Conditional Behavior Prediction of Interacting Agents on Map Graphs with Neural Networks

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

Florian Josef Wirth, M.Sc.
geb. in Friedrichshafen

Tag der mündlichen Prüfung: 31.07.2023

Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr.-Ing. Steven Peters

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mess- und Regelungstechnik (MRT) des Karlsruher Instituts für Technologie (KIT). Sie wäre nicht möglich gewesen ohne die vielfältige Unterstützung meiner Kollegen, Familie und Freunde. Ich möchte mich bei ein paar Leuten namentlich für deren Unterstützung bedanken, auch wenn die Liste der Leute, denen im Zusammenhang mit meiner Forschungsarbeