning:* having strong 2D sensor processing would help in a late fusion approach more than a weak sensor. Late fusion means that a vehicle with multiple sensor modalities processes each sensing modality independently and only fuses them at the final stages using probabilistic techniques. From this perspective, it is important to explore the limitations of the sensors. Also, humans have been shown to be able to drive a motor vehicle in day-to-day cases without explicit 3D information, i.e. by driving using only one eye [4, 56]. Although the driving performance did suffer compared to humans with stereo-vision, this gap between humans and

machines is still to be solved. Especially with the recent advances in monocular depth estimation [57], the 2D predictions can be uplifted to usable 3D coordinates to accommodate a complete monocular prediction pipeline.

- *Wide applicability and low-cost*: the only sensor requirements for the 2D on-board domain are a monocular camera and optionally ego-motion sensors, such as a GPS and IMU. Operating without the requirements of the additional sensors from the 3D domain, e.g., LiDAR/RaDAR/stereo, or prior knowledge, e.g., HD maps, means 2D on-board methods are widely applicable, and the associated sensor costs are relatively low.
- *Functional-safety*: The functional-safety standard is referred to in ISO 26262, which emphasises the failure-safe behaviour of a system, such as a vehicle, in case of malfunctioning electrical systems. In case of an error, the system must switch into a safe state where the system is no longer available or reliable, but at least safe [55]. Example safe states for an SAE level 2 and an SAE level 4/5 autonomous vehicle are that the driver takes over and that the vehicle stops in a safe place, respectively. Making hardware in autonomous vehicles failure-safe is one of the challenges in autonomous vehicles and requires redundancy [43]. Redundancy in an ADS that employs late-fusion (or a combination of late- and early-fusion) can be realised by assuring that sensors can operate largely independently from one another. In other words, in case of 3D sensor failure, knowing to what extent 2D sensor modalities can (temporarily) take over is critical, be it with the sole purpose of bringing the vehicle to a safe state. Therefore, it is relevant to assess how accurate trajectory prediction methods can be whilst only using minimum sensors, i.e. only monocamera and optionally IMU/GPS.

2. Regarding *adequately modeling complex motion behaviour of VRUs*: as has been described in section 1.1.2, the future motion of VRUs can depend on a large variety of stimuli. These stimuli can also include static- and dynamic context cues. For example, when a pedestrian walks towards a pedestrian crossing, it seems likely for an observing human that the pedestrian will cross the street using this pedestrian crossing. However, a machine can only know this when the interaction between the pedestrian crossing and the pedestrian is somehow accounted for. Hence, methods *that model interactions with static- and dynamic environmental cues* are the second focus point.

1.3. Problem statement and research question

Within the overall trajectory prediction literature, the top-down and 3D on-board trajectory prediction domains have recently seen strong growth of methods that account for interactions of the target agent with the environmental context to improve performance. These methods are either *distance-based* [1, 29, 21, 13, 46], *attention-based* [83, 84], *graph-based* [44, 59, 33, 85, 48, 38, 51] or *Transformer-based* [27, 108]. The environmental context refers to either the static environment (e.g., stoplight, pedestrian crossing, signs) or the dynamic environment (i.e. other agents). Although trajectory prediction in the 2D on-board domain has been explored [6, 53, 72, 54, 70, 93, 100, 91, 102, 90, 105, 104, 66, 9, 104, 98, 69, 62], modeling interactions has not received much attention. Only recently, the first two interaction modeling methods have been proposed [54, 73], both of which indicated that integrating interactions in the 2D domain is promising. This indicates the potential to further improve the 2D trajectory prediction domain, which is interesting for the reasons mentioned in section 1.2. Moreover, the closely related 2D on-board domain that predicts pedestrian crossing action, instead of trajectories, has recently seen a graph-based interaction modeling approach [50] indicating promising results. Since this method operates in the 2D on-board domain, it serves as an inspiration for graph-based interaction modeling approaches in the respective trajectory prediction domain.

Considering the relevance of the 2D on-board domain, as well as the recent advances of graph-based interaction modeling in the 3D on-board- and top-down domains [44, 59, 33, 85, 48, 38, 51], as well as in the 2D on-board crossing action prediction domain [50], an open research direction is to investigate a similar graph-based interaction modeling method in the 2D on-board trajectory prediction domain.

The above-stated problems lead to the formulation of the following research questions:

- *Can modeling the interactions between a pedestrian and its static- and dynamic context using a graph-based approach benefit the performance of a 2D trajectory prediction framework?*
- *Which interactions with dynamic- or static context provide the most significant contribution to the 2D on-board pedestrian trajectory predictions?*

1.4. Outline

First, chapter 2 discusses the previous work related to the topic of this thesis. Next, chapter 3 elaborates on the method and the contributions of this work. Third, chapter 4 explains the experimental setup, as well as the results and corresponding discussion points. Finally, chapter 5 concludes the thesis with an answer to the research questions, method limitations and future work.

2

Related Work

This chapter first discusses the relevant deep learning fundamentals used throughout this thesis. Next, the existing work in the field of pedestrian trajectory prediction is described. After that, the focus shifts towards those methods that model interactions between the target agent and its environmental context. Finally, the datasets and metrics used for evaluation are discussed.

2.1. Fundamentals

The section aims to build a strong foundation for understanding the remainder of this thesis.

2.1.1. Deep learning methods

This section will discuss the basics of deep learning methods often encountered in this thesis. The field surrounding each method is much broader than explained below and the focus is only on the parts of each method that will be used later on in this thesis.

2.1.1.1 Recurrent Neural Networks

A *Recurrent Neural Network*, or RNN for short [81], is a type of deep learning network with the purpose of processing sequential data, for example machine translation, stock-market prediction or trajectory prediction. The latter example indicates its relevance for this thesis. The networks are called *recurrent* because each element in a sequence is processed using identical RNN cells, i.e. cells with identical weight- and bias values. Thus, the RNN cell is used recurrently for all elements in the sequence. A key component of the RNN cell is that it does not only take the current element of a sequence as input, but also the output of the cell that processed the previous element. As such, RNNs are able to capture sequential patterns based on current and previous inputs.

A downside of regular RNNs is that they suffer from *vanishing gradients*, which is problematic for longer sequences. This means that the gradients, which are used for updating weights and thus for learning, become smaller and smaller when performing backpropagation through time through a long sequence. Eventually, they become so small that the weights of these far-away sequence elements can barely be updated, thus impairing the model to learn from these far-away elements, i.e. early inputs. A solution to this problem was posed in the form of a more elaborate RNN cell, namely the *Long-Short Term Memory* [32], or LSTM, cell. LSTMs extend the classical RNN cell by adding a *cell-state* and three *gates*. The cell-state serves to retain long- and short-term memory and the gates regulate what information is kept, added or forgotten from the cell-state, using the previous hidden-state and current input.

A less memory intensive alternative to the LSTM is the more recently introduced *Gated Recurrent Unit* [16], or GRU. Two key changes compared to the LSTM are firstly, the GRU gets rid of the cell-state, instead using the hidden-state to memorize relevant information and secondly, the GRU only uses two gates to regulate what information is kept in the hidden-state.

2.1.1.2 Convolutional Neural Networks

A *Convolutional Neural Network* [45], or CNN for short, is a type of deep learning network with the purpose of processing data with a known *grid-like topology* [20]. For example, this can refer to images, where the data represents a two-dimensional pixel grid, or to time-series, where the data represents a one-dimensional grid with timesteps as grid-points. Both of these applications are relevant for this thesis, as spatial processing of image data can be used to capture a large variety of contextual cues and processing time-series data can be used for past- and future trajectory data.

The key to CNNs is the convolution operation, which is a method to process the input data with grid-like topology to produce a feature map. This is done by sliding, or *convolving*, a matrix containing learn-able weights of a certain size, i.e. the *kernel*, across the input and performing matrix multiplications between the kernel and every subsection of the input grid it is slid across. The output is a *feature map*, which is a representation of certain patterns in the data.

2.1.1.3 Graph Neural Networks

A *Graph Neural Network*, or GNN for short, applies neural networks to data that is presented as *graphs*. Examples include node classification, link prediction and clustering [110]. Recently, GNNs have been used to model spatial and temporal interactions between objects and/or agents in the field of trajectory prediction, which why these type of neural networks are of interest to this thesis.

A graph is denoted as $\mathbf{G} = (V, E)$, where $|V| = N$ is the number of nodes and $|E| = N^e$ the number of edges. The adjacency matrix $\mathbf{A} \in R^{NxN}$ contains which nodes are connected. Each node contains some representation for an entity of some type. The *edges* represent the type of connection between each pair of entities. The nodes can send- and receive *messages* respectively to- and from neighbouring nodes over these connections. The messages allow the nodes to learn about neighbouring nodes and the type of connection that exists between them. Doing so, the graph representation is updated and can be used for further processing. GNN variants that are of specific interest to this thesis are the *Graph Convolution Network [39]*, i.e. GCN, and the *Graph Attention Network*, i.e. GAT.

GCNs generalize the convolution operation from CNNs to GNNs. A GCN layers works in three steps. First, each node collects the embeddings, or *messages*, of all connected neighbouring nodes. Second, these messages are *aggregated* using some aggregation function. Last, the aggregated messages are fed through a neural network whose output represents the *updated node embedding*, which are in turn used for further processing.

Graph Attention Neural Networks [96] apply the concept of *self-attention* [95], which is a method to extract global dependencies between inputs and outputs, to graph neural networks. This enables each node to specify a different weight values to each neighbouring node. This allows the model to learn which neighbouring nodes are most important for the state update. Also, the *multi-headed attention*, which performs the self-attention operation multiple times in parallel, is used in *GAT* to increase the expressive capabilities of the architecture.

2.2. Trajectory Prediction

As described in section 1.1, the trajectory prediction problem takes certain *stimuli* as input and processes them using some *prediction method* to generate a *prediction output*. First, a broad overview of trajectory prediction literature is provided based on the prediction method taxonomy defined by [80]. Subsequently, the trajectory prediction literature of interest is distilled from this overview. This literature is analysed in great detail to evaluate the stimuli, prediction methods and prediction outputs in both the 2D on-board domain, as well as the top-down and 3D on-board domains.

2.2.1. Trajectory prediction method taxonomy

In trajectory prediction literature a distinction is made between three types of prediction methods, namely physics-, planning- and pattern-based . First, *physics-based* methods use explicit dynamical models based on Newton's laws of motion, where a distinction is made between methods that define the kinematics using only a single dynamical model [31, 58, 103] and method that use multiple dynamical models [42, 41, 88]. Second, *planning-based* models aim to reach a particular goal via a route which minimizes an objective, or cost, function. The cost function is defined differently depending on whether the method is a forward-planning method [94, 10, 5], which means the cost function is predefined by the user, or an inverse-planning method

[111, 40, 77], where the cost function is crafted in an online fashion based on observations of human behavior. Lastly, *pattern-based* approaches learn motion dynamics from data patterns by approximating functions that fit these patterns. Depending on the type of approximation, these approaches can be divided into sequential methods [34, 1, 3], which assume that the current state has some conditional dependence on previous states, and non-sequential methods [24, 107, 22], which aim to model the distribution over the entire trajectory without any conditional dependencies between states. The pattern-based approaches have gained tremendous traction in recent years, partly due to the increasing amount of readily available large scale datasets. This thesis will revolve around prediction methods that use neural networks to fit data patterns and thus, look at *pattern-based methods*.

2.2.2. Stimuli

The *stimuli* are the internal and external influences that affect the target agent's future behavior. The various stimuli that have been researched in trajectory prediction literature are discussed below.

2.2.2.1 2D on-board domain

The *motion history* of the target agent represents an important cue for its likely future behavior. This becomes evident from literature, as all reviewed trajectory prediction papers rely on information that captures some sort of motion history of the target agent. This information includes *bounding boxes* [6, 53, 72, 54, 70, 93, 100, 91, 102, 90, 105, 104, 66, 92], *bounding box velocity* [9, 90, 92] or *implicit motion history* relative to contextual objects [50]. Note that the 2D on-board domain uses bounding boxes instead of only a pixel location, such as the bounding box center, as the bounding box size provides additional information on *agent size*. The size information enables implicit learning of the depth dimension to some extent, e.g. a small pedestrian bounding box is likely to be further away than a large one. Note that this requires certain *restrictive assumptions* on, for example, agent size, introducing an issue for agents of non-average size, such as children.

Section 1.1.2 mentioned that the observations and predicted trajectories are both relative to the motion of the ego-vehicle. To account for this, the majority of methods incorporate *ego-motion* information *explicitly* in their trajectory predictions, either by using an encoding of the *past ego-motion* [54, 93], *the predicted or ground truth future ego-motion* [53] or *both* [6, 72, 70]. The methods that do not take into account ego-motion at all [50, 100] likely do so due to the absence of ego-motion data in the dataset. Ego-motion can also be captured *implicitly* by inferring the camera motion from *dense optical flow maps* over the images [90, 91, 102] or, arguably, by feeding the model *image features*, like the pedestrian image patch [50] or environmental semantic segmentation [92].

Optical flow is not only used for inferring ego-motion, but also in conjunction with *Region-of-Interest pooling* [26], i.e. ROI Pooling. In this case, the dense optical flow map is evaluated only in the region of interest, that is, the target agent bounding box [53, 105]. The optical flow captures agent motion and appearance changes, both of which are considered relevant cues.

Several methods prove it is beneficial to take the *appearance* of the target agent into account [100, 93, 50], by encoding the target agents' bounding box using a CNN. This allows the model to *implicitly* learn *agent attributes* that can help with the prediction, e.g. gaze direction or body orientation. Information on agent attributes can also be provided *explicitly* to the model to improve performance. For example, [93] used body pose, [92] used body orientation and [54] integrated (predicted) agent actions, e.g. getting into a car, as a cue.

2.2.2.2 Top-down and 3D on-board domain

Similar to the 2D on-board domain, all reviewed trajectory prediction papers in the top-down and 3D on-board domain rely on information that captures some sort of *motion history* of the target agent. However, contrary to the 2D on-board domain the motion history is captured using *spatial coordinates* in the world coordinate system [46, 78, 21, 84, 83, 108, 1, 38, 48] or *dynamical states* [13, 85], which also include the coordinates. Next to the motion history, Ridel, Deo, Wolf, and Trivedi [78] show that incorporating *pedestrian head orientation* as a prior improves the trajectory prediction performance. Their method encodes past ego-motion, past trajectory and additionally the past head orientations of the pedestrian using LSTMs. This information is concatenated and decoded to obtain future target agent trajectory.

Also the 3D on-board methods have to deal with the challenges associated with the moving ego-vehicle. Therefore, these methods incorporate *ego-motion* information *explicitly* via an encoding of the *past ego-motion* [78, 106] or of the *predicted- or ground truth future ego-motion* [85]. On the other hand, [13] do *not explicitly* take ego-motion into account.

Some methods integrate stimuli from the *static context* which allows the model the learn interactions between the target agent and static objects. Such stimuli include a *top-down scene image* [46, 84, 83] or a *top-down binary obstacle- or semantic map* [85].

2.2.3. Prediction method

The *prediction method* represents the core of the trajectory prediction problem. It takes the stimuli as input and processes them in a highly non-linear manner using some neural network-based method. All prediction methods contain a *temporal sequence modeling* component, which involves processing the temporal observations of the agents' trajectory and use those to make a future prediction. Additionally, a subset of methods contain an *interaction modeling* component, which involves reasoning over interactions between the target agent and contextual cues. This section will discuss the temporal sequence modeling for all domains, whereas the interaction modeling will be broadly discussed in section 2.3

2.2.3.1 2D on-board domain

A large body of 2D on-board trajectory prediction literature heavily relies on *Recurrent Neural Network-based* architectures for predicting the future temporal sequence [6, 53, 54, 91, 105, 105, 70, 100, 72, 90, 66, 104, 9, 98]. In the field of RNNs for example, Bhattacharyya, Fritz, and Schiele [6] implement a *two-stream LSTM encoder-decoder* architecture. The first stream is responsible for future ego-motion prediction, i.e. the odometry prediction, which is taken care of by first encoding the past ego-motion using an LSTM encoder and concatenating the final hidden state with a representation of the most recent video frame. This representation is obtained using a CNN and allow the model to implicitly learn about road features that tell something about possible future ego-motion. A prediction of the future ego-motion is obtained by decoding this concatenated information using an LSTM-decoder. The second LSTM encoder-decoder stream is responsible for future bounding box predictions. The predictions are conditioned on the past ego-motion, past bounding boxes of the target agent, as well as the predicted future ego-motion. The model is illustrated in figure 2.1.

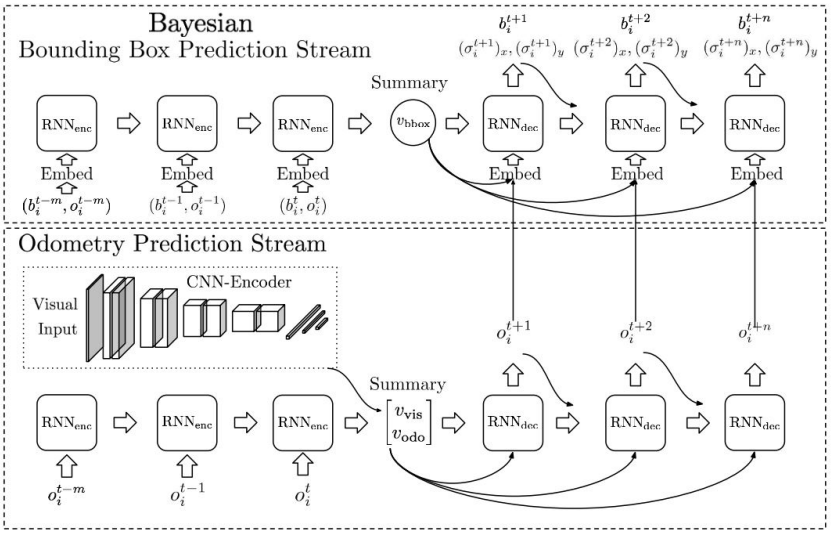


Figure 2.1: Two-stream trajectory prediction framework from [6]. The bottom stream predicts future ego-motion, conditioned on past ego-motion and a raw scene image. The top stream predicts future bounding boxes, conditioned on the past bounding boxes, ego-motion and future ego-motion

Many follow-up methods have taken inspiration from [6] and are therefore similar in design [53, 70, 105]. For example, Rasouli, Kotseruba, Kunic, and Tsotsos [70] add a *third stream* for the prediction of the pedestrians' *crossing intention*. This intention prediction stream uses an LSTM encoder-decoder with the bounding box and corresponding local context as input. The binary intention predictions serve as input in the trajectory decoder together with the representation of the past motion history. To weigh the importance of both representations *self-attention units* are added. Additionally, a *temporal attention module* is added to the bounding box prediction stream encoder to learn the important information of the input sequence. Conditioning the future bounding box predictions on the predicted latent intentions was shown to improve model

performance across all evaluated metrics. [105] and [53] condition the bounding box predictions additionally on *optical flow*, which allows the model to reason over the motion of the agent relative to the environment. Moreover, they swap out the LSTMs for more efficient *GRUs*. Styles, Guha, and Sanchez [90] follow up on [105] and state that using pose as a cue can be unreliable, as pose predictions tend to be difficult in low resolution and low light settings. Hence, they propose *STED* which uses a two stream architecture that encodes optical flow information using a CNN and past bounding box and velocity information using GRUs. The outputs are concatenated and decoded using another GRU to obtain future bounding box locations.

Opportunities to improve trajectory prediction performance have also been chased by approaching it from a *multi-task perspective*, i.e. predicting jointly with another task such as *crossing prediction* [72, 9, 92] or *end-point prediction* [72, 104, 98]. For example, Rasouli, Rohani, and Luo [72] first encode the past bounding boxes and past image-plane grid locations of the target agent, as well as ego-vehicle motion using LSTM encoders. The interactions are modeled using a *Categorical Interaction Module*, elaborated in chapter 2.3, to obtain the relevant interactions. The temporal encoding and decoding use the same concept - one module encodes or decodes using a single LSTM with shared weights, respectively *Joint en- & de-coding*, and one module encodes or decodes using three separate LSTMs, i.e. *Independent en- & de-coding*. The 'joint' and 'independent' information of the encoding and decoding modules is combined to obtain respectively the final encoded representation and the multi-task prediction output. Similar to [72], other endpoint-based methods [104, 98] train the end-point and trajectory prediction pipeline jointly. however, the latter two condition the trajectory predictions explicitly on the predicted endpoints. For example, Yao, Atkins, Johnson-Roberson, Vasudevan, and Du [104] leverage a custom bi-directional decoder that first predicts the end-point of the trajectory, after which the trajectory is reconstructed in a backwards fashion from the end-point to the current location. Wang, Wang, Xu, and Crandall [98] do not only predict a single end-point for a trajectory, but instead predict step-wise endpoints for each frame in the prediction horizon. An attention mechanism is used to reason over which step-wise endpoints are most important to the future trajectory, after which this is used as input in the trajectory decoder to make future trajectory predictions.

Next to RNNs, there is a small portion of papers that rely on *Convolutional Neural Networks*, either *entirely* [102] or *combined* with physics-based constant velocity model [91]. Yagi, Mangalam, Yonetani, and Sato [102] use agent pose, location, scale and camera ego-motion, inferred from optical flow, to make pedestrian trajectory predictions from a first-person view. The proposed method utilizes a three-stream architecture, where each stream encodes temporal information of each cue using 1D convolution layers. The outputs of each stream are concatenated and decoded using de-convolutions to obtain future location and scale predictions. Styles, Ross, and Sanchez [91] use the constant velocity model to make an initial trajectory prediction. The initial prediction is then corrected using optical flow map of the target pedestrian, which is processed with a CNN. The paper also evaluates the model when pre-training on *machine-annotated data*, which is shown to improve performance. Contrary to what one may think, using such a relatively simple method to make an initial prediction does not necessarily mean the performance will suffer [87].

2.2.3.2 Top-down and 3D on-board domain

Just as with the 2D on-board domain, the prediction method will be discussed in terms of *sequence modeling* and *interaction modeling*. Again, the latter is much more extensively elaborated in chapter 2.3.

temporal sequence modeling *Recurrent Neural Network-based* models have been used extensively in the top-down domain for prediction of temporal sequences [1, 29, 21, 84, 83, 59, 108, 33, 44, 38, 48, 13]. For example, [1] is an early work that that introduced an *LSTM encoder-decoder structure* with interaction modeling to jointly predict individual-aware trajectories of all agents in a scene. This paper has been an inspiration to many follow-up works [21, 33, 29, 84]. Deo and Trivedi [21] apply a similar concept to a top-down vehicular traffic situation, but where the observing camera moves with the the target agents coordinate system. To account for this, a representation of the vehicles ego-motion is also implemented. Huang, Bi, Li, Mao, and Wang [33] change the decoder LSTM to generate multiple trajectories for each agent per time-step using a *generator* based on *Generative Adversarial Networks*. Diversity in the trajectories is obtained by training with the variety loss function from [29]. This architecture is illustrated in figure 2.2.

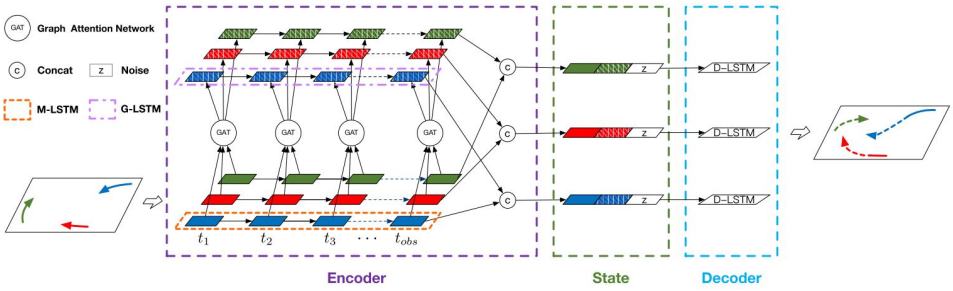


Figure 2.2: Trajectory prediction framework from [33]. First, the encoder (left) encodes the motion history of each agents using an LSTM encoder, i.e. *M-LSTM*. Next, the encoder performs interaction modeling using GAT modules and an additional LSTM encoder, i.e. *G-LSTM*, which is further elaborated in section 2.3.2. The final hidden states of both the M- and G-LSTM are concatenated with some randomly sampled noise into a state (middle). This state is decoded (right) using decoder LSTM cells to generate future trajectories.

Other works rely on *Convolutional Neural Networks* for temporal sequence prediction [59, 97]. Compared to RNNs, the use of CNNs improves computational efficiency because it enables parallel processing. For example, Mohamed, Qian, Elhoseiny, and Claudel [59] construct a *spatio-temporal graph* where all agents are connected in the spatial and temporal dimension. The graph is updated by feeding it through a *spatio-temporal graph CNN*, which is an extension of the known spatial graph convolution neural network. The resulting graph node embeddings are extrapolated by a temporal CNN to generate the future trajectory predictions.

Only recently people have started using *Transformers-based* architecture in temporal sequence prediction [108, 27, 49]. For example, [108] propose a network that encodes spatial and temporal correlations between pedestrians by using *Transformer-based encoders* in the temporal and spatial dimension. The resulting representation is fed through a fully connected layer to make future coordinate predictions one time-step into the future. The prediction is added to the motion history of the agent. These steps are performed in an *auto-regressive* fashion until the desired prediction horizon is obtained. Ablation studies indicated that temporal- and spatial modeling capabilities of Transformers are superior to several counterparts, such as *LSTMs* for temporal modeling and *GCN* and *GAT* for spatial modeling. An interesting notion is that the benefit of the Transformer-based spatial modeling over alternatives diminished for less crowded scenarios.

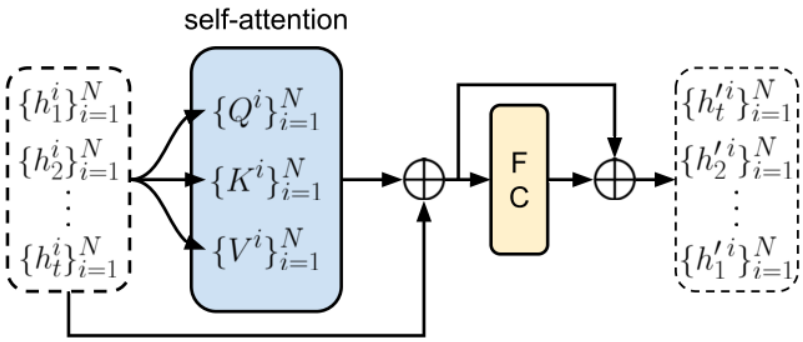


Figure 2.3: The Transformer-based temporal encoder from [108]. The encoder takes past trajectory embeddings h_t^i of a pedestrian as input. Consequently, it learns the query Q, key K and value V tensors before utilizing the multi-headed attention mechanism to learn meaningful temporal dependencies. Next, two skip connections and a fully connected layer are used to obtain the updated trajectory embeddings.

2.2.4. Prediction output

The *prediction output* is the third and final component in the trajectory prediction problem and discusses how the actual future trajectory predictions are represented.

2.2.4.1 2D on-board domain

The prediction output is discussed in three components. First, in terms of *output modality*. Second, in terms of *trajectory output representation*, which refers to how the agents' future location is represented and third,

in terms of *uncertainty*, which discusses to what extend methods capture the associated uncertainty in their predictions.

Modality Often, methods make a single future trajectory prediction, i.e. they are *unimodal* [72, 70, 100, 90, 91, 102, 105]. However, in real life an agent would have *multiple future modes*. Methods that cover this more representative and complex output are called *multi-modal*. The multi-modality is obtained either by predicting a *Gaussian distribution* from which samples are taken [6, 53, 54] or by using a *Conditional Variational Auto-encoders* where multiple random samples are taken from the latent space [66, 104, 98]. Although [6, 53] claim to predict multi-modal distributions by sampling from a Gaussian distribution, they suffer from *mode collapse*. This means that in actuality they only predict one mode with high variance [52], thus not capturing the different likely future trajectories associated with different motion modes.

Trajectory output representations The trajectory output for all the reviewed trajectory prediction papers is represented by four *bounding box coordinates*. This is essential because a real-life trajectory is *inherently 3-dimensional*, but only the 2D image plane is available. Hence, the size of the bounding box contains *implicit* information about depth, e.g. a small pedestrian bounding box is likely to be further away then a large one.

Uncertainty Knowledge of the uncertainty associated with the future predicted trajectory is of great importance. However, the majority of methods does not account for this. Their output is *deterministic* and is obtained by training using some *distance error function*, such as *L1-loss* [90], *L2-loss* [91, 72, 98] or *MSE loss* [102, 100, 70]. Methods that do account for uncertainty obtain this by training the model using a *probabilistic loss function*, such as *negative log-likelihood* [6, 53, 54, 13, 104] where the output of the model at each timestep is a *Gaussian distribution*. Most of these methods only capture the uncertainty associated with the data, i.e. *aleatoric uncertainty*. In this case, the uncertainty associated with the model parameters, i.e. *epistemic uncertainty*, is not taken into account. However, epistemic uncertainty is especially important in safety-critical applications such as autonomous driving [35] as it enables the model to learn what it knows and what it does not know. Therefore, there are works that aim to capture both aleatoric and epistemic uncertainty in the deep learning model using a *Bayesian formulation* [6, 53]. For example, Bhattacharyya, Fritz, and Schiele [6] implement a Bayesian bounding-box prediction stream to obtain an estimate of the full predictive uncertainty. In practice, this is done by combining the T outputs obtained by running inference T times. For each inference run the models' parameters are re-sampled from a certain distribution. Malla and Choi [53] additionally define the ego-motion prediction stream using a Bayesian formulation to account for the full predictive uncertainty in the predicted ego-motion. The future bounding box predictions are therefore conditioned on the uncertainty in the future ego-motion to improve prediction accuracy.

2.2.4.2 Top-down and 3D on-board domain

Just as with the 2D on-board domain, the prediction method will be discussed in terms of *output modality*, *trajectory output representations* and *uncertainty*.

Output modality Because the future trajectory of an agent has multiple modes, there are many methods that account for this multi-modality in their predictions. Different approaches have been considered, such as utilizing a *Conditional Variational Autoencoder* [85, 46, 98, 66, 104]. For example, Salzmann, Ivanovic, Chakravarty, and Pavone [85] use a *CVAE* to predict a discrete Categorical latent variable, where each category in this discrete distribution represents a different mode of behavior. Another approach is the implementation of a *GAN* in the network to generate multiple predictions per timestep [29, 44, 83, 2]. For example, Gupta, Johnson, Fei-Fei, Savarese, and Alahi [29] propose a method where the generator builds a representation of past agent trajectories which are individual-aware by combining an LSTM-encoder with a *social-pooling layer*, inspired by [1]. Next, the *generator* predicts multiple trajectories per time step conditioned on this representation and some added noise. The *discriminator* is used to pick out the socially not-acceptable trajectories from the socially-acceptable ones. Using a novel variety loss the generation of diverse multi-modal predictions is encouraged. This work is extended by Sadeghian, Kosaraju, Sadeghian, Hirose, Rezatofighi, and Savarese [83], in which the pair-wise interactions vary in strength via the use of a *social attention unit*. Moreover, they additionally model the agent interactions with the static environment via a so-called *Physical Attention Module*. Even though both of these works claim to make multi-modal predictions using a GAN, [44] state that they rather predict a single mode with a high variance. Consequently, they propose Social-BiGAT

which captures a trajectories' true multi-modality by encouraging the model to develop a bijection between the outputted trajectories and the latent space inputted to the generator. Another approach to modeling multi-modal output is presented by [21], where a single Gaussian distribution is predicted for the future trajectory conditioned on 6 different and predefined agent manoeuvres. These manoeuvres include going straight, left, right combined with either simultaneously braking or continue at the same speed.

Trajectory output representation In the 3D domain and top-down domain the future trajectory prediction is generally a sequence of *directly predicted coordinates* [1, 29, 21, 84, 83, 59, 108, 33, 44, 38, 48, 13, 78, 51]. There is also a work that predict the future *coordinates indirectly* [85] by predicting control inputs which are used to update the state space model of the target agent. Modeling each agent as a dynamical system and updating their state with control inputs, such as in [85], means the *dynamical constraints* corresponding to this agent are explicitly accounted for.

Uncertainty Just as in the 2D on-board domain, knowledge of the uncertainty associated with the future predicted trajectory is of great importance. There are methods that produce a *deterministic output* by training using some *distance error function*, such as *L2-loss* [84, 108, 83, 44], *variety loss* [33, 29] or *Mean-Squared Error* loss [78]. Other methods make a prediction that inherently contains an uncertainty measure by training the model using a *probabilistic loss function*, such as *negative log-likelihood* [13, 51, 21, 59, 1]. In these cases, the output of the model at each timestep is a Gaussian distribution.

2.3. Interaction modeling in trajectory prediction

This sections zooms in on the *interaction modeling methods* of the previously discussed trajectory prediction literature. The objective is to categorize the interaction modeling techniques for each domain. First, interaction modeling methods for the 2D on-board domain are discussed, after which the focus is shifted to the top-down and 3D on-board domains.

2.3.1. 2D on-board domain

It is *challenging* to model interactions between the target agent and its static and/or dynamic context in the 2D on-board domain, as discussed in section 1.1.2. The majority of trajectory prediction methods *do not account* for any form of contextual information [6, 53, 100, 105, 91, 90, 102, 104, 66, 98]. Attempts have been made to integrate interactions, for example by looking at the *local context* surrounding the target agent [70, 93], thereby *implicitly* accounting for either static or dynamic context cues. This is realised by encoding a bounding box *patch* exceeding the size of the agent using a CNN. Only very recently, papers in the 2D on-board trajectory prediction [54, 72, 92] and crossing- or intention prediction [50, 15, 92] domain have started to integrate interactions with dynamic- or static context features. Such interactions are captured using models based on a variety of concepts, namely *Recurrent Neural Networks* [54], *attention* [72] and *graphs* [50, 15]. Each concept is discussed below.

2.3.1.1 Recurrent Neural Network-based

Malla, Dariush, and Choi [54] reason over interactions using a so-called *interaction module*, which takes the positions and corresponding actions of each agent in the scene as input and outputs a vector representing the interactions between the target agent and its surrounding context. The core novelty of the interaction encoder lies in the *integration of both position and actions*; the goal is to learn *pair-wise interactions* between an agent and a contextual object in case the agents' action involves a contextual object. For example, a pedestrian action of *get in a vehicle* enables the model to build a relation between this pedestrian and the other relevant agent, in this case a vehicle. This pair-wise interaction should be strong if the two agents are positioned nearby. The interaction encoder is illustrated in figure 2.4.

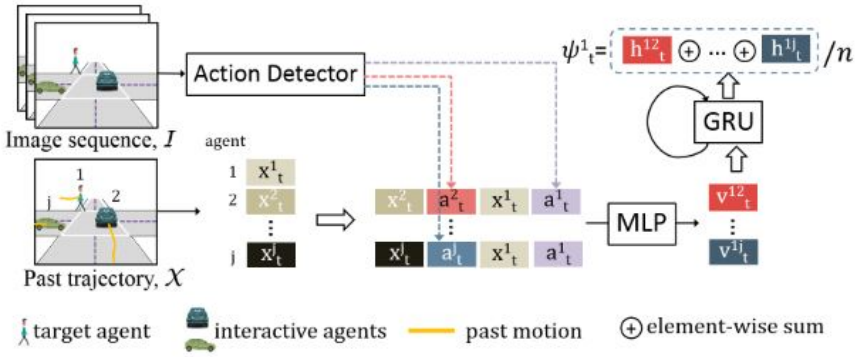


Figure 2.4: Interaction encoder from [54] using action- and position information jointly to reason over pair-wise interactions between the target- and surrounding agents.

First, the action, a, and position, x, information is extracted for all agents in a scene with one target agent and $(j-1)$ surrounding agents. This information is put in a matrix with $(j-1)$ rows, where each row contains both the actions & positions of the target agent, as well as of one surrounding agent. This matrix is embedded using an MLP into a vector of length $(j-1)$ where each element represents the pair-wise interactions between the target agent and one surrounding agent. This interaction vector is fed through a *spatial GRU* to obtain one context feature vector representing the most important interactions with respect to the target agent. Ablation studies indicate a strong benefit of using such action-based interactions by comparing.

2.3.1.2 Graph-based

The method by Liu, Adeli, Cao, Lee, Shenoi, Gaidon, et al. [50] proposes to model the spatio-temporal interactions between the target agent and its static- and dynamic context by using *spatial graphs* connected temporally using GRUs. The novelty of this method lies in the graph construction and the instance segmentation-based input. It is illustrated in figure 2.5.

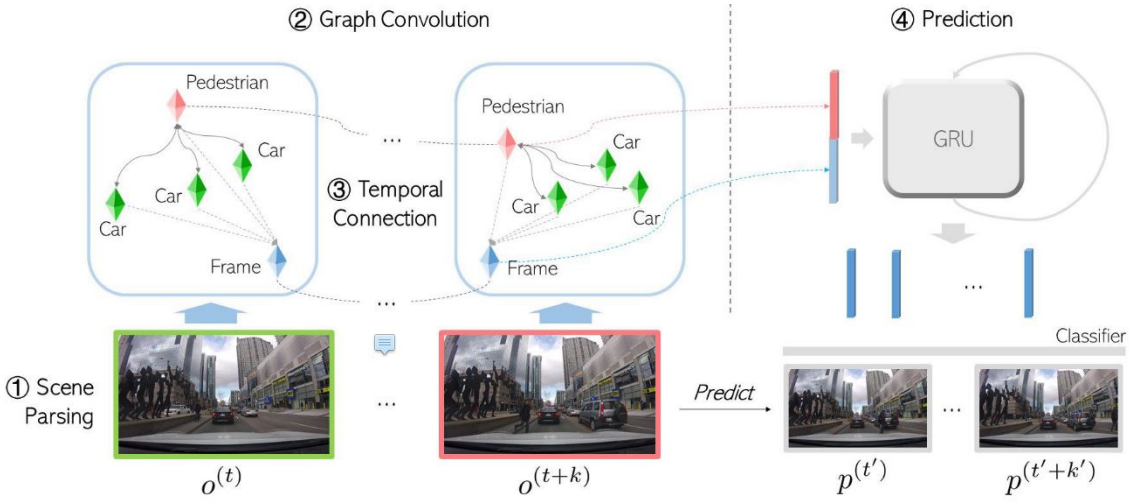


Figure 2.5: Interaction modeling method from [50]. First, the scene is parsed into static and dynamic objects. These are used to construct a spatial-graph centered at the target pedestrian, which is processed using graph convolution. The temporal dependencies are then captured via a GRUs connecting the pedestrian and context-node.

The scene is first parsed to obtain the target agent bounding box, as well as *binary masks* for all relevant objects in a scene using instance segmentation. The masks and bounding boxes are used to construct a sparse *star-graph* centered around the target pedestrian, i.e. the pedestrian node is connected to all object nodes. Additionally, a context node is connected to the pedestrian that contains an aggregation of all contextual visual information. The edge weights of the graph correspond to the *interaction importance* between the pedestrian and an object and are learnable based on two components. First, pedestrian appearance, which

can implicitly provide information on pedestrian intent, e.g. from head orientation. and second, the spatial relationship between the target agent & the object. Inspired by [109], the spatial relationship between the target agent and each contextual object is inferred by cropping out a *union binary bounding box* covering the target agent and the object, thereby preserving the relative spatial information. These union binary bounding boxes are encoded and used in the edge weight calculation, as well as in the object node representations. To account for temporal interactions between the spatial graphs at each timestep a *GRU* connects the target agent between each frame. Moreover, a *second GRU* is connected to the context nodes between each frame, although this is shown to not always be beneficial due to redundancy. By connecting the model sparsely in the the spatial and temporal domains the computational complexity is reduced. To improve the use of learnable edge weights the authors tried to add pose and object class information in the embedding, but this was was shown to decrease performance.

2.3.1.3 Attention-based

The interactions with the static- and dynamic context in the method by Rasouli, Rohani, and Luo [72] are modeled via the *Categorical Interaction Module (CIM)*, which is illustrated in figure 2.6. Similar to [50], CIM bases its modeling on semantic parsing to generate semantic maps of the scene, each of which contain objects of one semantic class in one time step. Where [50] uses these semantic maps to encode the relative distance between objects & target pedestrian and consequently employs this as a proxy for edge weighting to reason over a spatio-temporal graph, CIM takes a different approach. The *semantic maps* corresponding to a semantic class for each time step are *spatially encoded* using a *CNN* and these embeddings are *temporally encoded* using an *LSTM*. The final hidden states of these LSTMs are concatenated to obtain a *shared categorical representation*, i.e. a concatenation of all temporal embeddings for each semantic category. Finally, an *Interaction Attention Unit (IAU)* is used to determine relative importance of each category in the shared categorical representation. IAU first calculates attention weights based on the similarity between the last time-step and every other time step. Next, the scores per time-step are obtained via a softmax and used to calculate the relative importance of each category in the shared categorical representation vector for each time-step, called the context vector. The context vector and the last time-step representation are used to calculate the final interaction context.

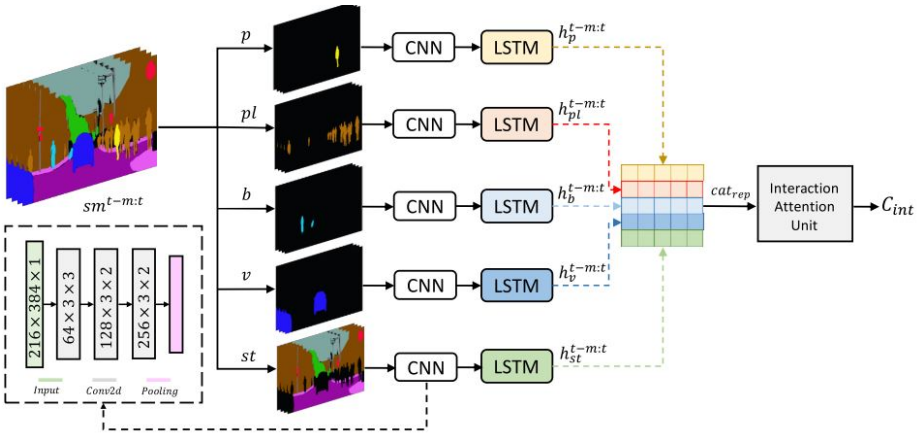


Figure 2.6: Categorical Interaction module from [72]. First, the scene is semantically parsed into different object categories. Per category, each sequence of semantic maps is processed using a CNN and LSTM. The outputs are concatenated and processed using the Interaction Attention Unit to obtain a context vector.

Ablation studies show several interesting components of the CMI. First, *categorizing* the semantic maps based on appearance and motion pattern improves results, irrespective of the used processing method. Second, the processing method used in CMI excluding the IAU already outperforms the alternative interaction processing methods, which include 3D convolution and 2D convolutions with averaging in the temporal dimension. Adding the IAU further improves the results.

2.3.2. Top-down and 3D on-board domain

Interaction modeling has been more regularly investigated in the top-down and 3D on-board trajectory prediction domains, compared to the 2D domain. From the reviewed literature, only [78] *does not account* for

contextual objects. There is a large chunk of literature that focuses on modeling interactions with the *dynamic context*, whereas interactions with the *static context* has seen less exploration [46, 85, 84, 44]. For the dynamic context cases, a distinction is made for interactions between *homogeneous agents* [1, 29, 21, 84, 83, 59, 108, 33, 44, 38] and between *heterogeneous agents* [85, 48, 13, 51, 106]. The latter poses an additional challenge as the agents have different shapes, sizes, maneuverability and behavior, thereby leading to more complex agent-agent interactions [14]. In order to deal with this, methods generally use some kind of different embedding for each of the semantic classes. For example, Li, Yang, Tomizuka, and Choi [48] accounts for the heterogeneous agents by calculating the agents' node embedding by feeding the agents coordinates through a semantic *class specific embedding function*. Yau, Malekmohammadi, Rasouli, Lakner, Rohani, and Luo [106] explicitly include object size, velocity and class in the embedding. Chandra, Bhattacharya, Bera, and Manocha [13] approach the problem by modeling each agent based on on their their differences on size, velocity and additionally shape and driving behavior.

The interactions can be captured using a variety of algorithms. These algorithms are either *distance-based* [1, 29, 21, 13], *attention-based* [83, 85, 84], *graph-based* [44, 59, 33, 85, 48, 38] or *Transformer-based* [27, 108]. Each of which will be discussed below.

2.3.2.1 Distance-based

Alahi, Goel, Ramanathan, Robicquet, Fei-Fei, and Savarese [1] models interaction between *homogeneous agents* by embedding the motion history of n agents using n LSTMs and allowing the LSTM hidden states of neighboring pedestrians to be shared via a *social pooling layer*. This pooling layer informs the target agent about the latent representation of the surrounding agents, i.e. agents within a certain *distance* from the target agent. The social pooling layer takes the form of a 'social' occupancy grid centered around the target agent, where each grid cell contains a summation of LSTM hidden states corresponding to surrounding agents located in that grid cell. This means the 'social' occupancy grid also preserves spatial information to some extent. The social occupancy grid and coordinates are both embedded via fully connected layers and used as input for the next LSTM cell. There are two main downsides to this interaction modeling method. First, it only takes into account *local information* and second, by feeding the social occupancy grid through a fully connected layer the *spatial structure* of the social occupancy grid is *nullified*. This means that two adjacent cells will be treated the same as two distant cells, which can cause generalization problems.

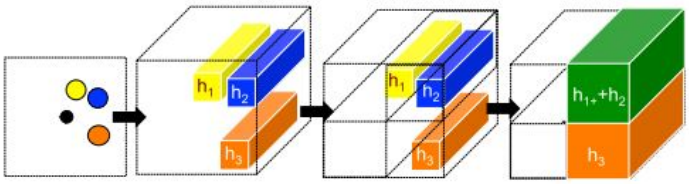


Figure 2.7: The social pooling operation from [1] uses a occupancy grid centered around the target agent to share spatial information and latent representations of surrounding agents

The social-module was improved by Gupta, Johnson, Fei-Fei, Savarese, and Alahi [29], who proposed to get rid of the grid-based pooling scheme at every time step and replace it with a *Multi-Layer Perceptron (MLP)* and *max pooling* operation at the end of the encoding process once. Consequently, a significant (16x) gain in time-efficiency is obtained. Moreover, because far-away agents can also have an influence on motion behaviour of the target agent, the pooling method is adopted to account for *global interactions*; the relative position of the target agent with respect to *all* other agents is calculated and processed using MLPs and a max-pooling operations to obtain a pooling vector for the target agent that takes into account all surrounding agents.

Deo and Trivedi [21] apply a similar social pooling system as [1], but instead of feeding the social occupancy grid through a fully-connected layer they process it using a *Convolutional Neural Network*. This means the *spatial structure* of the social occupancy grid is *preserved* by learning local features. This convolution-based method was shown to outperform the fully-connected method. Finally, the encoded social interactions are concatenated with the encoded positional motion history of the target agent to make trajectory predictions.

Chandra, Bhattacharya, Bera, and Manocha [13] approach the interaction modeling problem based on two core ideas. First, in dense traffic a road agent does not respond to all surrounding agents and second,

pair-wise agent interactions depend on difference between heterogeneous agents. These ideas are formalized through two encoding streams, respectively called the *horizon-* and *neighborhood stream*. Both of these streams use the *state space* of the agents located in their respective relevant region. This region is a semi-elliptical horizon in front of the target agent for the horizon stream and all agents in the scene for the neighbourhood stream. The state space contains information on the agent shape, velocity, position and driving behavior. These states are embedded and processed using an LSTM to obtain temporally relevant information. The resulting hidden states are pooled together in respectively a *horizon-* and *neighborhood map*, which are then processed using Convolutional Neural Networks to obtain information about scene dynamics around the target agent.

2.3.2.2 Attention-based

Sadeghian, Legros, Voisin, Vesel, Alahi, and Savarese [84] propose an interaction module that leverages an *attention-mechanism* to learn the most relevant static context cues of a scene. It does so by first extracting image features from a *raw top-down image* using a Convolutional Neural Network. These features are combined with the hidden state of the decoder RNN, which contain information on the agents' future trajectory. This information is fed through an a *physical attention module*, which learns the most influential static context features on the agents' predicted trajectory. The attention module combines two types of attention mechanisms, namely *single-source* and *multi-source*. The former learns to extract local visual cues in a single area of the image with an attention mechanism similar to [28], whereas the latter learns to extract local visual cues from different areas across the image using soft attention [101].

Sadeghian, Kosaraju, Sadeghian, Hirose, Rezatofighi, and Savarese [83] extends this work and adds a *'social'-attention module* to account for the dynamic context. A feature representation of the past trajectory of each agent is obtained using LSTMs and combined into a joint feature vector. This vector is fed into the social-attention module together with the hidden state of the decoder LSTM. By using the decoder LSTM hidden state, which contains information on the future trajectory of the target agent, the method can reason over which surrounding agents are most influential on the target agents' future trajectory. The trajectory prediction pipeline of [83] is outlined in figure 2.8.

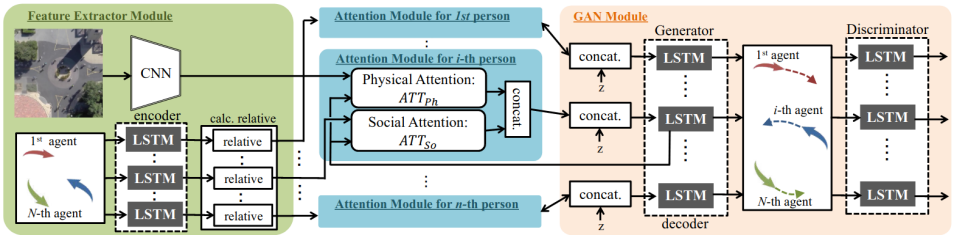


Figure 2.8: Trajectory prediction method from [83]. First, static context features are extracted from a raw image using a CNN and dynamic context features are extracted using LSTMs (left side). Next, the social- and physical attention modules are used for interaction modeling. The physical attention module takes as input the processed static context features & the decoder LSTM hidden states and the social attention module takes as input the processed motion history of the dynamic context & decoder LSTM hidden state (middle). Using the interaction-aware encoding a GAN is used to generate future trajectories (right).

2.3.2.3 Graph-based

Graph-based methods model the agents and contextual objects as nodes on a graph with the edges in between two nodes representing the pair-wise interactions. One can reason over such graphs in an *implicit* [44, 59, 33, 85] and *explicit* manner [38, 48]; implicit methods use a predefined graph structure and reason over the graph via the use of a certain message passing function. Explicit methods aim to first predict the underlying interaction graph explicitly, i.e. which nodes are connected and what type of connection define the pair-wise interaction. This predicted graph is then used to reason over via some message passing function. Both methods will be discussed below.

Graph-based: implicit interaction modeling Kosaraju, Sadeghian, Martín-Martín, Reid, Hamid Rezatofighi, and Savarese [44] propose a method that models interactions between the target agent and both static- and dynamic context. First, the agent-agent interactions are realized by constructing a fully-connected graph

where each node represents the motion history of an agent and each edge represents the pair-wise interaction between two connected nodes. The graph is updated using *Graph Attention* operations, i.e. the node embeddings are updated using a self-attention mechanism. The interactions with the static context is realized by performing *soft-attention* over the physical features relative to the target pedestrians' motion history, where the physical features are extracted from a *raw top-down scene image* using a Convolutional Neural Network. This is similar to [83] and [84], but they perform the attention relative to the agents' future trajectory embedding. The component of the model responsible for reasoning over the interactions is illustrated in figure 2.9

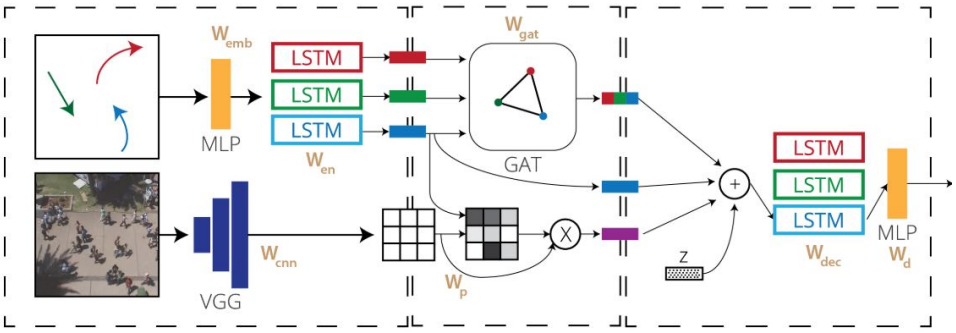


Figure 2.9: The component of the social-BiGAT model from [44] which is responsible for interaction modeling. First, static context features are extracted from a raw image using a CNN and dynamic context features are extracted using LSTMs (left side). Next, graph-attention is used to reason over dynamic context features (middle-top) and soft-attention to reason over the physical context features in relation to past motion history (middle bottom). Finally, the interactions embeddings and motion history embeddings are concatenated and decoded to obtain future trajectories (right)

This approach however, only uses the graph for spatial modeling, i.e. they essentially use it as a *sophisticated pooling mechanism* for the embedded motion history of each pedestrian. Accordingly, [33] and [59] propose different ways to improve this shortcoming. For example, Huang, Bi, Li, Mao, and Wang [33] put the graph structure to better use by also accounting for *temporal continuity of interactions*. Their method, *STGAT*, uses a similar approach as in [44], but the temporal continuity between interactions is preserved by employing a second LSTM connecting each graph output node embedding. It is shown that the performance is, on average, significantly improved by also modeling the temporal dependencies of interactions. Mohamed, Qian, Elhoseiny, and Claudel [59] propose *social-STGCNN*, which improves use of the graph structure by modeling the temporal dimension within the graph representation. This *spatio-temporal graph* is constructed by first embedding the motion history of all agents, in each timestep, as nodes on a spatial graph, G_t. The edges represent the interactions and edge weights are determined using a kernel function, i.e. the inverse L_2-distance between agents. To account for spatio-temporal dependencies, a second temporal graph, G, is introduced whose attributes are the set of attributes of the spatial graphs, G_t. The authors extend the conventional *spatial Graph convolution* operation to also account for the temporal domain. This spatio-temporal graph convolution is then performed to update the graph and obtain spatio-temporal embeddings for each node, i.e. agent. These embeddings are then used to generate a trajectory for the respective agent. A limitation of this method is that the spatial graph structure must be identical in each time step, i.e. if any agent is not present in a given frame in a sequence, the interactions with this agent cannot be modeled throughout the entire sequence.

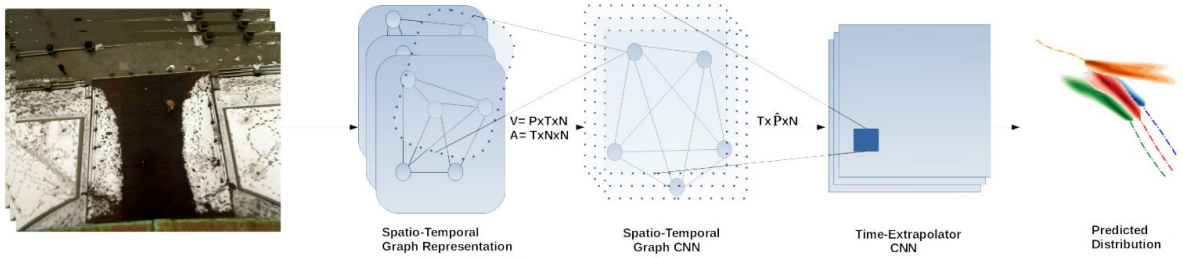


Figure 2.10: The social-STGCNN model from [59]. A spatio-temporal graph is constructed using the embeddings P of N pedestrians in T time frames. The agent embeddings are represented by V and the edges by A. Graph convolution operations are performed to update the spatio-temporal embeddings, after which the time-extrapolator CNN extends the temporal dimension of the embedding for the desired time steps into the future

Yau, Malekmohammadi, Rasouli, Lakner, Rohani, and Luo [106] improve upon [59] in three ways in their proposed method *Graph-SIM*. First, the agent embeddings in Social-STGCNN only contain coordinates. Graph-SIM embeds information on semantic class, size and motion status of each agent, as well as relative location to the target agent. Second, Social-STGCNN weighs interaction importance using a simple kernel function depending on the relative distance between agents, whereas graph-SIM makes the interactions dependent on a combination of Euclidean distance between agents and a distance metric based on the agents' location on the road and orientation. Third, the graph in Social-STGCNN is constructed in a simple manner, whereas graph-SIM constructs the graph in a more sophisticated manner; semantically identical agents are first clustered together based on velocity, direction and position, as objects interacting in a group usually exhibit similar behavior. Next, Inspired by [50], the target pedestrian is connected to all agents in a scene with a pedestrian-centric star graph and nodes that are within the same cluster are connected via a fully-connected graph. Next to modeling multi-agent interactions, the model additionally accounts for motion of the target pedestrian and ego-vehicle by encoding their motion history, i.e. position and velocity, using an LSTM. The complete model vastly outperforms the baselines [59, 93], where especially performance gains are obtained by adequately modeling group behavior. Another interesting observation is that results significantly improve by embedding both velocity and position of the target pedestrian- and ego-vehicle, instead of only one.

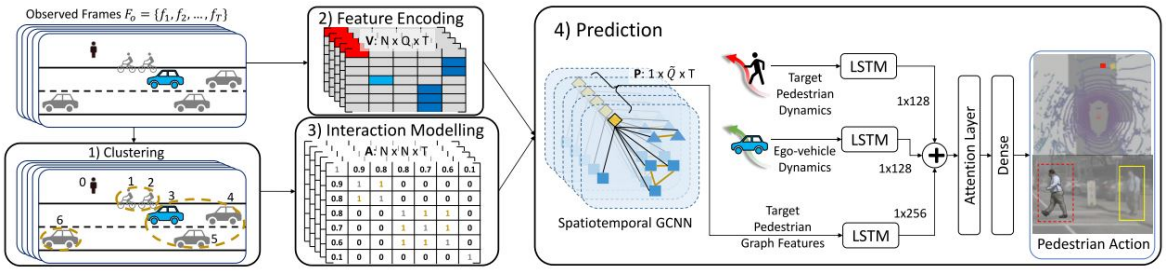


Figure 2.11: Graph-SIM model from [106]. The agents in the scene are clustered according to their similarity in behavior. Similar to Social-STCNN an embedding Q of all N agents in all T time frames is obtained. The clustered agents are transformed into an interaction tensor, which represent the edges in the spatio-temporal graph, and the agent embeddings represent the nodes. Spatio-temporal graph convolution is performed to update the graph and obtain target pedestrian graph features. Finally, these features are concatenated with a temporal embedding of the ego-vehicle & target pedestrian dynamics to make a action prediction.

Salzmann, Ivanovic, Chakravarty, and Pavone [85] account for dynamic context via a combination of a *spatial graph with temporal LSTMs and an attention module*. The first step is to construct a graph where the target agent node i is connected to a surrounding agent node j if the L_2-distance is within a certain threshold. This threshold corresponds to the *perception range* of the semantic class of agent j. The graph is modeled with directed edges only to account for *asymmetric influence* between nodes. e.g. a car driver tends to look further ahead than a pedestrian walking on the side walk. Next, the edge information of neighbouring agents of the same semantic class is aggregated and fed into a semantic class-specific LSTM. An attention module is used to combine the LSTM embeddings for each semantic class and determine the importance of each surrounding agent relative to the target agent. The interaction with the static context is implicitly modeled by encoding a local occupancy or semantic map, rotated to match the agents' heading.

Graph-based: explicit interaction modeling Kipf, Fetaya, Wang, Welling, and Zemel [38] starts with a fully connected graph, where each node represents an embedding of the agents' past trajectory. Message passing is performed to obtain edge embeddings with information on the *latent interactions* between nodes. Each edge embeddings is fed through a softmax to obtain a discrete probability distribution over K edge types. The latent interaction type between each node-pair is now sampled from this distribution to obtain the *latent interaction graph*. The decoder uses this graph to update the node representation by message passing and consequently generate trajectory predictions. Limitations include that the interaction graph cannot change over time during training, which is required for sufficiently training a model in scenarios where these graphs dynamically change, such as with autonomous driving. Moreover, it has not been applied to any real world trajectory prediction dataset and can only account for homogeneous agents. This is illustrated in figure 2.12.

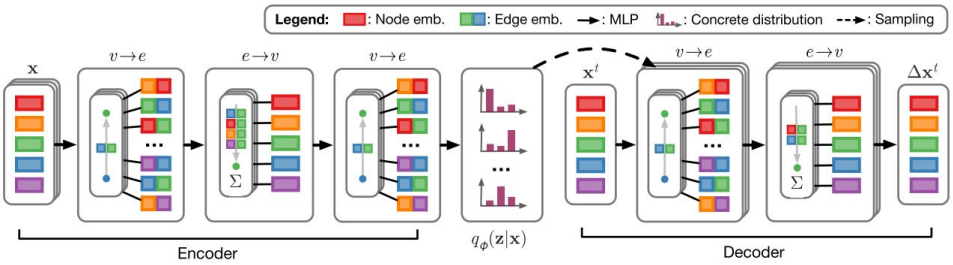


Figure 2.12: The NRI model from [38]. The encoder is a fully connected graph neural network which performs message passing to learn edge representations of the latent interaction between two nodes (left). Next, an edge type is sampled from each edge representation resulting in the final latent interaction graph structure (middle). The decoder then uses this graph structure, the motion history nodes and message passing operations to generate future trajectories. (right)

Li, Yang, Tomizuka, and Choi [48] improve on this work in three ways, namely by adding *static-context interactions*, allowing the latent interaction graph to *dynamically change* and introducing interactions between *heterogeneous agents*. Similar to [38], It starts with a fully-connected graph, but with an additional context node, called the *observation graph*. The context node contains some context embedding and each agent node embedding contains information on its own state, called *self-attributes*, as well as information on other nodes' states, called *social-attributes*. The self-attributes are embedded with a class-specific embedding function to account for heterogeneous agents. The nodes and edge embeddings are now updated using *Graph Attention*. The latent interaction graph is obtained using these edge embeddings in the same way as Kipf, Fetaya, Wang, Welling, and Zemel [38]. To make the interaction graph dynamic over time, the observation graph and corresponding static interaction graph are re-calculated every τ time steps. τ can be understood as the number of future time-step predictions using the same latent interaction graph. The re-calculated interaction graphs are connected temporally via a GRU because interaction graphs depend on their history. The output of each GRU cell is the adjusted interaction graph with time dependence. This method outperformed state-of-the-art baselines [33, 59] on real-life datasets.

2.3.2.4 Transformer-based

A recent development is the use of *Transformer-based* models in trajectory prediction literature [27, 108, 49]. The reason for this is that recent State-of-the-Art models in other domains, such as Natural Language Processing, rely on the Transformer-concept.

Li, Yang, Liang, Zeng, Ren, Segal, et al. [49] propose *Interaction Transformer*, which is an end-to-end framework consisting of three main components. First, multi-modal sensor data from LiDAR, maps and cameras are embedded and fused to contain a representation in *BEV-map* space. Second, this representation is fed into a *detection module* which detects spatial features of the surrounding agents. Last, and most importantly, the BEV representation and agent spatial features are fed into the *Transformer-based module*, which reasons over the interactions with the static- and dynamic environment and outputs agent interaction features. These features are used by a recurrent model to auto-regressively predict future agent features. In order to make the Transformer architecture work in the trajectory prediction domain, the authors made several changes. For example, the positional embeddings have been changed from absolute to relative, as the relative position between agents is most relevant in determining their interactions.

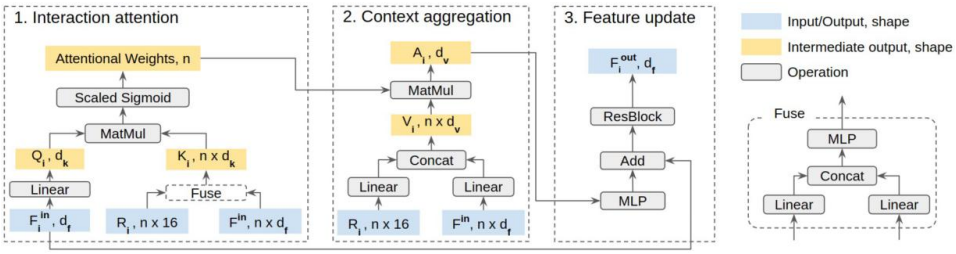


Figure 2.13: The interaction transformer module from [49]. First, the interaction attention part (1) calculates the attention weights using queries the features of all n agents, F^{in}, as queries and the spatial location, R_i, of the target agent relative to all other actors as keys. These weights are then used to determine which agent features are most important in the context aggregation part (2). Finally, the feature update part generates the future frame features F_{out} for the target agent.

The architecture proposed by [108] employs spatial- and temporal encoders based on the Transformer-architecture and uses them both parallel- and sequential to capture spatio-temporal interactions. While the temporal encoder is similar to the original Transformer encoder, the spatial encoder mainly leverages the *multi-headed attention mechanism* from Transformers on a fully-connected graph. This spatial transformer is named *TGConv* and ablation studies indicated that its spatial modeling capabilities were superior to existing counterparts, such as *GCN* and *GAT*. An interesting notion is that the benefit of the Transformer-based spatial modeling over alternatives diminished for less busy crowds.

2.4. Evaluation

The strengths and weaknesses of any trajectory prediction algorithm are evaluated on *datasets* and are quantified using *metrics*. Sharing metrics and datasets between methods provides a basis to compare performance. This section discusses the datasets and metrics used in the reviewed literature.

2.4.1. Datasets

This section first provides a tabular overview of the datasets used by the previously discussed literature and follows up with a more detailed explanation regarding these datasets.

2.4.1.1 overview

An overview of datasets used in the reviewed literature is provided below. For each dataset the following attributes are indicated; the *domain* and type of *environment* where it has been recorded, the *sensors* used to capture the data, the *type of annotations*, the *total number of frames* and how many of these frames are actually annotated, and finally the *number of annotated agents* and how many of those are *unique*.

				Sensors					Annotations							# of (annotations) agents		
Dataset	**Year**	**Type**	**Dom**	**Mono**	**Stereo**	**LiDAR**	**RaDAR**	**GPS/IMU**	**Annot freq**	**Pos**	**Track**	**C/NC**	**Agent att.**	**Scene att.**	**# of (annotated) frames**	**Pedestrian**	**Rider**	**Vehicle**
NGSIM	2006	Traf.	Bird	✓	-	-	-	-	10Hz	Coords	✓	-	✓	✓	✓	-	-	2.9K
UCY	2007	Surv.	Bird	2.5Hz	-	-	-	-	2.5Hz	Coords	✓	-	Gaze	-	✓	(✓) 600	-	-
ETH	2009	Surv.	Bird	2.5Hz	-	-	-	-	2.5Hz	Coords	✓	-	Vel	-	✓	(8.9K) 650	-	-
Stanford Drone	2016	Traf.	Bird	✓	-	-	-	-	✓	2D BB	✓	-	interactions	Sem map	(929K)	11.2K	6.4K	1.3K
CityWalks	2019	Urb.	2D ego	1x 30Hz	-	-	-	-	30Hz	2D BB	✓	-	-	-	(215K)	2.2K-3.6K	-	-
JAAD	2017	Traf.	2D Ego	1x 30Hz	-	-	-	-	30Hz	2D BB	✓	✓	✓	Agent/driver actions	(75K) 82K	(381K) 2.8K	-	-
PIE	2019	Traf.	2D ego	1x 30Hz	-	-	-	✓	30Hz	2D BB	✓	✓	✓	obj. BB + annot	(293K) 911K	(740K) 1.8K	-	-
STIP	2020	Traf.	2D ego	3x 20Hz	-	-	-	-	2/20Hz	2D BB	✓	✓	-	-	(1.1M) 1.1M	(3.5M) 25K	-	-
TITAN	2020	Traf.	2D ego	1x 60Hz	-	-	-	100Hz	10Hz	2D BB	✓	-	actions	-	(75K) 750K	(395K) 3.6K	-	(250K) 5.5K
KITTI Tracking	2012	Traf.	3D ego	-	2x 10Hz	1x 10Hz	-	100Hz	10Hz	3D/2D BB	✓	-	-	-	18K	✓	✓	✓
Apollo Scape	2018	Traf.	3D ego	1x	-	1x	1x	10Hz	2Hz	3D BB	✓	-	dir	-	93K	16.2K	5.5K	60.1K
NuScenes	2019	Traf.	3D ego	6x12Hz	-	5x 20Hz	1x 13Hz	100Hz	2Hz	3D/2D BB	✓	-	pose, activity	raster/sem map	(40K) 1.4M	(222K) 9.1K	(944K)	
Argoverse	2019	Traf.	3D ego	7x 30Hz	2x 5Hz	2x 10Hz	-	GPD	10Hz	3D BB	✓	-	-	Semantic map	(350K)	(132K) 1.5K	(11K) 7K	✓
Waymo Perception	2019	Traf.	3D ego	5x 10hz	-	5x 10Hz	-	-	10Hz	3D/2D BB	✓	-	-	Raster/sem map	(800K)	2D: (2.7M) 58K. 3D: (2.8M) 23K	2D: (81K) 1.7K. 3D: (67K) 620	2D: 6.1M (60k). 3D: 9M (194k)
PePScenes	2020	Traf.	3D ego	6x12Hz	-	5x 20Hz	1x 13Hz	100Hz	10Hz	3D/2D BB	✓	✓	pose, activity	raster/sem map	(1.4M)	(845K)	(3.58M)	
Waymo Motion	2021	Traf.	3D ego			?			10Hz	3D BB	✓	-	dir/vel	(Sem map, traffic-light states)	(20M)	(534M) 7.6M		
Euro-PVI	2021	Traf.	3D ego	2x 10Hz	-	1x 10Hz	-	✓	10Hz	Coords	✓	-	-	Sem map	83K	6.2K	1.6K	-

Table 2.1: Overview of the most common datasets used in the reviewed trajectory prediction literature, where the headers are explained as following. **Dataset:** dataset name, **Year**: year of release, **Type:** Type of dataset, which could be surveillance, traffic or urban. **Dom:** Domain, **Mono:** Presence (specifications) of monocamera, **Stereo:** Presence (specifications) of stereo cameras, **LiDAR:** Presence (specifications) of LiDAR, **RaDAR:** Presence (specifications) of RaDAR, **GPS/IMU:** Presence (specifications) of GPS or IMU, **Annot freq:** Annotation frequency, **Pos:** Type of positional agent annotations (2D/3D BB = 2D/3D bounding box, Coords = point coordinates), **Track:** Presence of tracking IDs **C/NC:** Presence of cross/no-cross labels **Agent Att.:** Presence of agent attribute annotations (Gaze = gaze direction, Vel = velocity, Dir = heading direction, ✓= too many types) **Scene Att:** Presence of scene attribute annotations (✓= too many types), **# of (annotated) frames:** Total number of frames (if provided, else ✓), where the number between brackets indicates the total number of annotated frames, **# of (annotated) agents**. Total number of agents (if provided, else ✓), where the number between brackets indicate the total number of (per-frame) annotations for these agents

2.4.1.2 Ego-centric naturalistic datasets

In the collection of ego-centric naturalistic datasets a divide is present between the datasets captured from a driving vehicle and datasets captured from a walking pedestrian. Both are discussed below.

Driving datasets

The *Joint Attention in Autonomous Driving (JAAD)* dataset [71] contains 346 videos that have been recorded in 5 different cities in Europe and North-America by an on-board camera with a resolution of 1920x1080 or 1280x720. The dataset annotations come in three different categories. First, pedestrian bounding boxes with tracking IDs, occlusion ratios and crossing/not-crossing actions. Second, pedestrian attributes with regards to its state before crossing, how the pedestrian becomes aware of an approaching vehicle and what action the pedestrian takes in response to the approaching vehicle. Third, scene attributes that list the environmental contextual elements. JAAD is used in a variety of action/intention [50, 70, 71, 9, 72] and trajectory prediction papers [70, 72, 104, 98]. The dataset has no ego-motion information, but it does contain high-level driver actions such as *speeding up, slowing down* etc. However, this information is less informative than actual ego-motion information [76].

The *Pedestrian Intention Estimation (PIE)* dataset [70] is the first dataset to have annotations for the latent intention of pedestrian crossing. Note that these annotations refer to the intention of crossing instead of the crossing action. It was recorded during daytime in 1 city in Canada with only sunny and over-cast weather conditions using a single camera with resolution 1920x1080. Similar to JAAD, the dataset includes pedestrian bounding boxes with tracking IDs and occlusion ratios. Moreover, pedestrian attributes are included for actions and the true intention of crossing. Finally, scene attributes and object annotations are present, which includes bounding boxes for pedestrian crossings, vehicles and stoplights. Contrary to JAAD, this dataset also comes with on-board diagnostics sensor data that provides GPS coordinates and vehicle information, such as velocity and heading angle. PIE is used in [70, 72, 93, 104, 98].

The *Stanford-TRI Intent Prediction (STIP)* dataset [50] was recorded under various weather conditions in dense urban environments in 8 different US cities. The recordings are made with 3 different cameras facing left, right and straight ahead, all with a resolution of 1216x1936. The frames have been labeled for pedestrian bounding boxes at 2Hz, which has subsequently been extrapolated to annotations at 20Hz. Each bounding box has tracking information and crossing/not crossing action information. Access to this dataset is restricted to universities and non-commercial organizations. The only paper to use this dataset is [50].

The *Trajectory Inference using Targeted Action priors Network (TITAN)* [54], has been recorded in highly interactive urban environment in Tokyo by a single camera with 1920x1200 resolution. Additionally, a IMU

sensor recorded the ego-motion at 100Hz. The TITAN dataset has been labeled for pedestrians, 2-wheel and 4-wheel vehicles, all of which contain tracking information. Vehicles have an motion-status label, as well as a door/trunk status label specifically for 4-wheel vehicles. Pedestrians are accompanied by an age group label and and a large variety of hierarchically annotated action labels. That means each pedestrian can have up to 5 different action labels, ranging from individual atomic actions (e.g. laying down) to contextual actions (e.g. getting into a car, bicycling) to transportive (e.g. pushing) to communicative (e.g. looking at phone). Access to this dataset is restricted to universities and non-commercial organizations. The only paper to use this dataset is [54].

The *NuTonomy Scenes (NuScenes)* dataset [11] has been recorded in 2 cities, namely Boston and Singapore, in multiple weather conditions in day- and night conditions. The sensor modalities included 6 cameras with resolution 1600x900, 1 LiDAR, 5 RaDARs and a GPS & IMU. It contains over 1000 hand-selected driving scenes, which are 20 seconds long and annotated at 2Hz. The annotations include 3D bounding boxes over 23 agent- and object (including static context) classes with corresponding attributes such as visibility, activity and pose. Additionally, there are raster maps and semantic maps with 11 semantic classes available. The *Pedestrian Prediction on NuScenes (PePScenes)* dataset [75] is an extension of the NuScenes dataset. The annotation density is increased from 2Hz to 10Hz by interpolation and additionally behavioral labels are added for crossing actions to a subset of the pedestrians. Both NuScenes and PePScenes also contain 2D bounding boxes, which are projections of the 3D bounding boxes on the image plane. Therefore, the 2D bounding boxes do generally not have a tight fit around an agent. The NuScenes dataset has been used in [85, 49, 7] and the PePScenes dataset in [106, 76].

The*Waymo Perception* dataset is similar to NuScenes datasets and has been recorded in three cities in North-America in Sunny, cloudy & rainy weather during the day- & night. The sensor modalities include 5 cameras with resolution 1920x1280 and 5 LiDARs. It contains annotations for pedestrians, riders, vehicles and signs with both 2D- and 3D bounding boxes. The main purpose of the Waymo Perception dataset is detection & tracking. To better accommodate trajectory prediction research, Waymo has recently released the *Waymo Motion* dataset [23]. This dataset has been recorded in six cities in the United States and covers over 100.000 scenes spread out over 570 hours of data. The data has been specifically selected to contain interesting scenarios with interactions. In addition to 3D bounding box labels for pedestrians, riders & vehicles, the dataset also includes high-definition 3D maps, velocity & heading annotations and traffic signal states.

The *KITTI tracking* dataset [25] has been recorded in- and around Karlsruhe in Germany using two sets of stereo cameras with a resolution of 1392x512, a LiDAR and a IMU/GPS localization unit. It is a relatively small dataset, counting 50 sequences with varying length from 78 to 1176 frames. This dataset has been used in [46].

The *Cityscapes* [19] dataset is not intended for the trajectory prediction task due to the absence of bounding box & corresponding track ID annotations. Hence, it has been excluded from table 2.1. Nonetheless, it has been used for trajectory prediction in [6], who added bounding box annotations and tracks themselves. CityScapes has been recorded in over three different seasons. It contains 5000 images, recorded in 27 different cities, with pixel-level instance segmentation of 30 different classes and 20000 additional images, recorded in another 23 different cities, with more coarsely annotated semantically segmented labels.

The *Euro-PVI* dataset [7] is a dataset that focuses on scenarios where interactions between the ego-vehicle and surrounding pedestrians or riders play a significant role. The clips are recorded dense urban scenes of Brussels and Leuven in Belgium with two cameras with a resolution of 1280x806 and a LiDAR (Velodyne HDL-64E). Annotations include world coordinates for agents, ego-motion and semantic maps of the environment. This dataset has been used in [7].

Other ego-centric walking datasets

The CityWalks dataset [90] is an ego-centric dataset recorded from the first-person view of a walking pedestrian. A total of 358 video sequences have been recorded using a camera with 1280x720 resolution in many different settings. These settings including in- and outdoor, during the day- & night, whilst sunny, cloudy, rainy and snowy and in a total of 21 cities in 10 European countries. The frames are annotated for 2D pedestrian bounding boxes. These are obtained using different detection algorithms, namely YOLO (2201 unique tracks) and Mask-RCNN (3623 unique tracks). The dataset was used in [90]

2.4.1.3 Top-down surveillance- and traffic datasets

The UCY dataset [47] is an outdoor surveillance top-down dataset comprising of 3 scenes, namely the sparsely populated scenes ZARA-01 & ZARA-02 and the densely populated UCY. The ZARA scenes It contains pedestrian position annotations with tracking labels and gaze directions. The pedestrians show complex behavior, such as walking in unpredictable non-linear fashion and walking in groups. It has been widely used as a benchmark for top-down methods [85, 83, 44, 29, 1, 108, 97, 59, 33]. The ETH dataset [65] is very similar to UCY and contains 2 moderately crowded sub-sets, namely ETH and Hotel. It also has seen widespread use as a benchmark dataset for top-down methods [85, 83, 44, 29, 1, 108, 97, 59, 33].

The *Stanford Drone Dataset (SDD)* [79] also is an outdoor surveillance top-down dataset, but with heterogeneous agents. The dataset is recorded in 20 unique scenes that contain physical obstacles that the agents avoid. It has been used in [46, 83, 48, 84]. The dataset contains top-down 2D bounding box annotations, scene semantics and additionally agent-agent & agent-environment interactions.

The *Next Generation Simulation (NGSIM)* [18, 17] dataset has been recorded on the US Highway 101 and interstate 80 and hence contains only vehicles. This top-down dataset has annotations at 10Hz for local and global positions, velocities, lanes, vehicle type and parameters. It has been used by [13, 21]

The Apollo Scape dataset [51] contains 155 minutes of clips with ± 1000km of trajectories for different agents. It has has been recorded under various lighting conditions and varying traffic densities using an RGB camera with resolution 1920x1080, a VeloDyne HDL-64ES3 LiDAR, a Continental ARS408-21 RaDAR and localization sensor. The annotations include full 3D bounding boxes with heading angle. Moreover, even though a localization system is present the ego-motion not among the annotations. This dataset has been used by [51].

2.4.2. Metrics

The type of metrics used for evaluation of future trajectory predictions is dependent on the prediction output type. For deterministic output types, a *geometric measure* of trajectory similarity or final displacement is required. In case the prediction output is stochastic a *probabilistic measure* for similarity between probability distributions is required. Table 2.2 summarizes the use of different metrics throughout the reviewed literature. The metrics are discussed in more detail below.

	Metric	Used by
Geometric	ADE	[102, 91, 72, 54, 90, 66, 46, 13, 78, 85, 51, 21, 84, 83, 108, 33, 29, 1, 44, 48, 49, 104]
	FDE	[91, 72, 54, 90, 66, 13, 85, 51, 84, 83, 108, 33, 29, 1, 44, 48, 7, 49, 104]
	MSE	[6, 70, 66, 69, 104, 38, 62]
	C_{MSE}	[70, 69, 104, 62]
	CF_{MSE}	[70, 69, 104, 62]
	A_{IOU}	[100, 90]
	F_{IOU}	[105, 53, 54, 90]
	$(A/F)_{RMSE}$ on bbox coordinates	[72]
	NADE	[100]
probabilistic	NLL	[6, 21]
	KDE-NLL	[104, 85, 7]
	Best-of-N	[66, 104, 85, 59, 1, 108, 33, 29, 44, 48, 7]

Table 2.2: Overview of the used geometric and probabilistic metrics in the reviewed literature

2.4.2.1 Geometric metrics

The majority of methods utilize a metric based on some sort of displacement error. The most commonly used ones are *Average Displacement Error (ADE)* and *Final Displacement Error (FDE)*. The former takes the

euclidean distance between the predicted and ground truth trajectory and averages the distances across all timesteps. The latter does the same, but instead of averaging it only takes the euclidean distance for the last timestep of the prediction horizon. In the 2D on-board domain ADE [105, 70, 100, 72, 54, 90] and FDE [105, 102, 70, 72, 54, 90] are widely adopted to measure the euclidean distance between the ground truth and predicted bounding box centroids. In the 3D on-board and top-down domain, ADE and FDE are used to measure the euclidean distance between predicted coordinates [66, 46, 13, 78, 85, 51, 21, 84, 83, 108, 33, 29, 1, 44, 48, 49, 104]

Some papers adopt distance metrics similar to ADE and FDE, such as *Mean Squared Error (MSE)*, *Average Root Mean Square Error (ARMSE)* and *Final Root Mean Squared Error (FRMSE)*. MSE is solely used in the 2D ego-centric domain [6, 70, 66, 69, 104, 62] to measure the squared distance error averaged over all bounding box coordinates across the prediction horizon. Alternatively, the mean squared error is calculated over the bounding box centroids across the prediction horizon, C_{MSE} [70, 69, 104, 62] or for the final timestep only, CF_{MSE} [70, 69, 104, 62]. Another alternative is A_{RMSE} [72], which calculates the average euclidean distance between the predicted and ground truth bounding box coordinates averaged over all timesteps, or F_{RMSE} [72] calculating the Root Mean Square Error over the bounding box coordinates in the last predicted timestep. Styles, Ross, and Sanchez [91] calculate the euclidean distance at different timesteps and calls it $DE@t$, where @ indicates the future timestep where the evaluation is made. To make a fair comparison between datasets with different image resolutions Xiong, Flohr, Wang, Wang, Wang, and Li [100] additionally normalise ADE with respect to image resolution.

There are also metrics that are used to jointly evaluate scale and position of the predicted bounding box. These include *Average Intersection Over Union* (AIOU) [100, 90], which calculates the intersection over union between the predicted and ground truth bounding box, and *Final Interaction Over Union (FIOU)* [53, 105, 54, 90], which calculates the intersection over union between the predicted and ground truth bounding box.

In addition to these distance metrics, two papers attempt to integrate a metric to measure social compliance between different actors. For example, [49] introduces Trajectory Collision Rate and [83] uses the average % of colliding pedestrians per frame for each scene as a try-out metric.

2.4.2.2 Probabilistic metrics

Geometric metrics can not be used for predictions with uncertainty or which are multi-modal. In these cases, probabilistic metrics are required. For example, *Negative Log-Likelihood (NLL)* provides a measure of the assigned probability to the true sequence by the predicted distribution. It is used in the 2D on-board domain [6] and in the top-down domain [21]. An alternative to this is the *Kernel Density Estimation-based Negative Log Likelihood (KDE-NLL)* [104, 7, 85], which evaluates the ground truth negative log-likelihood under the predicted distribution estimated by a Gaussian Kernel.

Some papers use a sampling approach to evaluate a stochastic trajectory prediction using the *best-of-N* metric [66, 104, 85, 59, 1, 108, 33, 29, 44, 48, 7], where N trajectories are sampled from the predicted distribution and the the evaluation is performed with the best one. Concretely, these metrics take shape in the form of *minimum Average Displacement Error (mADE), minimum Final Displacement Error (mFDE), minimum Final Intersection Over Union (mFIOU).*

3

Method

This chapter elaborates on the proposed method in detail and concludes with the contributions of this work.

3.1. Overview

The proposed method, Interaction-Aware Pedestrian Trajectory Prediction (*IA-PTP*), performs trajectory prediction while accounting for interactions with the static- and dynamic context in the 2D on-board domain. At time step t, given a sequence of video frames for the last τ time steps, $\overline{\mathbf{X}}_t = \{\mathbf{X}_{t-\tau+1}, ..., \mathbf{X}_t\}$, where $\mathbf{X}$ contains information for the target pedestrian (i.e. bounding box coordinates and appearance) and contextual objects (i.e. binary instance masks) the goal is to predict the future trajectory of the target pedestrian $\mathbf{Y}_t = \{\mathbf{y}_{t+1}, ..., \mathbf{y}_{t+T}\}$, where $\mathbf{y}$ represents four bounding box corner coordinates and T the prediction horizon.

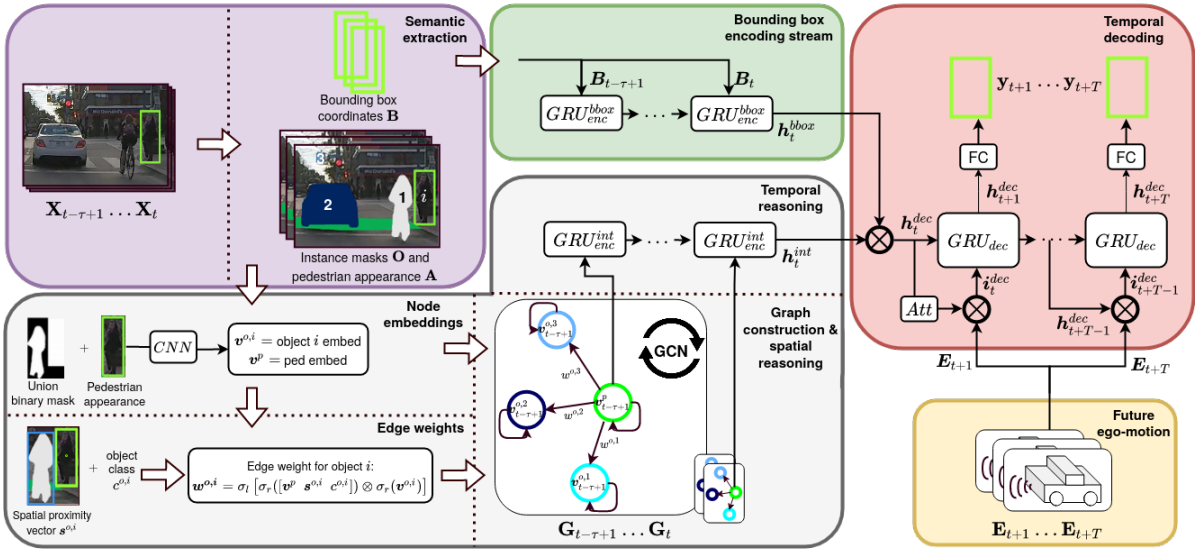


Figure 3.1: Overview of the proposed method, IA-PTP.

An overview of the proposed model is illustrated in figure 3.1, where each component is colourised to enable easy referral to components in the figure later on. First, the target pedestrian and other semantic entities are extracted from the observation data (purple). The extracted information is then processed using two separate encoding streams. The first encodes the motion history of the pedestrian (green). The second encodes the interactions between the pedestrian and its static- and dynamic environment (light-grey), and is inspired by *STIP* [50]. The embeddings resulting from each encoding stream, respectively $\boldsymbol{h}_t^{bbox}$ and $\boldsymbol{h}_t^{int}$, are fused to form a joint representation $\boldsymbol{h}_t^{dec}$. This joint representation serves as the initial hidden state- and input for the recurrent decoder (red). For the decoder input specifically, the representation is first updated using a soft-attention mechanism to generate $\boldsymbol{i}_t^{dec}$. Each decoder cell output is linearly mapped using a fully

connected layer to obtain a bounding box prediction $\mathbf{y}_{t+n}$, where $n \in [1..T]$. Optionally, the future ego-motion is fused with the input of each decoder cell (yellow module).

The remainder of this chapter will provide a detailed elaboration of the architecture. More specifically, the semantic extraction is discussed in section 3.2, after which the interaction modeling stream is elaborated in section 3.3. The bounding box encoding stream is described in section 3.4. Finally, section 3.5 discusses the temporal decoding, including the fusion of future ego-motion.

3.2. Semantic extraction of objects and pedestrian

First, each of the τ observation frames is parsed for the *target pedestrian* and *contextual objects*. This step is illustrated in figure 3.1 by the purple module. The parsing procedure for the target pedestrian and contextual objects is discussed below.

Target pedestrian the pedestrian location is defined by four ground truth bounding box corner coordinates in pixels, i.e. x- and y-pixel location for the bounding box top-left and bottom-right. The ground truth is used instead of predictions from a pedestrian detector because it makes model performance independent of pedestrian detector performance. This allows for a more careful evaluation of the proposed method. Moreover, the large majority of previous work [105, 70, 104, 62, 98, 72, 54] uses ground truth coordinates. Thus, doing the same provides a more solid basis for comparison.

Contextual objects the contextual objects are detected and semantically segmented using an off-the-shelf instance segmentation network, *Seamseg* [68], pre-trained on the *Mapillary Vistas V1.0* dataset [61]. *Seamseg* is the network of choice because of the open-source code, available pre-trained model on Mapillary Vistas, as well as ease of implementation while having decent performance. Pre-training on Mapillary Vistas is chosen because it contains pedestrian crosswalks, contrary to the alternative, i.e. the CityScapes dataset [19]. The parsed contextual objects represent only a subset of all labelled instance categories contained within the Mapillary Vistas dataset. The objects in this subset are called the *contextual objects of interest* and the selection is based on the set of objects used in *STIP* [50]. An overview of all available instance categories within the Mapillary Vistas dataset is provided in appendix A.1. An overview of the contextual objects of interest is provided in table 3.1.

Cue type	Category	Objects
Dynamic- or static	Vehicles	Bicycle, Bus, Car, Motorcycle, Trailer, Truck
Static	Infrastructure	Plain crosswalk, Zebra crosswalk, Traffic light

Table 3.1: The contextual objects of interest including their cue type and category

3.3. Interaction modeling stream

modeling the interactions between the target pedestrian and contextual objects is approached from a graph-based perspective. The interaction modeling stream is inspired by *STIP* [50] and contains three components, which are shown within the (light-gray) module in figure 3.1. The first step is to use the parsed information described in the previous section to construct τ spatial graphs. For each graph, the nodes contain a representation of the target pedestrian or a contextual object. The edges between two nodes define the interaction importance (section 3.3.1). Second, Graph Convolution operations are performed to reason over each graph. This process aims to inform the pedestrian node of the presence of different contextual objects, i.e. update the pedestrian node representation based on the object node representations (section 3.3.2). Finally, the pedestrian node of each graph is connected via a temporal GRU to aggregate the temporal correlations between the spatial interactions (section 3.3.3).

3.3.1. Graph construction

For each observed frame, the parsed information is used to construct a graph that represents the spatial relationships between the pedestrian and each contextual object. Assume a graph to be defined as $\boldsymbol{G} = (V, E)$, where $|V| = N_{node}$ is the number of nodes and $|E| = N_{edge}$ is the number of edges. The square adjacency matrix $\boldsymbol{A} \in R^{N_{node} x N_{node}}$ stores information on which nodes are connected, as well as the importance of these connections. Formally, the importance of each connection is expressed using an edge weight, w. The graph

construction will be discussed in two components. First, the *node definitions* and second, the *adjacency-matrix and edge weights*.

3.3.1.1 Node definitions

The graph contains two types of nodes, namely one for the target pedestrian and one for each contextual object. Both are discussed below.

- **Target pedestrian node:** contains an embedding of the pedestrian appearance. The appearance provides implicit cues on pedestrian behaviour, such as body- and head orientation or hand gestures.
- **Object nodes:** contain an embedding of the binarised union of the pedestrian bounding box and the object instance mask. The union is chosen, as opposed to only the object instance mask, because it captures the relative location- and shape of an object with respect to the pedestrian. An example of a union binary mask is provided in figure 3.2.

The pedestrian appearance image patch or binary union masks are processed in three steps. First, the image patch or union mask is reshaped to $224x224$ using bilinear interpolation to prepare it for CNN processing. Next, the reshaped image patch or union mask is fed through a *ResNet-18* backbone [30] pre-trained on *ImageNet* [82] to generate a a 512-dimensional output embedding for each node. *ResNet-18* is the backbone of choice because it is relatively fast in terms of inference time while having decent performance [8].

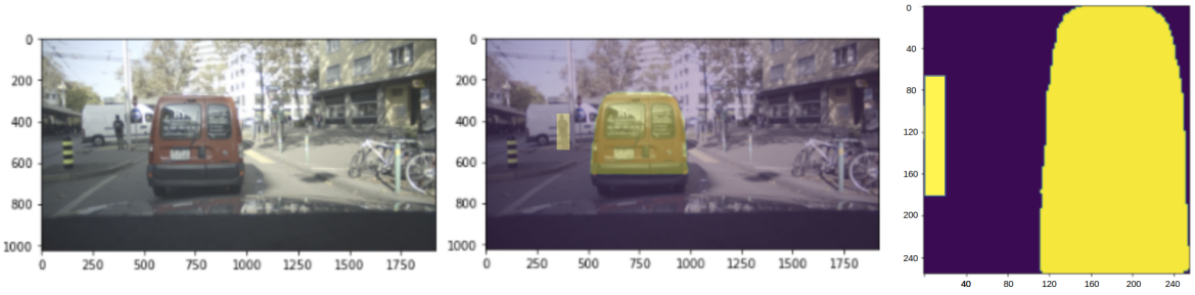


Figure 3.2: Visualization of the binary union mask representing an object node. left: current frame. Middle: object instance mask and pedestrian bounding box printed over the current frame. Right: cropped and reshaped binary union mask covering both the pedestrian and object instance.

3.3.1.2 The adjacency matrix and edge weights

The graph aims to inform the pedestrian about surrounding contextual objects that influence its future behaviour. This is accommodated by connecting each object node with the pedestrian node, creating a pedestrian-centric star-graph. Each connection, i.e. edge, is weighted to differentiate between more- and less important objects. By making the edge weights a learnable feature of the graph, the model becomes more explainable regarding which objects have the most- and least influence on trajectory prediction performance. Thus, the adjacency matrix A of this pedestrian-centric star-graph with weighted edges takes the shape of a symmetric matrix with unit weights corresponding to the self-loops on the diagonal and learned weights on the first column- and row- corresponding to the bi-directional weighted edges between the target pedestrian and each object.

$$\boldsymbol{A}[i,j] = \begin{cases} 1 & i = j \\ w^{o,i} & i = 1, j \neq 1 \\ 0 & else. \end{cases} \tag{3.1}$$

The edge weight $w^{o,i}$ between object i and the pedestrian is based on the *binary union mask* of the object i- and pedestrian, i.e. object i node embedding, as well as a *spatially-aware pedestrian appearance embedding*, i.e. an extended version of the pedestrian node embedding. Both elements are used as follows in the edge weight calculation.

- *spatially-aware pedestrian appearance embedding*: the pedestrian node embedding, $\boldsymbol{v}^p$, is represented by the pedestrian appearance. This embedding is concatenated with a spatial vector $s^{o,i}$, where $s^{o,i}$ captures the spatial relationship between the bounding box of object i and the pedestrian bounding

box. Additionally, object class $c^{o,i}$ is added. The results is a pedestrian appearance embedding, $\boldsymbol{v}^{ps,i}$, that is spatially- and class-aware of object i. Finally, this is fed through a fully-connected layer with a *ReLU* activation function to get $\hat{\boldsymbol{v}}^{ps,i}$. The calculation of $\hat{\boldsymbol{v}}^{ps,i}$ is shown in equation 3.2 and the calculation of the spatial relationship vector $s^{o,i}$ is shown in equation 3.3.

$$\hat{\boldsymbol{v}}^{ps,i} = \sigma_r\left(\boldsymbol{v}^{ps,i}\right) = \sigma_r\left(\begin{bmatrix}\boldsymbol{v}^{p} & \sigma_r(\boldsymbol{s}^{o,i}) & c^{o,i}\end{bmatrix}^T\right) \tag{3.2}$$

$$\boldsymbol{s}^{o,i} = \begin{bmatrix}\Delta x^{o,i}_{min} & \Delta y^{o,i}_{min} & \Delta x^{o,i}_{max} & \Delta y^{o,i}_{max} & \Delta x^{o,i}_{c} & \Delta y^{o,i}_{c} & w^{o,i} & h^{o,i}\end{bmatrix}^T \tag{3.3}$$

Where $\Delta x^{o,i}_{min}$, $\Delta x^{o,i}_{max}$, $\Delta y^{o,i}_{min}$ and $\Delta y^{o,i}_{max}$ indicate the minimum- and maximum pixel distance in bounding box corners in x- and y-direction, $\Delta x^{o,i}_{c}$ and $\Delta y^{o,i}_{c}$ indicate the pixel distance between bounding box centers in x- and y-direction, $w^{o,i}$ and $h^{o,i}$ indicate the pixel width- and height of the union.

- *Binary union mask embedding*: the object node is represented by a binary union mask embedding, $\boldsymbol{v}^{o,i}$. This embedding contains information regarding the relative location between object- and pedestrian, as well as on object shape. The mask is embedded using a fully-connected layer with a *ReLU* activation function, shown in equation 3.4.

$$\hat{\boldsymbol{v}}^{o,i} = \sigma_r\left(\boldsymbol{v}^{o,i}\right) \tag{3.4}$$

Using the two above calculated components, the edge weight is obtained according to equation 3.5.

$$\begin{aligned}\hat{\boldsymbol{w}}^{o,i} &= sigmoid\left(\hat{\boldsymbol{v}}^{ps,i} \otimes \hat{\boldsymbol{v}}^{o,i}\right)\\ w^{o,i} &= \overline{\hat{\boldsymbol{w}}^{o,i}}\end{aligned} \tag{3.5}$$

Where the spatially-aware pedestrian appearance embedding, $\hat{\boldsymbol{v}}^{ps,i}$, and the binary union mask embedding, $\hat{\boldsymbol{v}}^{o,i}$ are combined via an element-wise multiplication, $\otimes$. The combined information is activated using a *sigmoid* activation, resulting in $\hat{\boldsymbol{w}}^{o,i}$. By taking the average over all of the embedding's elements, a final edge weight value, $w^{o,i}$, is obtained

3.3.2. Spatial reasoning over the graph

A graph message passing algorithm is adopted to reason over each spatial graph, i.e. to refine the pedestrian node embedding with information from the object node embeddings. Specifically, the Graph Convolution [39] operation is implemented, which is defined by equation 3.6:

$$\boldsymbol{Z} = (\boldsymbol{A} \cdot \boldsymbol{X})\boldsymbol{W}^T \tag{3.6}$$

Where $\boldsymbol{A}$ is the adjacency matrix, $\boldsymbol{W}$ the learnable weight matrix, $\boldsymbol{X}$ the feature matrix containing the row-wise concatenated node embeddings and $\boldsymbol{Z}$ is the refined feature matrix. Following [50], A is not normalised, and no activation is added to the refined feature matrix, even though that is customary in the original Graph Convolution Network implementation. Moreover, only one layer of graph convolution is performed. This configuration showed the best results, as shown in the experiments in appendix A.2.2.

3.3.3. Temporal reasoning

Not only the spatial interactions matter, but also the temporal correlations between spatial interactions. For example, without temporal correlations, a pedestrian can be aware of a nearby car at time-step n, and have no awareness of the same vehicle at time-step $n+1$. To encourage temporal consistency, a GRU connects the pedestrian nodes between each graph.

3.4. Bounding box encoding stream

As has been shown by previous work, the motion history of the target pedestrian is a cue for its future trajectory [6, 53, 72, 54, 70, 93, 100, 91, 102, 90, 105, 104, 66, 92]. Therefore, a bounding box encoding stream is adopted that uses a *GRU* to embed the pedestrian's motion history, as is illustrated by the red module in figure 3.1.

Given an observation sequence of pedestrian bounding box coordinates, $\boldsymbol{B}_t = \boldsymbol{b}_{t-\tau+1}, ..., \boldsymbol{b}_t$, the GRU encodes temporal correlations between these into a final hidden state embedding $\boldsymbol{h}_t^{bbox}$. Each bounding box is described by x- and y-pixel location for the bounding box top-left and bottom-right pixel coordinates, i.e. $\boldsymbol{b}_t = \begin{bmatrix} x_t^{tl} & y_t^{tl} & x_t^{br} & y_t^{br} \end{bmatrix}^T$.

The motion history is embedded separately from the interaction modeling stream instead of jointly, because this was proven most beneficial for model performance in experiments, as shown in appendix A.2.1.

3.5. Temporal decoding

The embeddings generated by the interaction modeling- and bounding box stream combined, contain a rich representation of the scene history. The final step is to use this information in a useful way to generate future trajectory predictions. To accommodate this, a *GRU* decoder is adopted that uses both encoding stream embeddings, and optionally ego-motion information. The decoding module is illustrated in yellow in figure 3.1.

First, the final hidden state, or embedding, of the interaction modeling- and bounding box encoding streams are aggregated into a joint representation, $\boldsymbol{h}_t^{dec}$.

$$\boldsymbol{h}_t^{dec} = \boldsymbol{h}_t^{int} \oplus \boldsymbol{h}_t^{bbox} \tag{3.7}$$

Where $\oplus$ is the concatenation operator. The joint representation, $\boldsymbol{h}_t^{dec}$, is used for the initial hidden state- and input of the recurrent decoder. Before using it in the input however, an additional soft-attention mechanism is applied. Moreover, the future ego-motion, $\boldsymbol{E} = \boldsymbol{e}_{t+1}, ..., \boldsymbol{e}_{t+T}$, is optionally concatenated with each GRU input. The calculation for the first decoder cell is provided in equation 3.8.

$$\boldsymbol{i}_t^{dec} = \left(\sigma_{sm}(\boldsymbol{h}_t^{dec}) \otimes \boldsymbol{h}_t^{dec}\right) \oplus \boldsymbol{e}_{t+1} \tag{3.8}$$

Where σ_{sm} is a fully-connected layer with a *softmax* activation function, $\otimes$ an element-wise multiplication, $\oplus$ the concatenation operation and $\boldsymbol{e}_{t+1}$ the future ego-motion at the time-step t+1. Note that the soft-attention mechanism, i.e. the addition of $\sigma_{sm}(\boldsymbol{h}_t^{dec})$ with an element-wise multiplication, is not present for all subsequent decoder cells.

For each future time-step $n \in [1, T]$ in the prediction horizon, the corresponding GRU cell outputs a hidden state $\boldsymbol{h}_{t+n}^{dec}$. This state is subsequently fed through a fully-connected layer with linear activation function, σ_l, which maps the high-dimensional hidden state to a four-dimensional bounding box prediction, $\boldsymbol{b}_{t+n}$.

$$\boldsymbol{b}_{t+n} = \sigma_l(\boldsymbol{h}_{t+n}^{dec}) \tag{3.9}$$

Where σ_l is a fully-connected layer with a linear activation function

3.6. Main contributions

Table 3.2 provides a tabular overview that puts the proposed method, IA-PTP, into the context of previous methods. IA-PTP is distinguished based on three main classifiers. Firstly, the *prediction task*, which discusses the task(s) the method performs. Secondly, the *interaction modeling module*, which discusses the interaction modeling method in terms of used *architecture*, used *stimuli* and *context types* with which interactions are modelled. Finally, *other modules*, which discusses other architectural modules used by the method to improve trajectory prediction performance. Note that all tabulated methods operate in the 2D on-board domain.

			Interaction modeling module			
#	**Method**	**Prediction task**	**Architecture**	**Context types**	**Stimuli**	**Other modules**
1	B-LSTM [6]	• trajectory	-	-	-	-
2	FOL-X [105]	• trajectory	-	-	-	• motion appearance encoder
3	PIE$_{traj}$ [70]	• trajectory	-	-	-	-
4	PIE$_{full}$ [70]	• trajectory • intention	-	-	-	• intention prediction
5	BiTraP-D [104]	• trajectory	-	-	-	• endpoint prediction
6	SGNet-D [98]	• trajectory	-	-	-	• step-wise endpoint prediction
7	H-LSTM [69]	• trajectory	-	-	-	• correlation • custom LSTM • intention prediction
8	PFT [62]	• trajectory	-	-	-	• ego-motion disentangling
9	TITAN [54]	• trajectory	• RNN-based	• dynamic • static	• agent actions • agent location	-
10	BiPed [72]	• trajectory • final location • crossing action	• attention-based	• dynamic • static	• class-wise semantic maps	-
11	STIP [50]	• crossing action	• graph-based	• dynamic • static	• instance masks • ped appearance	-
12	**IA-PTP**	**• trajectory**	**• graph-based**	**• dynamic • static**	**• instance masks • ped appearance • object class**	-

Table 3.2: Overview of the proposed (bold-faced) and existing methods in the 2D on-board domain. Each method is discussed in terms of the prediction task it performs, the interaction modeling module, and other type of modules. In turn, the interaction modeling module is discussed in terms of the used architecture, context types, and used stimuli.

As can be observed from table 3.2, there are no previous 2D on-board trajectory prediction methods that reason over interactions with the static- and dynamic context using a graph-based approach. One recent work in the 2D on-board domain, *STIP* [50], does perform this type of interaction modeling, but it performs pedestrian crossing action prediction. Therefore, this graph-based interaction modeling approach is the inspiration for this work to investigate a similar method in the trajectory prediction domain. Even though the interaction modeling module is inspired by *STIP*, there are some key differences. First, the object class has been added to the edge embedding in IA-PTP to differentiate between object classes explicitly. Second, *STIP* uses an additional node in each graph that contains an aggregation of all other context nodes, i.e. an aggregated context node. These aggregated context nodes are temporally connected with an additional GRU, similar to the pedestrian nodes. This aggregated context node and the corresponding GRU have been removed, as early experiments showed difficulties with generalisation. Third, the architecture surrounding the interaction modeling module, which includes the bounding box encoding stream and the decoder, is new.

The contributions of this work can be summarised as follows:

1. This work proposes the first pedestrian trajectory prediction method in the 2D on-board domain that incorporates interactions between the target pedestrian and its static- and dynamic environment using a graph-based approach.
   (a) It is shown that graph-based interaction modeling enables the proposed method to outperform all considered baselines for the *PIE* and *JAAD*$_{full}$ datasets on nearly all metrics.
   (b) It is shown that the performance gain on *JAAD*$_{full}$ is most significant for the close-by pedestrians by evaluating the model on the *JAAD*$_{beh}$ subset containing only such pedestrians.
2. This work shows that modeling interactions with all considered objects, i.e. crosswalks, traffic lights, and vehicles, improves trajectory prediction performance most compared to only modeling interactions with a subset of the aforementioned objects.

4

Experiments

This section describes the experimental setup adopted for the proposed approach. First, the baselines are discussed (section 4.1), after which the evaluation procedure is explained in terms of metrics (section 4.2) and datasets (section 4.3). Next, data pre-processing (section 4.4) and training setup (section 4.5) are discussed. Finally, the experimental results and discussion are elaborated on (section 4.6).

4.1. Overview of models and baselines

The proposed- and baseline methods are described below.

- **PIE$_{\text{traj}}$** [70]: Attentive single-stream LSTM encoder-decoder model which encodes past bounding box coordinates. The architecture is illustrated in figure 4.1. The method was implemented using the publicly available code with the same parameters settings as described in the paper.
  - **PIE$_{\text{full}}$**: extends PIE_{traj} by conditioning the predictions additionally on predicted ego-vehicle speed and pedestrian intention. These predictions are then concatenated with the embedded encoder hidden-state before using the soft-attention mechanism. to generate bounding box predictions.

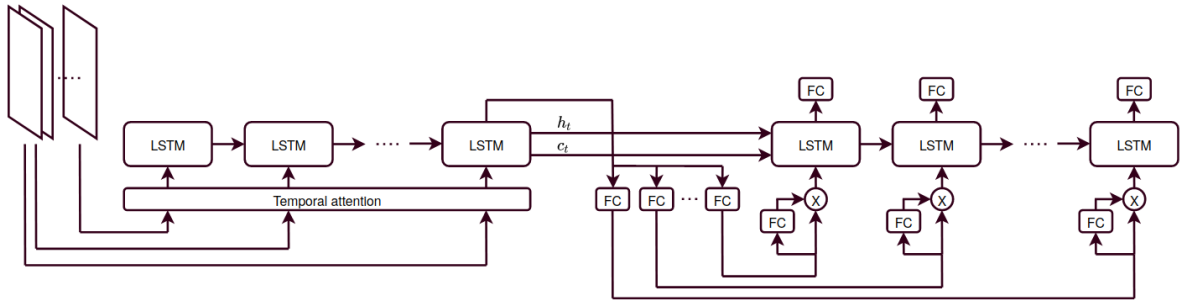


Figure 4.1: The PIE_{traj} baseline architecture. First, the bounding box sequence is fed through a temporal attention element weighing the relevance of each element in the input sequence. The weighted bounding box coordinates are used as input in the LSTM encoder to extract the most relevant temporal correlations. The resulting final cell state- and hidden state- are then used as initial states for the LSTM decoder. The input for each decoder LSTM cell is obtained by first embedding the final encoder hidden-state and subsequently performing soft-attention on this embedding.

- ***FOL-X*** **[105]**: multi-stream GRU encoder-decoder model. The first stream encodes past bounding box coordinates, and the second stream dense optical flow maps localized around the target agent. Both encodings are combined and fed into a GRU decoder. In parallel, a third stream feeds past ego-motion into a GRU encoder-decoder model to generate future ego-motion predictions, which are consequently used to condition the future predictions on.
- ***B-LSTM*** [6]: LSTM encoder-decoder model using a Bayesian formulation. The encoder encodes past bounding box locations, after which this is decoded to generate future bounding box predictions with an uncertainty measure.

- ***Linear Kalman Filter*** [70]: Linear Kalman Filter with a constant velocity motion model. This model is integrated as a recent publication [87] demonstrates the relevance of the constant velocity model for trajectory prediction, even in the current research space dominated by complex models.
- ***IA-PTP***: this work.

Looking at the current state of literature in the 2D on-board domain, other recent methods rise to the surface. However, these are not considered baselines for various reasons. Firstly, while formulating the thesis proposal, these methods were not yet published (*reason I*). Second, the methods are not considered a direct comparison to the proposed method because their main objective is to investigate the effect of other architectural components on trajectory prediction performance (*reason II*). However, integrating the proposed interaction modeling module in one of these methods could be an interesting future research direction. Lastly, the evaluation procedure is different, and no open-source implementation exists to evaluate with the same procedure as used in this work (*reason III*). The roman numbers behind each method below indicate what reason(s) led to excluding this method from the baselines.

- *BiTraP-D* [104] (I/II): goal-conditioned trajectory prediction method that combines a GRU encoder-decoder architecture with a goal prediction module. The goal prediction module predicts the end-point of the trajectory, which is consequently combined with a bi-directional trajectory decoder to improve the trajectory prediction. Both modules are trained jointly with a combined loss function.
- *SGNet-D* [98] (I/II): goal-conditioned trajectory prediction module that combines a GRU encoder-decoder architecture with a goal prediction module that predicts not only end-points like [104], but a goal for each future timestep. Subsequently, these goals are used to improve the trajectory prediction. Both modules are trained jointly with a combined loss function.
- *PFT* [62] (I/II): a method that explicitly disentangles the target pedestrian motion from the ego-motion. By first disentangling, the method uses a simple linear model to achieve very good performance. The objective of this method is to illustrate the importance of disentangling pedestrian motion from ego-motion.
- *BiPed* [72] (I/III): multi-modal model that, jointly- and separately, encodes past agent bounding boxes, image-plane grid locations and ego-motion using LSTMs. The interactions between the target agent and surrounding static- and dynamic context are modeled using a *Categorical Interaction Module* (see chap. 2.3) using class-wise semantic maps. The resulting joint- and separate encoding vectors are combined with the interaction information before decoding to obtain a multi-task prediction output containing future trajectory, crossing action and final location.
- *TITAN* [54] (III): multi-modal model that uses agent actions- and locations to reason over interactions between agents using a spatial GRU. Consequently, the resulting interaction information is jointly encoded with past target agent bounding boxes, agent actions and ego-motion using a GRU before decoding it into future bounding box predictions.

4.2. Metrics

The proposed method is quantitatively compared against the baselines based on metrics. These metrics should contain a measure of bounding box location, as well as a measure for bounding box size as a way to compensate for the missing depth dimension. Following previous work [105, 70, 6, 104, 62, 98], various metrics are adopted to evaluate the trajectory prediction in terms of bounding box size- and location.

1. ***MSE***: bounding box *Mean Squared Error* measures the mean squared error in pixels averaged over the four bounding box coordinates for each timestep, after which the value for each timestep is averaged across the entire prediction horizon. This metric is adopted for a prediction horizon of 0.5, 1.0 and 1.5 seconds.
2. $\boldsymbol{C_{MSE}}$: bounding box *Center Mean Squared Error* measures the mean squared error in pixels over the bounding box centre coordinates for each timestep, after which the value for each timestep is averaged across the entire prediction horizon. This metric is adopted for a prediction horizon of 1.5 seconds.
3. $\boldsymbol{C_{FMSE}}$: bounding box *Centre Final Mean Squared Error* measures the mean squared error in pixels over the bounding box centre coordinates at the last timestep of the prediction horizon. This metric is adopted for a prediction horizon of 1.5 seconds.

4.3. Datasets

The proposed method and baselines are evaluated on multiple datasets, each of which is discussed below.

1. **Joint Attention in Autonomous Driving (*JAAD*$_{full}$)**: The *Joint Attention in Autonomous Driving (JAAD)* dataset [71] contains 346 videos that have been recorded in 5 different cities in Europe and North America by an on-board camera with a resolution of 1920x1080 or 1280x720. The dataset annotations come in three different categories. First, pedestrian bounding boxes with tracking IDs, occlusion ratios and crossing/not-crossing actions. Second, pedestrian attributes, i.e. behavioural annotations, regarding its state before crossing, how the pedestrian becomes aware of an approaching vehicle and what action the pedestrian takes in response to the approaching vehicle. Third, scene attributes that list the environmental contextual elements. The dataset contains 2.8K pedestrians, out of which 686 contain behavioural annotations. *JAAD* is used in a variety of action/intention [50, 70, 71, 9, 72] and trajectory prediction papers [70, 72, 104, 98, 62, 69]. The dataset has no ego-motion information, but it does contain high-level driver actions, such as *speeding up* and *slowing down*.

   (a) ***JAAD* behavioural subset (*JAAD*$_{beh}$)**: the *JAAD*$_{beh}$ dataset is a subset of *JAAD*$_{full}$ and contains only the pedestrians that come with behavioural annotations, i.e. the far-away or non-interacting pedestrians are filtered out and only the ones that are close-by and potentially interacting with the ego-vehicle remain. This subset contains a total of 686 pedestrians.

2. **Pedestrian Intention Estimation (PIE)**: The *Pedestrian Intention Estimation (PIE)* dataset [70] has been recorded during daytime in 1 city in Canada with only sunny and overcast weather conditions using a single camera with a resolution of 1920x1080. Similar to *JAAD*, the dataset includes pedestrian bounding boxes with tracking IDs and occlusion ratios. Moreover, pedestrian attributes are included for actions and the intention of crossing. Finally, scene attributes and object annotations are present, including bounding boxes for pedestrian crossings, vehicles and stoplights. The dataset contains 1.8K pedestrians, which are all fully annotated. Contrary to *JAAD*, this dataset also comes with on-board diagnostics sensor data that provides GPS coordinates and vehicle information, such as velocity and heading angle. PIE is used in [70, 72, 93, 104, 98, 62, 69, 36].

4.4. Data pre-processing

The data requires to be pre-processed before use. First, pedestrian tracks of 2 seconds are sampled, containing 0.5s seconds observation and 1.5 seconds prediction. This equates to 15 frames observation and 45 frames prediction for both *JAAD* and *PIE*. Following baseline work [70], the tracks overlap by 80% for *JAAD* and 50% for PIE to generate more training data. Moreover, the bounding boxes are normalized by subtracting the first bounding box of a two-second track from all successive boxes, i.e. bounding box motion relative to the first bounding box is preserved. Next, the masks and pedestrian crops corresponding to each track are generated and fed through a pre-trained Instance Segmentation Network to generate binary object masks (see section 3.2). An illustration of the masks printed on a frame of the *PIE* dataset is shown in figure 4.2.

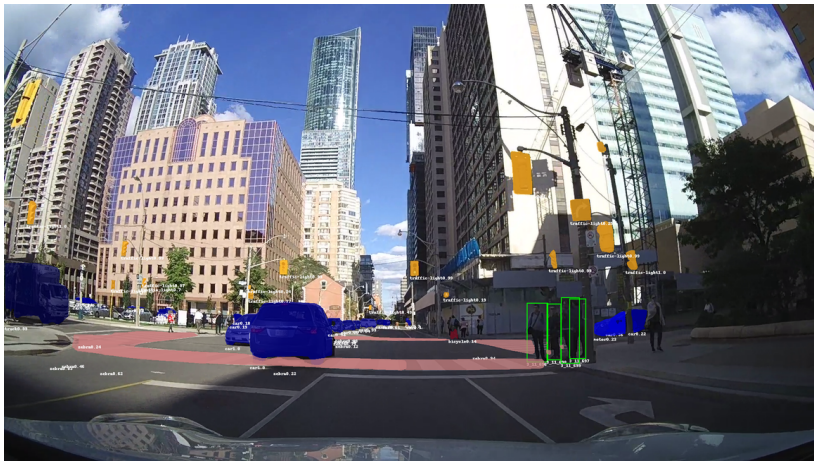


Figure 4.2: Visualisation of the predicted instance masks using Seamseg [68] pre-trained on Mapillary Vistas on a frame of the PIE dataset.

The union of the pedestrian bounding box and each object mask is cropped and encoded using a pre-trained *CNN* to obtain feature vectors. Additionally, the pedestrian appearance is encoded using the same pre-trained *CNN* (see section 3.3.1). The object- and pedestrian feature vectors are combined with the object bounding boxes, classes, and confidence scores and stored as pickle files to be extracted during data loading.

An overview of the total number of modelled interactions, as well as the average number of interactions per frame, is provided in table 4.1. Note that the number of interactions is not equal to the number of detected objects. Instead, each pedestrian, in each observation frame of each track, interacts once with each detected object in that frame. For example, assuming no track overlap, a single frame containing n pedestrians and m objects will result in $n \times m$ interactions, spread out evenly over n tracks. Moreover, as discussed in section 4.4, the observation horizon overlaps between tracks resulting in identical interactions in multiple tracks. A more detailed breakdown of the total number of interactions with each class is provided in figure A.1 in the appendix.

	PIE						**JAAD$_{full}$**					
	Train		**Testing**		**Train+Test**		**Train**		**Test**		**Train+Test**	
Object category	**Num masks**	**Avg per frame**	**Num masks**	**Avg per frame**	**Num masks**	**Avg per frame**	**Num masks**	**Avg per frame**	**Num masks**	**Avg per frame**	**Num masks**	**Avg per frame**
Crosswalks	4.5M	5.5	2.7M	5.0	7.2M	5.3	3.5M	3.5	2.6M	3.6	6.1M	3.5
Vehicles	17.7M	21.5	10.8M	19.9	28.5M	20.9	13.7M	13.4	8.8M	12.6	22.5M	13.1
Traffic light	11.0M	13.4	7.9M	14.5	18.9M	13.9	1.8M	1.8	0.9M	1.3	2.7M	1.6

Table 4.1: Number of interactions with each object category for train- and test-split of the *PIE* and *JAAD$_{full}$* datasets. Moreover, the average number of interactions per frame is provided within each data split. Note that the number of interactions is not equal to the total number of detected objects. Each pedestrian, in each observation frame of each track, interacts once with each detected object.

4.5. Training

The model is implemented in PyTorch [64] and has been trained on a single Tesla V100 16GB GPU. Across all datasets, the Adam [37] optimizer is used, combined with a plateau learning rate scheduler that reduces the learning rate by a factor of five after the validation loss has not improved for five consecutive epochs. Additionally, early-stopping is adopted to terminate training after the validation loss has not improved for ten epochs. *L2* regularisation is set to 10^{-4}, batch size to 32 and the maximum number of epochs to 80. The initial learning rate for *JAAD* is set to 7.5^{-5}. For *PIE*, the model's parameters are split into two parameter groups, where the initial learning rate of the graph module parameters are set to $2.5e^{-5}$ and the parameters of all other layers set to $7.5e^{-5}$. During training, the data is randomly shuffled. The hidden states of the bounding box-, interaction modeling- and decoder-GRU are set to respectively 256, 512 and 768 units. The graph node features are 512 dimensional, and the fully-connected layers for embedding the spatial relationship vector $\boldsymbol{s}^{o,i}$, the spatially-aware pedestrian appearance $\boldsymbol{v}^{ps,i}$ and the binary union mask $\boldsymbol{v}^{o,i}$ have respectively 64, 128 and 128 units.

4.6. Results and discussion

This section describes the experimental results, where each experiment is concluded with a discussion. First, the quantitative results are described and second, qualitative results that support the quantitative findings are presented.

4.6.1. Benchmark trajectory prediction performance using graph-based interactions

The performance of the proposed model is compared against the baselines using the benchmark evaluation procedure, i.e. evaluating on the $JAAD_{full}$ and *PIE* datasets using the metrics as described in section 4.2. The results are provided in table 4.2.

					PIE			$JAAD_{full}$		
	Stimuli				MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}
Method	**BB**	**Ego-motion**	**Intent**	**Other**	**0.5s/1.0s/1.5s**	**1.5s**	**1.5s**	**0.5s/1.0s/1.5s**	**1.5s**	**1.5s**
Linear [70]	✓	-	-	-	123/477/1365	950	3983	233/857/2303	1565	6111
B-LSTM [6]	✓	-	-	-	101/296/855	811	3259	159/539/1535	1447	5615
PIE_{traj} [70]	✓	-	-	-	58/200/636	596	2477	**110**/399/1248	1183	4780
FOL-X [105]	✓	yaw/speed	-	optical flow	**47**/183/584	546	2303	147/484/1374	1290	4924
PIE_{full} [70]	✓	speed	✓	-	-/-/559	520	2162	-/-/-	-	-
IA-PTP (this work)	✓	-	-	interactions	52/**177/550**	**513**	**2156**	**110/386/1205**	**1145**	**4653**

Table 4.2: Results for the baseline- and proposed methods on the $JAAD_{full}$ and PIE benchmark datasets. Every method contains a description of the used stimuli (for $JAAD_{full}$, no ego-motion is used). The best results for each dataset and metric are boldfaced.

The observations from table 4.2 are discussed for each dataset individually.

- ***PIE* dataset**: it can be observed that the baseline PIE_{traj} is outperformed by a large margin on all metrics for all prediction horizons (10.34/11.50/13.52% *MSE*, 13.93% C_{MSE} and 12.96% C_{FMSE}). The proposed method is only outperformed on short-term 0.5s predictions (9.62%) by FOL-X, which also uses ego-motion information. Additionally, the proposed method performs similarly to PIE_{full}, which uses predicted ego-vehicle speed and pedestrian intention.
- ***$JAAD_{full}$* dataset**: comparing the proposed method, IA-PTP, to the baseline PIE_{traj}, it can be observed that the short-term 0.5s predictions are similar in performance, whereas the longer 1.0/1.5s prediction horizon results are improved by 3.26/3.45% *MSE*, 4.90% C_{MSE} and 2.66% C_{FMSE}.

To gain a better understanding in which scenarios of $JAAD_{full}$ the performance gains are most significant, IA-PTP and PIE_{traj} are both trained on $JAAD_{full}$ and evaluated on the $JAAD_{beh}$ subset. The latter contains only close-by pedestrians that potentially interact with the ego-vehicle, for example by crossing, and are of special interest. The results are provided in table 4.3.

	Stimuli				MSE	C_{MSE}	C_{FMSE}
Method	**BB**	**Ego-motion**	**Intent**	**Other**	**0.5s/1.0s/1.5s**	**1.5s**	**1.5s**
PIE_{traj} [70]	✓	-	-	-	218/742/2304	2166	8735
IA-PTP	✓	-	-	interactions	**188/645/2054**	**1926**	**7992**

Table 4.3: Results for the baseline- and proposed method on $JAAD_{beh}$. Both methods are trained on $JAAD_{full}$.

Comparing IA-PTP to PIE_{traj} suddenly shows a much larger improvement across the whole range of prediction horizons, with performance gains of 13.76/13.07/10.85% *MSE*, 11.08% C_{MSE} and 8.51% C_{FMSE}.

Discussion From the benchmark results in table 4.2, we observe a clear improvement in trajectory prediction performance on all metrics across all datasets, except for the short-term 0.5s prediction when compared with FOL-X on the *PIE* dataset. This performance difference can likely be attributed to FOL-X using future ego-motion information, which our model in its current benchmark-setting does not use. Because

ego-motion can help disentangle the bounding box predictions from ego-motion induced image translation and rotation, integrating this cue is expected to further improve the proposed model's performance. This is investigated in section 4.6.1.1

Moreover, the model performs on-par with PIE_{full}, which additionally uses intention- and ego-motion information. The authors of PIE_{full} [70] also provide the results of their model without ego-motion, called $PIE_{traj+int}$, on the *PIE* dataset, with scores of 611 (1.5s *MSE*), 570 C_{MSE} and 2414 C_{FMSE}. Similar to our proposed method, the intention module of $PIE_{traj+int}$ uses pedestrian appearance information, although differently. $PIE_{traj+int}$ processes the appearance information to predict future pedestrian intention, which is consequently used to condition the trajectory predictions on. Our method on the other hand, uses the appearance jointly with contextual objects to reason over interactions with the environment. Our approach shows an average improvement of 11.4% over $PIE_{traj+int}$ on the aforementioned metrics. This suggests that modeling graph-based interactions could be more useful than using pedestrian intention in a way that the authors of $PIE_{traj+int}$ did.

Additionally, an interesting observation is that our proposed method outperforms the baseline on $JAAD_{beh}$ more significantly than on $JAAD_{full}$. The pedestrians contained in $JAAD_{beh}$ are all close-by or interacting with the ego-vehicle, thus generally more important to accurately predict, with regards to safety, than the far-away non-interacting pedestrians. This suggests that our proposed method improves most in places where it is most critical.

4.6.1.1 Oracle case with ego-motion

The benchmark performance has been evaluated without explicitly accounting for ego-motion. However, ego-motion plays an important role (see section 1.1.2), especially in the 2D on-board domain, where the trajectories are relative to the moving image plane. By explicitly integrating ego-motion, the model can, in theory, better compensate for this in its predictions. To assess the capabilities of the model when accounting for ego-motion, an oracle case is discussed where future ground truth ego-motion is used in the trajectory decoder. Two different types of ego-motion information are integrated. First, only ego-motion speed and second, ego-motion speed combined with yaw angle. Because *JAAD* does not contain ego-motion information, the experiment is performed only on *PIE*. The results are provided in table 4.4.

		Stimuli		**MSE**	$\mathbf{C_{MSE}}$	$\mathbf{C_{FMSE}}$
#	**Method**	**Ego-motion**	**Other**	**0.5s/1.0s/1.5s**	**1.5s**	**1.5s**
1	IA-PTP	-	-	52/177/550	513	2156
2	PIE_{full}	speed (GT)	intention (GT)	-/-/473	453	1741
3	IA-PTP	speed and yaw (GT)	interactions	51/**158**/456	422	1678
4	IA-PTP	speed (GT)	interactions	**49**/160/**453**	**419**	**1626**

Table 4.4: Oracle case with ground truth ego-motion on the *PIE* dataset

From table 4.4, it can be observed that the results of IA-PTP improve dramatically across all metrics, when feeding any type of ground truth ego-motion into the decoder. The variants of IA-PTP using either speed (row 3) or both yaw- and speed (row 4) perform similar, only showing a significant difference for the short-term 0.5s prediction (4.1%) and C_{FSME} (3.2%). Overall, the proposed model using only ground truth speed performs best, improving on its counterpart without ego-motion by 6.1/10.6/21.4% *MSE*, 22.4% C_{MSE} and 32.6% C_{FMSE}. Additionally, the proposed method outperforms the extended baseline PIE_{full}, using ground truth speed- and pedestrian intentions (row 2), by 4.4% 1.5*s MSE*, 8.1% C_{MSE} and 7.1% C_{FMSE}. Unfortunately, PIE_{full} with ground truth ego-motion and without ground truth intention is not available.

Discussion The oracle case experiment brings up two discussion points. First, the results of the oracle case show a significant improvement when ego-motion is added. This reinforces the knowledge that ego-motion is important in the 2D on-board trajectory prediction domain.

Second, it can be seen that using both yaw- and speed performs on-par with only using speed. This seems counter-intuitive, as excluding yaw would make it more difficult to compensate for camera yaw. A potential explanation is that the *PIE* test-set does not contain many scenarios where the ego-vehicle changes its yaw

angle, i.e. turns. To verify this, the maximum change in yaw angle within each track of the *PIE* testset is extracted and shown as a histogram in figure 4.3.

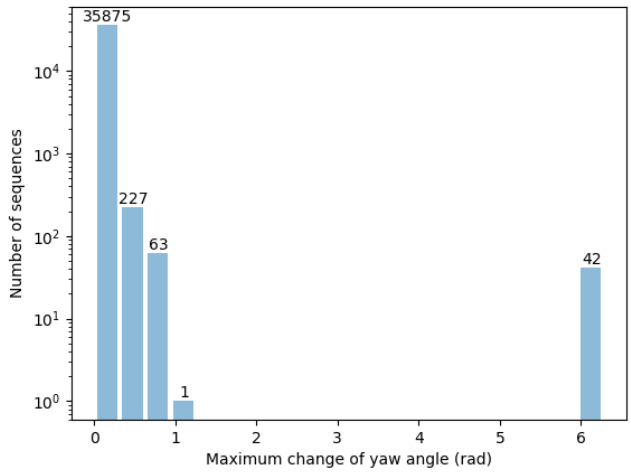


Figure 4.3: Histogram showing the distribution of maximum change of yaw angle in radians within a sequence for the *PIE* test-set. The horizontal axis shows maximum change in yaw angle with twenty equally distributed bins. The vertical axis shows the number of sequences contained within each bin on a log-scale.

As can be observed in the histogram, 35875 sequences have a maximum change in yaw angle corresponding to the first bin, i.e. not exceeding 0.314 rad. This equates to 99.1% of all sequences and explains the negligible effect of integrating yaw. We note that on other datasets, in which yaw angle plays a more significant role, the explicit integration of speed- and yaw would likely improve results over the case where only speed is used.

4.6.2. Contribution of various contextual objects to trajectory prediction performance

The experiments discussed below analyze the contribution of various contextual objects on trajectory prediction performance.

4.6.2.1 Analysis of different sets of contextual objects

The previous section showed that the proposed method outperforms the baselines on the benchmark experiments. These experiments were performed while taking into account interactions with all contextual object types mentioned in table 3.1, i.e. the contextual objects of interest. This ablation study investigates the contribution of different contextual objects to model performance. Four different sets of contextual objects are investigated:

1. Set with all context objects
2. Set with all contextual objects excluding the *plain- and zebra- pedestrian crosswalks*.
3. Set with all contextual objects excluding all objects belonging to the category *vehicles*, i.e. excluding bicycle, bus, car, motorcycle, trailer and truck.
4. Set with all contextual objects excluding the *traffic lights*.

The proposed IA-PTP method is trained- and tested using each of the four sets of contextual objects on *PIE* and $JAAD_{full}$. Moreover, the model trained on $JAAD_{full}$ is additionally evaluated on $JAAD_{beh}$. The results are provided in table 4.5.

				PIE			$JAAD_{full}$			$JAAD_{beh}$		
	Contextual object			MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}
#	Vehicles	Crosswalks	Traffic lights	0.5s/1.0s/1.5s	1.5s	1.5s	0.5s/1.0s/1.5s	1.5s	1.5s	0.5s/1.0s/1.5s	1.5s	1.5s
1	✓	✓	✓	**52/177/550**	**513**	**2156**	**110/386/1205**	**1125**	**4653**	**188/645/2054**	**1926**	**7992**
2	✗	✓	✓	57/202/668	627	2679	114/428/1356	1295	5243	198/725/2347	2214	9159
3	✓	✗	✓	**52**/187/612	573	2442	113/413/1284	1223	4932	193/700/2231	2100	8640
4	✓	✓	✗	85/235/658	619	2460	116/451/1412	1353	5394	200/760/2402	2276	9204

Table 4.5: Results of IA-PTP which has been trained with different sets of contextual objects. ✓ and ✗ indicate which contextual objects are respectively used and not used. The best results are boldfaced.

The observations from table 4.5 are discussed for each dataset individually. Note that the performance differences are expressed in percentage relative to the full model (row 1).

- ***PIE* dataset**: in general, it can be observed that removing any type of contextual object results in deteriorating performance across nearly all metrics. More specifically, removing the *vehicles* (row two) leads to performance loss on all metrics of 9.6/14.1/21.5% *MSE*, 22.2% C_{MSE} and 24.3% C_{FMSE}. Removing the pedestrian crosswalks (row three) also leads to performance loss on all metrics with the exception of the short-term 0.5s prediction, with losses equating to 5.7/11.3% *MSE*, 11.7% C_{MSE} and 13.3% C_{FMSE}. Finally, the case without *traffic lights* (row four) sees short-term predictions suffering most, with losses of 63.5/32.8/19.6% *MSE*, 20.7% C_{MSE} and 14.1% C_{FMSE}. Overall, long-term 1.5s predictions suffer most when the vehicles are removed, whereas short-term 0.5/1.0s prediction suffers most when removing traffic lights.

- ***$JAAD_{full}$* and *$JAAD_{beh}$* datasets**: similar to the *PIE* evaluation, performance loss is observed for removing any type of contextual objects. More specifically, removing the *vehicles* reduces performance with 3.6/10.9/12.5% *MSE*, 15.1% C_{MSE} and 12.7% C_{FMSE} on $JAAD_{full}$, whereas $JAAD_{beh}$ shows slightly worse performance, with losses of 5.3/12.4/14.3% *MSE*, 15.0% C_{MSE} and 14.6% C_{FMSE}. Training without *crosswalks* leads to a performance loss of 2.7/7/6.6% *MSE*, 8.7% C_{MSE} and 6.0% C_{FMSE} on the $JAAD_{full}$ dataset and a again slightly worse performance on $JAAD_{beh}$, showing losses of 2.7/8.5/8.6% *MSE*, 9.0% C_{MSE} and 8.1% C_{FMSE}. Finally, the case without *traffic lights* sees a drop in performance of 5.5/16.8/17.2% *MSE*, 20.3% C_{MSE} and 15.9% on the $JAAD_{full}$ dataset and 6.4/17.8/16.9% *MSE*, 18.2% C_{MSE} and 15.2% on the $JAAD_{beh}$ dataset. Overall, removing traffic lights leads to the greatest performance loss for both *JAAD* datasets, especially for the 1.0s/1.5s metrics.

Discussion This experiment brings forward three discussion points. First, we observe a performance loss when excluding any contextual object compared to the model trained on the set with all context objects. This suggests that all contextual objects contribute in some meaningful way to improve overall trajectory performance.

Second, re-training the model excluding *crosswalks* results in the most negligible performance loss on all metrics for all datasets. This implies that the pedestrian crossings are least important to the overall trajectory prediction performance in their currently integrated way. However, this does not necessarily always have to be the case. For example, it could also just be that the cues typically obtained from crosswalks can be captured quite well in some way by the other objects, for example, the presence of a traffic light could mean that there will also be a crosswalk. Therefore, an interesting direction for future work would be to try out other sets of contextual objects to further investigate dependencies between objects.

Third, this analysis shows that the traffic lights are relatively important, whereas crosswalks are not. This is especially interesting on the *JAAD* datasets, where the average number of traffic lights per frame, provided in table 4.1, is relatively low. This could suggest that the presence of a relatively rare traffic light would have a relatively large impact on the trajectory prediction.

4.6.2.2 Sensitivity analysis

This ablation study discusses the model's sensitivity to scenarios with limited presence of the contextual objects the model was trained on. Therefore, IA-PTP is trained on the set of all contextual objects and subsequently evaluated on each of the three sets of contextual objects that exclude one type of object category. In addition to this, the results will be compared to the respective case from the previous ablation study, where the model has not seen the removed objects during training either. All results are provided in table 4.6. Row one shows the model trained- and tested using the set of all contextual objects, i.e. the *full model*. Rows two, four, and six show the sensitivity analysis results. Rows three, five and seven have been copied from table 4.5 of the previous ablation study and have been added to easily compare these with the sensitivity analysis results, as part of this ablation study.

				PIE			$JAAD_{full}$			$JAAD_{beh}$		
	Contextual objects (train/test)			MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}
#	Vehicles	Crosswalks	Traffic lights	0.5s/1.0s/1.5s	1.5s	1.5s	0.5s/1.0s/1.5s	1.5s	1.5s	0.5s/1.0s/1.5s	1.5s	1.5s
1	✓/✓	✓/✓	✓/✓	**52/177/550**	**513**	**2156**	**110/386/1205**	**1125**	**4653**	**188/645/2054**	**1926**	**7992**
2	✓/✗	✓/✓	✓/✓	120/350/886	843	3091	425/1155/2568	2499	7948	846/2129/4554	4411	13797
3	✗/✗	✓/✓	✓/✓	57/202/668	627	2679	114/428/1356	1295	5243	198/725/2347	2214	9159
4	✓/✓	✓/✗	✓/✓	53/184/574	537	2239	151/464/1350	1289	5041	286/831/2400	2271	8925
5	✓/✓	✗/✗	✓/✓	**52**/187/612	573	2442	113/413/1284	1223	4932	193/700/2231	2100	8640
6	✓/✓	✓/✓	✓/✗	53/188/603	565	2390	156/469/1323	1263	4882	301/839/2310	2181	8418
7	✓/✓	✓/✓	✗/✗	85/235/658	619	2460	116/451/1412	1353	5394	200/760/2402	2276	9204

Table 4.6: Sensitivity analysis of proposed IA-PTP method by training and/or testing using different sets of contextual objects. Which contextual objects are used for training and/or testing is indicated with ✓ or ✗. In the case $JAAD_{beh}$, the model was trained on $JAAD_{full}$ and tested on $JAAD_{beh}$

First, the focus is on the sensitivity analysis, as shown in rows two, four and six of table 4.6. The performance differences are expressed in percentage relative to the proposed model.

- ***PIE* dataset**: the performance deteriorates in all cases when removing any type of contextual object compared to the full model. The performance drop is most significant for all metrics when removing the *vehicles* (row two) during test-time (130.8/97.7/61.1% *MSE*, 64.3% C_{MSE} and 43.4% C_{FMSE}). The experiment without *traffic lights* (row six) results in a less severe performance decrease on long-term 1.5s metrics (6.2% *MSE*, 10.1% C_{MSE} and 10.9% C_{FMSE}), whereas short-term 0.5/1.0s prediction horizons show only minor losses compared to the full model (3.8/1.9 *MSE*). Similarly, removing the *pedestrian crosswalks* (row four) also shows minor losses for the short-term 0.5/1.0s prediction horizons (1.9/4.0 *MSE*). Furthermore, excluding crosswalks shows relatively the smallest performance loss on the long-term 1.5s metrics (4.4% *MSE*, 4.7% C_{MSE} and 3.8% C_{FMSE}).

- ***$JAAD_{full}$* and *$JAAD_{beh}$* datasets**: similar to the evaluation on *PIE*, performance deteriorates in all cases when removing any type of contextual object. On $JAAD_{full}$, excluding the *vehicles* reduces performance most significantly for all metrics (286.4/199.2/113.1%T, 122.1% C_{MSE} and 70.8% C_{FMSE}). This is also the case for $JAAD_{beh}$, where the losses are even more pronounced (350.0/230.1/121.7% *MSE*, 129.0% C_{MSE} and 72.6% C_{FMSE}). The experiment excluding *traffic lights* mainly sees a drop in performance on $JAAD_{full}$ for short-term 0.5/1.0s predictions, and less so for the long-term 1.5s metrics (44.3/21.1/9.8% *MSE*, 12.3% C_{MSE} and 4.9% C_{FMSE}). The same pattern is visible for excluding traffic lights on $JAAD_{beh}$, but the losses are slightly more pronounced (60.1/30.1/12.5% *MSE*, 13.2% C_{MSE} and 5.3%). Finally, excluding *crosswalks* leads to a performance loss on $JAAD_{full}$ that is, on one hand, less severe for the short-term 0.5/1.0s predictions (37.3/20.2 *MSE*) compared to excluding traffic lights and, on the other hand, more severe for the long-term 1.5s predictions (12.0% *MSE*, 14.6% C_{MSE} and 8.3% C_{FMSE}). Again, excluding crosswalks on $JAAD_{beh}$ shows a similar pattern as excluding them on $JAAD_{full}$, but with more significant losses (52.1/28.8/16.8% *MSE*, 17.9% C_{MSE} and 11.7% C_{FMSE}).

The focus is shifted toward comparing the sensitivity analysis result in rows two, four, and six in table 4.5 with the results of table 4.6. As mentioned before, to ease comparison, the results of table 4.6 have been added to rows three, five and seven of table 4.5.

- ***PIE* dataset**: It can be observed that the model not using *vehicles* during both training- and test-time outperforms its counterpart, where vehicles are only excluded during test time, on all metrics (110.5/73.3/32.6% *MSE*, 34.4% C_{MSE} and 15.4% C_{FMSE}). On the contrary, the model that excludes *crosswalks* during both training- and testing, performs worse on the 1.5s metrics than its counterpart (6.6% *MSE*, 6.7% C_{MSE} and 9.1% C_{FMSE}). The performance on the 0.5/1.0s metrics is similar. Finally, comparing *traffic lights* exclusion during both training- and testing with its counterpart shows similar behavior on the 1.5s metrics as for the crosswalks case (9.1% *MSE*, 9.6% C_{MSE} and 2.9% C_{FMSE}). Differently though, it performs much worse for the short-term 0.5s metrics (60.4/25.0% *MSE*)
- ***JAAD*$_{full}$ and *JAAD*$_{beh}$ datasets**: the model that excludes *crosswalks* or *vehicles* during both training- and test-time outperforms its counterpart, that only excludes the context category during test-time, on all metrics. The difference for the crosswalk case is 33.6/12.3/5.1% *MSE*, 5.4% C_{MSE} and 2.2% C_{FMSE}, whereas the difference for the vehicles case is 272.8/169.9/89.4% *MSE*, 93.0% C_{MSE} and 51.6% C_{FMSE}. This behavior is similar to the no-vehicles case on the *PIE* dataset. Removing *traffic lights* also shows similar behaviour for short-term 0.5/1.0s predictions (37.1/4.0% *MSE*), as was seen for *PIE*. On the contrary, the model excluding *traffic lights* only during test-time outperforms the model also trained without these contextual objects on the long-term 1.5s metrics (6.7% *MSE*, 7.1% C_{MSE} and 10.5% C_{FMSE}). The performance differences on *JAAD*$_{beh}$ show similar behavior, but generally slightly more exaggerated.

Discussion The sensitivity analysis brings forward two discussion points. First, we look at the model's performance where one object category (i.e. vehicles, crosswalks or traffic lights) has been excluded only during test time. Based on the results in table 4.6, the model seems most sensitive to scenes without vehicles for both datasets. For *PIE*, the model seems least sensitive to scenes without crosswalks. For *JAAD*, the model is least sensitive to scenes without crosswalks for short-term predictions, whereas removing traffic lights has the least influence on the long-term predictions. It is hard to draw real conclusions from this, as the performance loss- or gain from removing certain contextual objects might arise due to other reasons. For example, the performance loss could be related to the number of object interactions that is removed during test time, where a larger number has a more significant impact. This could have an impact because we do not employ normalization in the *GCN* based on the number of interactions. Therefore, we turn to the number of object interactions in the test-set for both *JAAD* and *PIE*, which can be retrieved from figure 4.4a. For *PIE*, we observe that the majority of interactions is with vehicles (10.8M), followed by traffic lights (7.9M), and the minority of interactions is with crosswalks (2.7M). For *JAAD*, we see that the majority of interactions are with vehicles (8.8M), followed by crosswalks (2.6M) and, finally, traffic lights (0.9M). We see a pattern that the performance for long-term predictions suffers more when more interactions are removed during test time. This could suggest that the number of interactions with an object category potentially plays a role in the model's sensitivity regarding that specific object category.

Second, the general drop in performance when removing certain object categories during test-time only, is the expected behaviour. The model learns to use each type of object category in some way during training. By suddenly removing all objects of a certain category, the model loses certain cues. Cues from other objects can likely not compensate for the missing cues, as they did not need to compensate for the missing cues during training either. That brings us to the counter-intuitive result, namely that in the case of removing crosswalks and traffic lights from *PIE*, as well as when removing traffic lights from *JAAD*, the performance is worse compared to when these objects were already excluded during training. A possible explanation for this could be that the masks for these classes in the training set are relatively noisy. Noisy masks include partly- or completely erroneous detected instances, or multiple detected instances of one object. These noisy masks make the graph input noisier, which impairs the learning process resulting in a worse model. Moreover, it is complicated for the model to selectively ignore noisy classes due to the current implementation of the GCN. The *GCN* uses a single weight matrix to combine the feature vectors of all objects with that of the pedestrian. In other words, it aims to generalize over all object classes, which becomes increasingly complex in case there are many noisy masks.

4.6.2.3 Analysis of learned edge weights

To evaluate the capabilities of the model to learn useful edge weights, a comparison is made between the proposed method with learnable- and non-learnable edge weights. In the latter case, each edge weight is set

to $1/n$, where n equals the number of objects. The models are trained from scratch on $JAAD_{full}$ and *PIE*, after which they are evaluated. The results are shown in table 4.7.

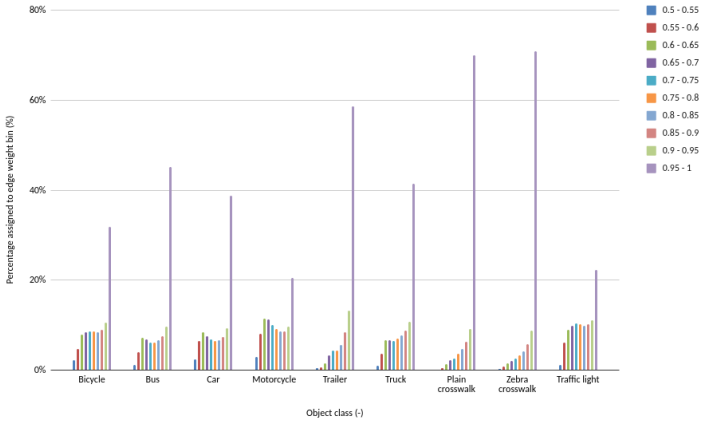
		PIE			$JAAD_{full}$			$JAAD_{beh}$		
		MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}	MSE	C_{MSE}	C_{FMSE}
#	Ablation	0.5s/1.0s/1.5s	1.5s	1.5s	0.5s/1.0s/1.5s	1.5s	1.5s	0.5s/1.0s/1.5s	1.5s	1.5s
1	Learned edge weights	**52/177/550**	**513**	**2156**	**110/386/1205**	**1125**	**4653**	**188/645/2054**	**1926**	**7992**
2	Uniform edge weights	57/208/701	659	2846	114/441/1374	1315	5224	193/731/2321	2195	8916

Table 4.7: Results of using learned edge weights versus uniform edge weights on the *PIE*, $JAAD_{full}$ and $JAAD_{beh}$ datasets

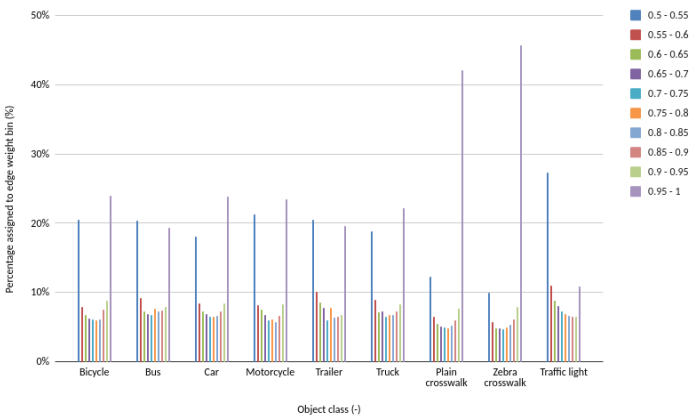
The results are discussed for all datasets individually.

- ***PIE* dataset**: Replacing the learnable edge weights with uniform edge weights results in a loss on all metrics of 9.6/17.5/27.4% *MSE*, 28.5% C_{MSE} and 32% C_{FMSE}.
- ***$JAAD_{full}$* and *$JAAD_{beh}$* datasets**: Replacing the learnable edge weights with uniform edge weights results in a loss on all metrics of 3.6/14.2/14% *MSE*, 16.9% C_{MSE} and 12.3% C_{FMSE} on $JAAD_{full}$ and 2.7/13.3/13% *MSE*, 14% C_{MSE} and 11.6% C_{FMSE} on $JAAD_{beh}$.

Next, a quantitative analysis of the assigned edge weights during test-time for the best model on the *PIE* and $JAAD_{full}$ datasets is performed. The results are shown in figure 4.4, which shows a bar-plot for each object class for both datasets. Each bar colour represents an edge weight bin of size 0.05. Since the edge weights are $[0.5, 1]$, this bin size results in a total of ten bins. The vertical axis shows the percentage of the total number of interactions with objects of class c that have been assigned to a particular bin. Each set of ten bars sums to 100%.



(a) Edge weight distribution for *PIE*



(b) Edge weight distribution for $JAAD_{full}$

Figure 4.4: Overview illustrating the percentage of total interactions assigned to each edge weight bin per object class for the *PIE* and $JAAD_{full}$ datasets. Each class, shown on the horizontal axis, contains ten bars, corresponding to the ten edge weight bins. For each of these ten bars, the left-most bar corresponds to the lowest edge weight bin (0.5-0.55) and the right-most bar corresponds to the highest edge weight bin (0.95-1.00). Each set of ten bars sums to 100%

First, the edge weight distribution for *PIE* is analyzed. From figure 4.4a, it can be observed that there is a strong bias toward edge weight bin 0.95-1. Moreover, the percentage of objects assigned to the highest weight bin is highest for both crosswalk classes, followed by the trailer class. Also, the contribution of the lowest weight bins is, in all cases, very minor.

Next, the edge weight distribution on *JAAD* is investigated. From figure 4.4b, it can be observed that the weights are distributed more equally across the bins, with peaks for both the lowest- and highest edge weight bins across all object classes. The peaks for the highest edge weight bins are relatively high for both crosswalk classes and relatively low for the traffic light class.

Discussion The edge weight investigation brings up three discussion points. First, table 4.7 suggests that the learned edge weights provide a substantial benefit over uniform edge weights on all metrics for all datasets.

Secondly, figure 4.4a indicates a strong bias towards high edge weights for the *PIE* dataset. This begs the question whether the edge weights are being optimally used, as it seems unlikely that such a large percentage of all interactions should contribute equally. An interesting future research direction would be to craft a different edge weight equation that can better learn to differentiate between more- and less important weights on *PIE*.

On the other hand, the edge weight distribution for *JAAD* is more like we would expect, with the majority of edge weights in the lowest- and highest bin. This suggests that the model is better able to differentiate between more- and less important objects on *JAAD*, compared to *PIE*.

The third discussion point is that the edge weights cannot get below 0.5. This is attributed to the combination of *Sigmoid* and *ReLU* activation functions, as shown in equation 3.5. This property seems sub-optimal, as it does not allow the method to completely ignore certain contextual objects by assigning them a weight of 0. This property would allow the model to completely ignore certain masks. This could help the model train more effectively, for example, by completely ignoring noisy masks.

4.7. Qualitative analysis

A qualitative analysis is performed to gain more intuitive knowledge regarding the graph-based interactions and how they relate to the predicted trajectories. Even though qualitative results can help understand what a model is doing for specific cases, one must tread carefully not to over-interpret the results.

4.7.1. Visualization protocol

Each visualization, shown in figures 4.5-4.7, contains two frames of one track. The left image is the last frame of the observation horizon, i.e. timestep t, and the right image is the last frame of the prediction horizon, i.e. timestep $t+45$ (1.5s). The following content is printed on the observation frame:

- The ground truth bounding box in green, i.e. *GT*.
- The predicted instance masks in various colours. Each mask colour corresponds to a bin of edge weights, i.e. the edge weights for all objects are divided over three bins (0.5-0.7, $0.7-0.9$ and $0.9-1.0$) depending on the learned weight value.
- A dotted line is printed from the pedestrian to the centre of the most important object masks, i.e. to all objects with an edge weight in the highest bin (0.9-1.0).

The following content is printed on the prediction frame:

- Ground truth bounding box in green, i.e. *GT*.
- Bounding box prediction of proposed IA-PTP method trained and tested on the set of all contextual objects in black, i.e. *graph*.
- Bounding box prediction of proposed IA-PTP method trained on the set of all contextual objects and tested on the set without plain- and zebra pedestrian crossings in red, i.e. *graph_no_crossings*. This has been added, as it provides an interesting qualitative insight.
- Bounding box prediction of baseline PIE$_{traj}$ in blue, i.e. *bbox*.

Discussion The first qualitative result is illustrated in figure 4.5. It shows a pedestrian motion change from an almost stand-still to crossing the street over a pedestrian crossing right in front of the ego-vehicle. The pedestrian crossing in front of the ego-vehicle has been correctly detected and is part of the most important objects. Looking at the predictions, *graph* outperforms *graph_no_crossings*, which in turn outperforms *bbox*. This demonstrates the expected behaviour in this scenario. The improvement of both graph models over *bbox* could be explained because they additionally use appearance features. The improvement of *graph* over *graph_no_crossings* could be explained due to the detected and heavily weighed pedestrian crossing in front of the ego-vehicle.

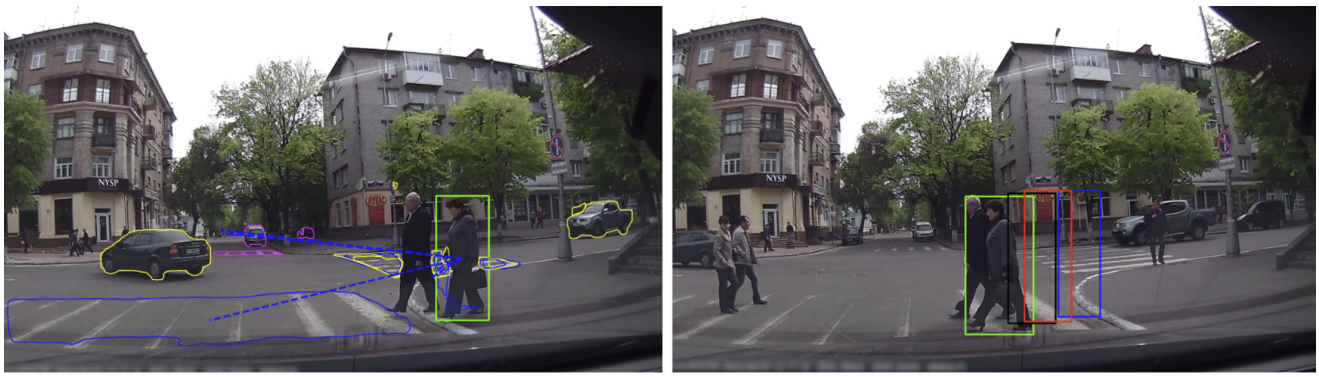


Figure 4.5: Visualization of a track on *JAAD* showing the last frame of the observation horizon (left) and the last frame of 1.5s prediction horizon (right). The track contains a pedestrian motion change from an almost standstill into crossing. The motion change occurs close to the final observation frame. Poor crossing prediction by *bbox* (blue) model, better prediction by *graph_no_crossings* (red) and best prediction by *graph* (black). The most important objects include three vehicles (31, two zebra- (8) and one unmarked- (3) crosswalk, which are outlined in blue on the observation image.

The second qualitative result is illustrated in figure 4.6. It shows a pedestrian that just crossed the street over a pedestrian crossing on the right side of the intersection and continues walking on the sidewalk. The crosswalk in front of the ego-vehicle and the crosswalk on the right side have been detected and assigned high importance. Moreover, the image also contains several noisy and erroneous crosswalk predictions, among which the crosswalks in the middle of the street and on the sidewalk. Looking at the predictions, it can be observed that both the *bbox* model and *graph* model make a prediction that steers the pedestrian towards his right, i.e. towards the crosswalk in front of the ego-vehicle. The *graph_no_crossings* model, however, makes a correct prediction following the pedestrian along the sidewalk. First, the use of appearance information in *graph_no_crossings* could explain why it outperforms *bbox*, because the orientation of the pedestrian matches the direction of his future trajectory. Second, *graph_no_crossings* outperforming *graph* could be explained by *graph_no_crossings* not using the high edge-weight crosswalks in front of the ego-vehicle, contrary to *graph*.

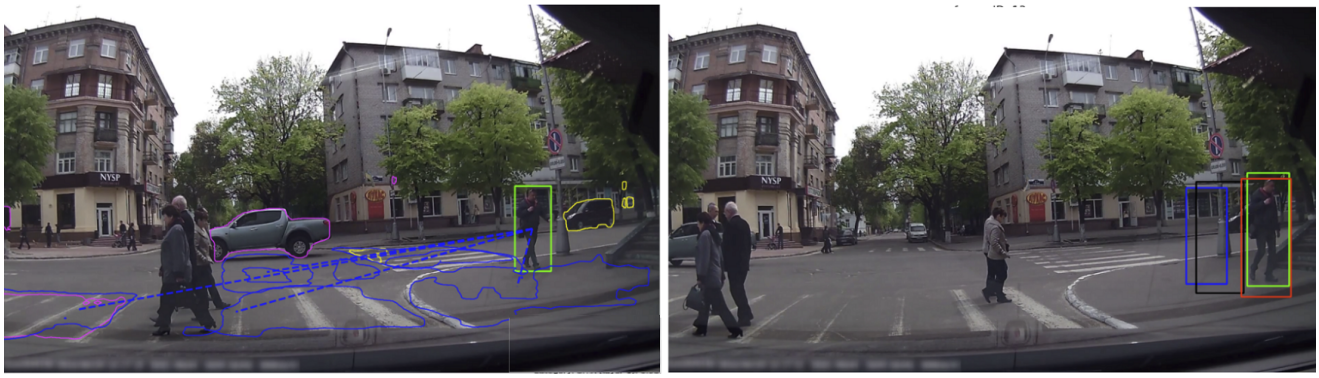


Figure 4.6: Visualization of a track on *JAAD* showing the last frame of the observation horizon (left) and the last frame of 1.5s prediction horizon (right). The track contains a pedestrian that just crossed the crosswalk on the right side and continues walking along the sidewalk, instead of crossing the road in front of the ego-vehicle. Poor crossing prediction by *bbox* (blue), better prediction by *graph* (black) and best prediction by *graph_no_crossings* (red). The most important objects include four zebra- (8) and one unmarked (3) crosswalk, which are outlined in blue on the observation image.

The third qualitative result is illustrated in figure 4.7. It shows a pedestrian motion change from a stand-still to crossing the street over a poorly marked pedestrian crossing right in front of the ego-vehicle. Looking at the predictions, it can be observed that *graph* performs worst by staying in roughly the same location. *Graph_no_crossings* captures the motion change much better, performing best on this example. This could be explained by the fact that the pedestrian crosswalk, that the pedestrian intends to cross has not been correctly detected. Instead, a noisy and erroneous mask has been detected on the left side of the pedestrian, which additionally has been assigned the highest importance. Thus, *graph* is distracted and steers the prediction towards the left, i.e. wrong, direction. This is contrary to *graph_no_crossings*, which does not see the erroneously detected pedestrian crossing on the left side and makes a prediction in the correct direction.

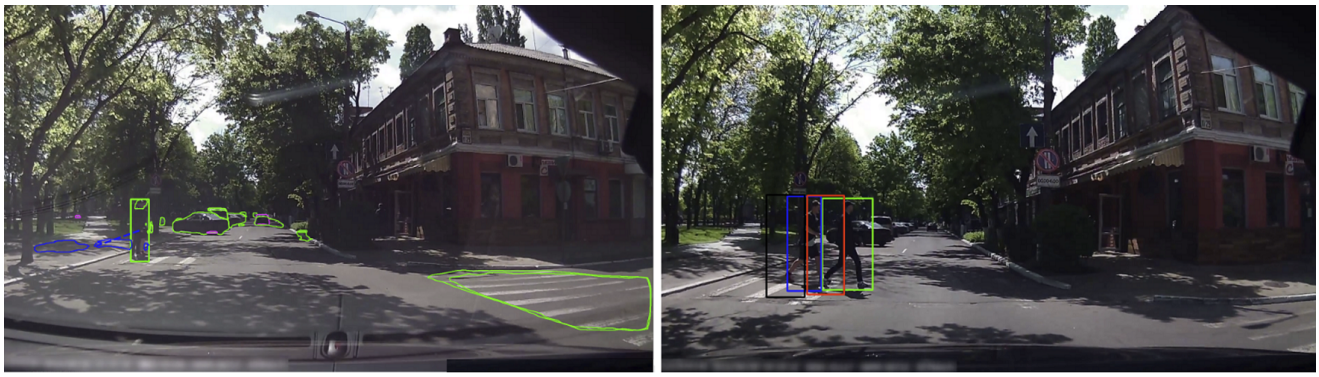


Figure 4.7: Visualization of a track on *JAAD* showing the last frame of the observation horizon (left) and the last frame of 1.5s prediction horizon (right). The track contains a pedestrian motion change from stand-still to crossing a pedestrian crosswalk (which has not been detected correctly). Motion change is poorly predicted by *graph* (black), slightly better by *bbox* (blue) and well captured by *graph_no_crossings* (red)

5

Conclusion

5.1. Summary

This work presented the first 2D on-board pedestrian trajectory prediction method performing graph-based interaction modeling with static- and dynamic contextual objects. As discussed in section 1.3, the objective of this research was to answer two research questions.

- *Can modeling the interactions between a pedestrian and its static- and dynamic context using a graph-based approach benefit the performance of a 2D trajectory prediction framework?*
- *Which interactions with dynamic- or static context provide the most significant contribution to the 2D on-board pedestrian trajectory predictions?*

The proposed method works by first extracting the target pedestrian and other semantic entities from the observation data. The extracted information is then processed using two separate encoding streams. The first stream encodes the motion history of the target pedestrian using a Gated Recurrent Unit (GRU). The second stream encodes the interactions between the target pedestrian and its static- and dynamic environment. These interactions are modelled by connecting the various semantic entities in each time-step using a spatial graph and subsequently reasoning over each graph using Graph Convolution Networks (GCN). The graphs are connected temporally using a GRU to account for temporal correlations between the spatial interactions. The embeddings resulting from both encoding streams are fused to form a joint representation, which serves as initial hidden state- and input for the GRU decoder. For the decoder input specifically, the representation is first updated using a soft-attention mechanism. Each decoder cell output is linearly mapped using a fully connected layer to obtain a bounding box prediction. Optionally, the future ego-motion is fused with the input of each decoder cell.

Our experiments in section 4.6.1 have shown that using graph-based interaction modeling with the static- and dynamic context improves trajectory prediction performance, outperforming the baseline method on all metrics by 10-14% for *PIE*, as well as on $JAAD_{full}$ for the longer (>1s) prediction horizons between 3-4.9%. In addition, by evaluating the model, that was trained on $JAAD_{full}$, on the subset $JAAD_{beh}$, we saw an improvement of between 8.5-13.8% on all metrics. This is considered evidence of the potential for modeling interactions with the static- and dynamic context using our graph-based approach in the 2D on-board trajectory prediction domain.

Our ablation studies in section 4.6.2 have indicated that it is relevant to model interactions with all of the contextual objects proposed in this work, more specifically, vehicles, crosswalks and traffic lights. Removing any of these contextual objects leads to reduced model performance. This is the case for removing objects both during train- and test-time, as well as only during test-time. However, it is difficult to draw a definitive conclusion regarding which contextual objects have the most significant influence. We observed inconsistent results with regards to performance loss when removing certain contextual objects depending on the dataset. This could have various reasons; firstly, due to noisy instance masks that impair the learning process. Secondly, because there is a potential relation between the number of masks of a certain object class and the impact of this object class on performance. Thirdly, because the graph edge weights are calculated in a way that noisy masks and irrelevant objects cannot be ignored completely, and finally, because a *GCN*

approach is adopted that tries to generalize over all objects classes. Future work in this field is required to better determine which contextual objects contribute most.

5.2. Limitations

The proposed method has several limitations. First of all, the model is computationally expensive, even without the pre-processing steps included. Other recent approaches propose a computationally less demanding method for improving trajectory prediction performance, for example, the goal-conditioned approaches *SGNet* [98] and *BiTraP* [104].

Second, the model is complex to train, likely because of the noisy graph input. The model required monitoring of the gradient magnitudes to see if the graph was being used. In the case of *JAAD*, any initial learning rate greater than $1e^{-4}$ renders the graph module obsolete, whereas for *PIE* this upper bound was equal to $5e^{-5}$. The training process will likely improve by reducing noise, for example by using a confidence threshold on the instance predictions or by merging multiple instance predictions of the same object.

Third, the edge weight lower bound is equal to 0.5 due to the combination of the *Sigmoid* and *ReLU* activation functions, as shown in equation 3.5. This seems suboptimal, as this does not allow the method to ignore certain contextual objects completely. The ability to learn weights equal to 0 would be beneficial in selectively ignoring noisy masks and possibly improving model performance.

Fourth, the model uses a regular Graph Convolution operation, which tries to generalise over all nodes on the graph. This means that the *GCN* weights try to generalise over the various object categories, as well as the pedestrian appearance. It would make more sense to learn class- and pedestrian specific weights in the *GCN*, for example via the implementation of a Relational *GCN* [86].

Finally, a general limitation of the 2D on-board domain is that the benchmark metrics are not normalised using image size. This means that results cannot be compared across datasets with different image resolutions, as is the case for *PIE* and *JAAD*. Moreover, there is an imbalance in penalising far-away and close-by pedestrians due to the pixel error metrics. If a far-away and close-by pedestrian both have the same pixel error in the horizontal direction, the model would be penalised identically for both pedestrians. However, the actual distance error would be larger for the far-away pedestrian since every pixel equates to a greater real-world distance.

5.3. Future work

Several recommendations for future work are made. First, the noise in the graph model should be reduced to enable better generalisation. This can be achieved by, for example, pre-processing the instance masks by merging multiple detections of the same object before using them in the graph or by integrating a threshold on the prediction confidence of the instance masks.

Second, a Graph Neural Network should be implemented that can operate on heterogeneous graphs, i.e. where the graph structure is assumed to be heterogeneous with various semantic classes. For example via the implementation of a Relational *GCN* [86]. This would also be an important step in future work where the ego-vehicle is integrated into the graph.

Third, the pre-processing step where the graph node embeddings are generated should be replaced with trainable Convolution Neural Networks. This means the whole network, excluding the scene parsing, can be trained in an end-to-end fashion. This could potentially lead to richer representations of the graph nodes and edge weights.

Fourth, different sets of contextual features should be evaluated; for example, only training and/or testing with crosswalks, vehicles or traffic lights. Moreover, the vehicles category could be split into multiple different types of vehicles, for which the impact on performance can be evaluated. Additionally, more semantic classes could be added to the contextual objects of interest.

Fifth, various recent publications have shown to improve 2D trajectory prediction performance using a different approach compared to this work, for example by conditioning the predictions on future goals [104, 98] or by explicitly disentangling ego-motion from the trajectory predictions via viewpoint normalisation [62]. An interesting future research direction is to combine the graph-based interaction modeling module with either of the previous approaches [98].

Sixth, the current output of all methods in the 2D on-board trajectory prediction domain is on the image plane. This is not directly useable by the ego-vehicle, as this operates in the 3D world. Therefore, this method should be coupled to a monocular depth estimation network [57] to uplift the predictions into 3D. This would also be the start of an interesting experimental study investigating the difference in performance between 3D

on-board methods and 2D on-board methods, whose predictions have been uplifted to 3D.

Finally, this work has indicated the potential of integrating the ground truth ego-motion in the oracle-case experiment, as this led to a performance increase on all metrics between 6.1-32.6%. Therefore, future work should integrate non-ground truth ego-motion into this framework; for example, by using a parallel ego-motion prediction stream.

A

Appendix

A.1. Semantic extraction

Figure A.1 illustrates the number of labeled instances per class contained within the *Mapillary Vitas V1.0*. Any of these instances could be integrated into the interaction modeling module of the proposed method.

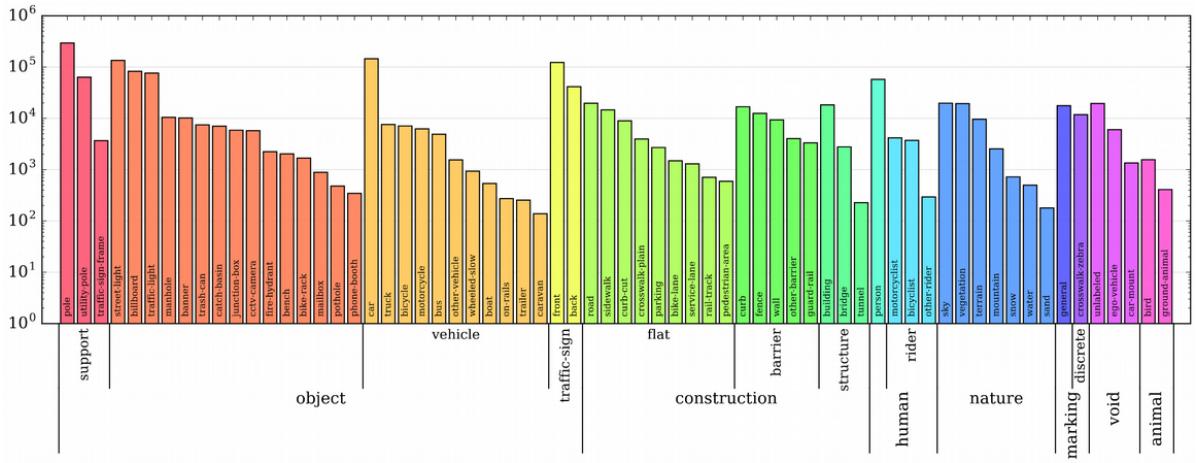


Figure A.1: Illustration of number of labeled instances per class for the *Mapillary Vistas V1.0* dataset. Figure adopted from [61]

Table A.1 provides an overview of the total number of interactions with each object class for the complete *PIE* and $JAAD_{full}$ datasets. The interactions are counted with a track overlap of 80% on $JAAD_{full}$ and 50% on *PIE*.

Object category	**PIE**	**$JAAD_{full}$**
Bicycle	5.1M	1.1M
Bus	0.9M	0.6M
Car	19.6M	18.7M
Motorcycle	1.6M	0.9M
Trailer	0.03M	0.1M
Truck	1.2M	1.0M
Plain crosswalk	0.8M	2.0M
Zebra crosswalk	6.4M	4.1M
Traffic light	18.9M	2.7M

Table A.1: The total number of interactions with each object class for the complete *PIE* and $JAAD_{full}$ datasets. The interactions are counted with a track overlap of 80% on $JAAD_{full}$ and 50% on *PIE*.

A.2. Additional experiments

This appendix section contains additional experiments focusing on architectural choices within the proposed method.

A.2.1. Joint and separate encoding of modalities

The proposed method separately encodes pedestrian motion history and the interactions before fusing them. Alternatively, a combination of joint- and separate encoding has been investigated. Joint encoding adds bounding box information at each time step t-τ+n, where $n \in [1, \tau]$, to the refined pedestrian node embedding of the corresponding time-step before feeding it into the GRU_{enc}^{int}. Jointly encoding interactions and motion history could lead to a better understanding of temporal correlations between these modalities.

		PIE		
		MSE	**C_{MSE}**	**C_{FMSE}**
#	**Ablation**	**0.5s/1.0s/1.5s**	**1.5s**	**1.5s**
1	separate encoding	52/177/550	513	2156
2	separate + joint encoding	51/184/597	561	2367

Table A.2: Ablation study on various encoding paradigms for the interactions- and motion history on the *PIE* dataset

The results show that performance is similar for the short-term 0.5s prediction horizon. However, performance decreases when combining joint- and separate fusion for the 1.0 and 1.5s prediction horizons by 4.0/8.5% *MSE*, 9.4% C_{MSE} and 9.8% C_{FMSE}.

Discussion We see that combined joint- and separate encoding performs worse than only separate encoding. A possible explanation is that the interaction modeling module starts to overfit on the bounding box coordinates, thereby losing information of the relevant interactions. Looking at the literature, we observe that the interaction-aware 2D on-board trajectory prediction method *BiPed* [72] employs a combined joint- and separate encoding paradigm for three different modalities (i.e. ego-motion, motion history and grid locations). However, their interaction modeling module is excluded from this and is fused with the other modalities after separate encoding only. This corresponds to our findings.

A.2.2. Configuration of Graph Convolutional Network

The original implementation of the Graph Convolution Network [39] uses a non-linearity and normalises the adjacency matrix A to account for a varying number of nodes. Moreover, graph convolution layers can be stacked on top of each other to gain more representational power. Therefore, three different graph configurations are ablated. First, normalisation is added by multiplying the adjacency matrix A with its degree matrix D (row 1), i.e. normalising each node based on its number of connections. Second, the adjacency matrix is normalised, and the refined pedestrian node representation after one layer is activated using a *ReLU* activation function (row 3). Third, two *GCN* layers are stacked without any normalisation or activation function (row 4). The results are shown in table A.3.

		PIE		
		MSE	**C_{MSE}**	**C_{FMSE}**
#	**Ablation**	**0.5s/1.0s/1.5s**	**1.5s**	**1.5s**
1	GCN (1 layer)	52/177/550	513	2156
2	GCN (1 layer) + Norm	53/180/565	527	2212
3	GCN (1 layer) + Norm + *ReLU*	53/186/586	547	2286
4	GCN (2 layer)	53/188/606	568	2384

Table A.3: Ablation study on settings of the Graph Convolution Network on the *PIE* dataset

It can be observed that complicating the *GCN* generally leads to performance loss relative to the proposed configuration (row 1), with two exceptions. First, all ablated models perform similar on the the short-term 0.5s prediction horizons and second, only adding normalization (row 2) shows similar performance on all metrics. When additionally adding *ReLU* activations (row 3), the performance drop for the 1.0/1.5s metrics becomes more significant, with losses of 5.1/6.5% *MSE*, 6.6% C_{MSE} and 6.0% C_{FMSE}. Using two *GCN* layers, without any normalization or activations, leads to the greatest performance loss, with values of 6.2/10.2% *MSE*, 10.7% C_{MSE} and 10.6% C_{FMSE}.

Discussion The discussion will focus on the performance difference between the proposed graph configuration (row 1) and the two-layer counter-part (row 4), because this performance difference is most significant. One might expect that a two-layer *GCN* would work better because it adds more parameters to the node refinement process. This would provide, in theory, more representational power. However, it could alternatively give the model too much freedom, impairing the learning process. Moreover, stacking K graphs is used in graphs to reach K^{th}-order nodes [39], i.e. nodes that are K steps away from the pedestrian node. However, our graph only contains 1^{th}-order nodes, i.e. nodes that are only one step away from the pedestrian node. Stacking multiple layers is therefore not necessary in our method for using all nodes.

Bibliography

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. "Social LSTM: Human trajectory prediction in crowded spaces". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.110.

[2] Javad Amirian, Jean Bernard Hayet, and Julien Pettré. "Social ways: Learning multi-modal distributions of pedestrian trajectories with GANs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2019). ISSN: 23318422.

[3] Lamberto Ballan, Francesco Castaldo, Alexandre Alahi, Francesco Palmieri, and Silvio Savarese. "Knowledge transfer for scene-specific motion prediction". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9905 LNCS. Springer Verlag, 2016, pp. 697–713. ISBN: 9783319464473. DOI: 10.1007/978-3-319-46448-0{_}42.

[4] Andrea Bauer, Klaus Dietz, Gerold Kolling, William Hart, and Ulrich Schiefer. "The relevance of stereopsis for motorists: A pilot study". In: *Graefe's Archive for Clinical and Experimental Ophthalmology* 239.6 (2001), pp. 400–406. ISSN: 0721832X. DOI: 10.1007/s004170100273.

[5] Graeme Best and Robert Fitch. "Bayesian intention inference for trajectory prediction with an unknown goal destination". In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2015-December. Institute of Electrical and Electronics Engineers Inc., Dec. 2015, pp. 5817–5823. ISBN: 9781479999941. DOI: 10.1109/IROS.2015.7354203.

[6] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. "Long-Term On-board Prediction of People in Traffic Scenes under Uncertainty". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 457 (2018), pp. 4194–4202. ISSN: 10636919. DOI: 10.1109/CVPR.2018.00441. URL: https://www.dnp.gov.co/estudios-y-publicaciones/estudios-economicos/Paginas/archivos-de-economia.aspx%0Ahttp://www.dotec-colombia.org/index.php/series/118-departamento-nacional-de-planeacion/archivos-de-economia%0Ahttps://www.dnp.gov.co/estudios-y-publica.

[7] Apratim Bhattacharyya, Daniel Olmeda Reino, Mario Fritz, and Bernt Schiele. "Euro-PVI: Pedestrian Vehicle Interactions in Dense Urban Centers". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2021), pp. 6408–6417. DOI: 10.1109/cvpr46437.2021.00634. URL: http://arxiv.org/abs/2106.12442.

[8] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. "Benchmark analysis of representative deep neural network architectures". In: *IEEE Access* 6 (2018), pp. 64270–64277. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2877890.

[9] Smail Ait Bouhsain, Saeed Saadatnejad, and Alexandre Alahi. "Pedestrian Intention Prediction: A Multi-task Perspective". In: *Arxiv* arXiv:2010.10270 (2020). URL: http://arxiv.org/abs/2010.10270.

[10] Allison Bruce and Geoffrey Gordon. *Better Motion Prediction for People-tracking*. Tech. rep.

[11] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, et al. "nuScenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* March (2019), pp. 11621–11631.

[12] Mohamed Chaabane, Ameni Trabelsi, Nathaniel Blanchard, and Ross Beveridge. "Looking ahead: Anticipating pedestrians crossing with future frames prediction". In: *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020* (2020), pp. 2286–2295. DOI: 10.1109/WACV45572.2020.9093426.

[13] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and DInesh Manocha. "Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2019), pp. 8475–8484. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00868.

[14] Rohan Chandra, Tanmay Randhavane, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. "DeepTAgent: Realtime Tracking of Dense Traffic Agents Using Heterogeneous Interaction". In: *Technical Report* (2018). URL: http://gamma.cs.unc.edu/HTI/DeepTAgent.pdf.

[15] Tina Chen, Renran Tian, and Zhengming Ding. *Visual Reasoning using Graph Convolutional Networks for Predicting Pedestrian Crossing Intention*. 2021. DOI: 10.1109/ICCVW54120.2021.00345.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (2014), pp. 1724–1734. DOI: 10.3115/v1/d14-1179.

[17] J Colyar and J Halkias. *US highway 101 dataset*. Tech. rep. Federal Highway Administration (FHWA), 2007.

[18] J Colyar and J Halkias. *US highway 80 dataset*. Tech. rep. Federal Highway Administration (FHWA), 2006.

[19] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *IEEE Conference on Computer Vision and Pattern Recognition* 29.5 (2016), pp. 3213–3223. ISSN: 00015512. DOI: 10.1080/17843286.1974.11735726.

[20] Aaron; Courville, Ian; Goodfellow, and Yoshua Bengio. *Deep Learning*. MIT Press, 2016.

[21] Nachiket Deo and Mohan M. Trivedi. "Convolutional social pooling for vehicle trajectory prediction". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018). ISSN: 23318422.

[22] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang Chieh Chou, Tsung Han Lin, et al. "Uncertainty-aware short-term Motion Prediction of Traffic Actors for Autonomous Driving". In: *arXiv* (2018), pp. 2095–2104.

[23] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, et al. "Large Scale Interactive Motion Forecasting for Autonomous Driving : The Waymo Open Motion Dataset". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021. URL: http://arxiv.org/abs/2104.10133.

[24] J. Frederico Carvalho, Mikael Vejdemo-Johansson, Florian T. Pokorny, and Danica Kragic. "Long-term Prediction of Motion Trajectories Using Path Homology Clusters". In: *IEEE International Conference on Intelligent Robots and Systems* (2019), pp. 765–772. ISSN: 21530866. DOI: 10.1109/IROS40897.2019.8968125.

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the KITTI vision benchmark suite". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2012), pp. 3354–3361. ISSN: 10636919. DOI: 10.1109/CVPR.2012.6248074.

[26] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision* 2015 Inter (2015), pp. 1440–1448. ISSN: 15505499. DOI: 10.1109/ICCV.2015.169.

[27] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. "Transformer Networks for Trajectory Forecasting". In: *In Proceedings of International Conference on Pattern Recognition (ICPR)* preprint arXiv:2003.08111 (2020). URL: http://arxiv.org/abs/2003.08111.

[28] Karol Gregor, Ivo; Danihelka, Alex; Graves, Danilo Jimenez; Rezende, and Daan; Wierstra. "DRAW: A Recurrent Neural Network For Image Generation". In: *Proceedings of the 32nd International Conference on Machine Learning* 37 (2015).

[29] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. "Social GAN: Socially acceptable trajectories with generative adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2255–2264. ISSN: 23318422.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem (2016), pp. 770–778. ISSN: 10636919. DOI: 10.1109/CVPR.2016.90.

[31] Dirk Helbing and Péter Molnár. "Social force model for pedestrian dynamics". In: *Physical Review E* 51.5 (1995), pp. 4282–4286. ISSN: 1063651X. DOI: 10.1103/PhysRevE.51.4282.

[32] Sepp Hochreiter; and Jürgen Schmidhuber; "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 00309923. DOI: 10.17582/journal.pjz/2018.50.6.2199.2207.

[33] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. "STGAT: Modeling spatial-temporal interactions for human trajectory prediction". In: *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 6271–6280. ISSN: 15505499. DOI: 10.1109/ICCV.2019.00637.

[34] Joshua Joseph, Finale Doshi-Velez, Albert S Huang, Nicholas Roy, J Joseph, F Doshi-Velez, et al. "A Bayesian nonparametric approach to modeling motion patterns". In: *Springer* 31.4 (Nov. 2011), pp. 383–400. DOI: 10.1007/s10514-011-9248-x. URL: http://maps.google.com..

[35] Alex Kendall and Yarin Gal. "What uncertainties do we need in Bayesian deep learning for computer vision?" In: *Advances in Neural Information Processing Systems* 2017-Decem.Nips (2017), pp. 5575–5585. ISSN: 10495258.

[36] Kyungdo Kim, Yoon Kyung Lee, Hyemin Ahn, Sowon Hahn, and Songhwai Oh. "Pedestrian Intention Prediction for Autonomous Driving Using a Multiple Stakeholder Perspective Model". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2020), pp. 7957–7962.

[37] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015), pp. 1–15.

[38] Thomas Kipf, Ethan Fetaya, Kuan Chieh Wang, Max Welling, and Richard Zemel. "Neural Relational Inference for Interacting Systems". In: *International Conference on Machine Learning* (2018), pp. 2688–2697. ISSN: 23318422.

[39] Thomas N. Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2017), pp. 1–14.

[40] Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. "Activity forecasting". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7575 LNCS. PART 4. Springer, Berlin, Heidelberg, 2012, pp. 201–214. ISBN: 9783642337642. DOI: 10.1007/978-3-642-33765-9{_}15. URL: https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-642-33765-9_15.

[41] Julian F.P. Kooij, Fabian Flohr, Ewoud A.I. Pool, and Dariu M. Gavrila. "Context-Based Path Prediction for Targets with Switching Dynamics". In: *International Journal of Computer Vision* 127.3 (2019), pp. 239–262. ISSN: 15731405. DOI: 10.1007/s11263-018-1104-4. URL: https://doi.org/10.1007/s11263-018-1104-4.

[42] Julian Francisco Pieter Kooij, Nicolas Schneider, Fabian Flohr, and Dariu M. Gavrila. "Context-based pedestrian path prediction". In: *European Conference on Computer Vision*. 2014, pp. 618–633. ISBN: 9783319105987. DOI: 10.1007/978-3-319-10599-4{_}40.

[43] Philip Koopman and Michael Wagner. "Autonomous Vehicle Safety: An Interdisciplinary Challenge". In: *IEEE Intelligent Transportation Systems Magazine* 9.1 (2017), pp. 90–96. ISSN: 19391390. DOI: 10.1109/MITS.2016.2583491.

[44] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S. Hamid Rezatofighi, and Silvio Savarese. "Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks". In: *Advances in Neural Information Processing Systems* 33 (2019), pp. 1–10. ISSN: 10495258.

[45] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. ISSN: 00189219. DOI: 10.1109/5.726791.

[46] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip H S Torr, and Manmohan Chandraker. "DESIRE : Distant Future Prediction in Dynamic Scenes with Interacting Agents". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2017), pp. 336–345.

[47] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. "Crowds by example". In: *Computer Graphics Forum* 26.3 (2007), pp. 655–664. ISSN: 14678659. DOI: 10.1111/j.1467-8659.2007.01089.x.

[48] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. "EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning". In: *Proceedings of the Neural Information Processing Systems* (2020). URL: http://arxiv.org/abs/2003.13924.

[49] Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, et al. "End-to-end contextual perception and prediction with interaction transformer". In: *IEEE International Conference on Intelligent Robots and Systems* (2020), pp. 5784–5791. ISSN: 21530866. DOI: 10.1109/IROS45743.2020.9341392. URL: http://arxiv.org/abs/2008.05927.

[50] Bingbin Liu, Ehsan Adeli, Zhangjie Cao, Kuan Hui Lee, Abhijeet Shenoi, Adrien Gaidon, et al. "Spatiotemporal Relationship Reasoning for Pedestrian Intent Prediction". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3485–3492. ISSN: 23773766. DOI: 10.1109/LRA.2020.2976305.

[51] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2018), pp. 6120–6127. ISSN: 23318422.

[52] Osama Makansi, Ozgun Cicek, Kevin Buchicchio, and Thomas Brox. "Multimodal Future Localization and Emergence Prediction for Objects in Egocentric View with a Reachability Prior". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), pp. 4353–4362. ISSN: 10636919. DOI: 10.1109/CVPR42600.2020.00441.

[53] Srikanth Malla and Chiho Choi. "NEMO: Future object localization using noisy ego priors". In: *arXiv preprint arXiv:1909.08150* (2019).

[54] Srikanth Malla, Behzad Dariush, and Chiho Choi. "TITAN: Future forecast using action priors". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11186–11196. DOI: 10.1109/CVPR42600.2020.01120.

[55] H Martin, K Tschabuschnig, O Bridal, and D Watzenig. *Functional Safety of Automated Driving Systems: Does ISO 26262 Meet the Challenges?* 2017, pp. 387–416. ISBN: 9783319318950. DOI: 10.1007/978-3-319-31895-0.

[56] A. J; McKnight, D; Shinar, and B Hilburn. "The visual and driving performance of monocular and binocular heavy-duty truck drivers". In: *Accident Analysis & Prevention* 23.4 (1991), pp. 225–237.

[57] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. "Deep learning for monocular depth estimation: A review". In: *Neurocomputing* 438 (2021), pp. 14–33. ISSN: 18728286. DOI: 10.1016/j.neucom.2020.12.089. URL: https://doi.org/10.1016/j.neucom.2020.12.089.

[58] Andreas Mogelmose, Mohan M. Trivedi, and Thomas B. Moeslund. "Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations". In: *IEEE Intelligent Vehicles Symposium, Proceedings* 2015-Augus.Iv (2015), pp. 330–335. DOI: 10.1109/IVS.2015.7225707.

[59] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. "Social-STGCNN: A social spatio-temporal graph convolutional neural network for human trajectory prediction". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), pp. 14412–14420. ISSN: 10636919. DOI: 10.1109/CVPR42600.2020.01443.

[60] Satyajit Neogi, Michael Hoy, Kang Dang, Hang Yu, and Justin Dauwels. "Context model for pedestrian intention prediction using factored latent-dynamic conditional random fields". In: *IEEE Transactions on Intelligent Transportation Systems* (2019), pp. 1–12. ISSN: 1524-9050. DOI: 10.1109/tits.2020.2995166.

[61] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *Proceedings of the IEEE International Conference on Computer Vision* 2017-Octob (2017), pp. 5000–5009. ISSN: 15505499. DOI: 10.1109/ICCV.2017.534.

[62] Lukas Neumann and Andrea Vedaldi. "Pedestrian and Ego-vehicle Trajectory Prediction from Monocular Camera". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2021), pp. 10204–10212.

[63] U.S. Department of Trnasportation NHTSA. "National Motor Vehicle Crash Causation Survey: Report to Congress". In: *NHTSA* DOT HS 811.July (2008).

[64] Adam; Paszke, Sam; Gross, Francisco; Massa, Adam; Lerer, James; Bradbury, Gregory; Chanan, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[65] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *Proceedings of the IEEE International Conference on Computer Vision* Iccv (2009), pp. 261–268. DOI: 10.1109/ICCV.2009.5459260.

[66] Atanas Poibrenski, Matthias Klusch, Igor Vozniak, and Christian Müller. "M2P3: Multimodal multi-pedestrian path prediction by self-driving cars with egocentric vision". In: *Proceedings of the ACM Symposium on Applied Computing* (2020), pp. 190–197. DOI: 10.1145/3341105.3373877.

[67] Ewoud A.I. Pool, Julian F.P. Kooij, and Dariu M. Gavrila. "Context-based cyclist path prediction using Recurrent Neural Networks". In: *IEEE Intelligent Vehicles Symposium, Proceedings* 2019-June.Iv (2019), pp. 824–830. DOI: 10.1109/IVS.2019.8813889.

[68] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. "Seamless scene segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June (2019), pp. 8269–8278. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00847.

[69] Ruijie Quan, Linchao Zhu, Yu Wu, and Yi Yang. "Holistic LSTM for Pedestrian Trajectory Prediction". In: *IEEE Transactions on Image Processing* 30.8 (2021), pp. 3229–3239. ISSN: 19410042. DOI: 10.1109/TIP.2021.3058599.

[70] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John Tsotsos. "PIE: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction". In: *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 6261–6270. ISSN: 15505499. DOI: 10.1109/ICCV.2019.00636.

[71] Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos. "Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior". In: *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017* 2018-Janua (2017), pp. 206–213. DOI: 10.1109/ICCVW.2017.33.

[72] Amir Rasouli, Mohsen Rohani, and Jun Luo. "Bifold and Semantic Reasoning for Pedestrian Behavior Prediction". In: *Proceedings of the IEEE International Conference on Computer Vision* (2021), pp. 15600–15610. URL: http://arxiv.org/abs/2012.03298.

[73] Amir Rasouli, Mohsen Rohani, and Jun Luo. "Pedestrian behavior prediction via multitask learning and categorical interaction modeling". In: *arXiv preprint arXiv:2012.03298* (2020). ISSN: 23318422. URL: http://arxiv.org/abs/2012.03298.

[74] Amir Rasouli and John K. Tsotsos. "Autonomous vehicles that interact with pedestrians: A survey of theory and practice". In: *IEEE Transactions on Intelligent Transportation Systems* 21.3 (2020), pp. 900–918. ISSN: 15580016. DOI: 10.1109/TITS.2019.2901817.

[75] Amir Rasouli, Tiffany Yau, Peter Lakner, Saber Malekmohammadi, Mohsen Rohani, and Jun Luo. "PePScenes: A novel dataset and baseline for pedestrian action prediction in 3D". In: *arXiv* NeurIPS (2020). ISSN: 23318422. URL: http://arxiv.org/abs/2012.07773.

[76] Amir Rasouli, Tiffany Yau, Mohsen Rohani, and Jun Luo. "Multi-modal hybrid architecture for pedestrian action prediction". In: *arXiv* (2020). ISSN: 23318422.

[77] Eike Rehder, Florian Wirth, Martin Lauer, and Christoph Stiller. "Pedestrian Prediction by Planning Using Deep Neural Networks". In: *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., Sept. 2018, pp. 5903–5908. ISBN: 9781538630815. DOI: 10.1109/ICRA.2018.8460203.

[78] Daniela A. Ridel, Nachiket Deo, Denis Wolf, and Mohan Trivedi. "Understanding pedestrian-vehicle interactions with vehicle mounted vision: An LSTM model and empirical analysis". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV) 2019* (2019), pp. 913–918. ISSN: 23318422.

[79] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. "Learning social etiquette: Human trajectory understanding in crowded scenes". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2016. ISBN: 9783319464831. DOI: 10.1007/978-3-319-46484-8{_}33.

[80] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M. Kitani, Dariu M. Gavrila, and Kai O. Arras. "Human motion trajectory prediction: a survey". In: *International Journal of Robotics Research* (2020). ISSN: 17413176. DOI: 10.1177/0278364920917446.

[81] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. "Learning Representations by Back-Propagating Errors". In: *Nature* 323 (1986), pp. 533–536. DOI: 10.7551/mitpress/1888.003.0013.

[82] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. ISSN: 15731405. DOI: 10.1007/s11263-015-0816-y. URL: http://dx.doi.org/10.1007/s11263-015-0816-y.

[83] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, S. Hamid Rezatofighi, and Silvio Savarese. "Sophie: An attentive GAN for predicting paths compliant to social and physical constraints". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2019), pp. 1349–1358. ISSN: 23318422.

[84] Amir Sadeghian, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese. "CAR-Net: Clairvoyant Attentive Recurrent Network". In: *Proceedings of the European Conference on Computer Vision* (2018), pp. 162–180. ISSN: 16113349. DOI: 10.1007/978-3-030-01252-6{_}10.

[85] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. "Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data". In: *Proceedings of the European Conference on Computer Vision* (2020), pp. 683–700. ISSN: 16113349. DOI: 10.1007/978-3-030-58523-5{_}40.

[86] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. "Modeling Relational Data with Graph Convolutional Networks". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10843 LNCS.1 (2018), pp. 593–607. ISSN: 16113349. DOI: 10.1007/978-3-319-93417-4{_}38.

[87] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. "What the constant velocity model can teach us about pedestrian motion prediction". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1696–1703.

[88] AT Schulz, R Stiefelhagen - 2015 IEEE 18th International, and undefined 2015. "A controlled interactive multiple model filter for combined pedestrian intention recognition and path prediction". In: *ieeexplore.ieee.org* (). URL: https://ieeexplore.ieee.org/abstract/document/7313129/?casa_token=-TQX1KlFCC4AAAAA:hhedEe_MtIBe_FW6SbF4h1b_Y0Z_hqe_fhXlIknP9kacTl11hAP2VzxuMcybtJN15HNLZ-itPw.

[89] Jens Schulz, Constantin Hubmann, Julian Löchner, and Darius Burschka. "Interaction-aware probabilistic behavior prediction in urban environments". In: *Proceedingsd of the IEEE/RSL International Conference on Intelligent Robots and Systems* (2018), pp. 3999–4006. ISSN: 23318422.

[90] Olly Styles, Tanaya Guha, and Victor Sanchez. "Multiple object forecasting: Predicting future object locations in diverse environments". In: *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020* (2020), pp. 690–699. ISSN: 23318422.

[91] Olly Styles, Arun Ross, and Victor Sanchez. "Forecasting pedestrian trajectory with machine-annotated training data". In: *IEEE Intelligent Vehicles Symposium (IV)* Iv (2019), pp. 716–721. ISSN: 23318422.

[92] Ze Sui, Yue Zhou, Xu Zhao, Ao Chen, and Yiyang Ni. "Joint Intention and Trajectory Prediction Based on Transformer". In: *IEEE International Conference on Intelligent Robots and Systems* (2021), pp. 7082–7088. ISSN: 21530866. DOI: 10.1109/IROS51168.2021.9636241.

[93] John K. Tsotsos, Amir Rasouli, and Iuliia Kotseruba. "Pedestrian action anticipation using contextual feature fusion in stacked RNNs". In: *British Machine Vision Conference* (2020), pp. 1–13.

[94] Dizan Vasquez. "Novel planning-based algorithms for human motion prediction". In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2016-June. Institute of Electrical and Electronics Engineers Inc., June 2016, pp. 3317–3322. ISBN: 9781467380263. DOI: 10.1109/ICRA.2016.7487505.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems* 2017-Decem.Nips (2017), pp. 5999–6009. ISSN: 10495258.

[96] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph attention networks". In: *arXiv* (2017), pp. 1–12. ISSN: 23318422.

[97] Chengxin Wang, Shaofeng Cai, and Gary Tan. "GraphTCN: Spatio-Temporal Interaction Modeling for Human Trajectory Prediction". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020), pp. 3450–3459. ISSN: 23318422.

[98] Chuhua Wang, Yuchen Wang, Mingze Xu, and David J. Crandall. "Stepwise Goal-Driven Networks for Trajectory Prediction". In: *IEEE Robotics and Automation Letters*. 2. 2021. URL: http://arxiv.org/abs/2103.14107.

[99] World Health Organization (WHO). "Global Status Report on Road Safety". In: *World Health Organization* (2018), p. 20.

[100] Hui Xiong, Fabian B. Flohr, Sijia Wang, Baofeng Wang, Jianqiang Wang, and Keqiang Li. "Recurrent neural network architectures for vulnerable road user trajectory prediction". In: *IEEE Intelligent Vehicles Symposium, Proceedings* 2019-June.May 2020 (2019), pp. 171–178. DOI: 10.1109/IVS.2019.8814275.

[101] Kelvin; Xu, Jimmy; Lei Ba, Ryan; Kiros, Kyunghyun; Cho, Aaron; Courville, Ruslan; Salakhutdinov, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of the 32nd International Conference on Machine Learning*, 37 (2015).

[102] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. "Future Person Localization in First-Person Videos". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 7593–7602. ISSN: 10636919.

[103] Kota Yamaguchi, Alexander C. Berg, Luis E. Ortiz, and Tamara L. Berg. "Who are you with and where are you going?" In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2011. ISBN: 9781457703942. DOI: 10.1109/CVPR.2011.5995468.

[104] Yu Yao, Ella Atkins, Matthew Johnson-Roberson, Ram Vasudevan, and Xiaoxiao Du. "BiTraP: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation". In: *proceedings of the IEEE Robotics and Automation Letters 2021* 6.2 (2021), pp. 1463–1470. ISSN: 23318422. DOI: 10.1109/lra.2021.3056339.

[105] Yu Yao, Mingze Xu, Yuchen Wang, David J Crandall, Ella M Atkins, and C V Jul. "Unsupervised Traffic Accident Detection in First-Person Videos". In: *IEEE International Conference on Intelligent Robots and Systems*. 2019, pp. 273–280.

[106] Tiffany Yau, Saber Malekmohammadi, Amir Rasouli, Peter Lakner, Mohsen Rohani, and Jun Luo. "GraphSIM: A graph-based spatiotemporal interaction modelling for pedestrian action prediction". In: *EEE International Conference on Robotics and Automation* (2021). ISSN: 23318422. URL: http://arxiv.org/abs/2012.02148.

[107] Youngjoon Yoo, Kimin Yun, Sangdoo Yun, Jonghee Hong, Hawook Jeong, and Jin Young Choi. "Visual Path Prediction in Complex Scenes with Crowded Moving Objects". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem (2016), pp. 2668–2677. ISSN: 10636919. DOI: 10.1109/CVPR.2016.292.

[108] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. "Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction". In: *Proceedings of the European Conference on Computer Vision* (2020), pp. 507–523. ISSN: 16113349. DOI: 10.1007/978-3-030-58610-2{_}30.

[109] Kuo Hao Zeng, Shih Han Chou, Fu Hsiang Chan, Juan Carlos Niebles, and Min Sun. "Agent-centric risk assessment: Accident anticipation and risky region localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2017-Janua (2017), pp. 1330–1338. DOI: 10.1109/CVPR.2017.146.

[110] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, et al. "Graph neural networks: A review of methods and applications". In: (2021). DOI: 10.1016/j.aiopen.2021.01.001. URL: https://doi.org/10.1016/j.aiopen.2021.01.001.

[111] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, et al. “Planning-based prediction for pedestrians”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Dec. 2009, pp. 3931–3936. ISBN: 9781424438044. DOI: 10.1109/IROS.2009.5354147.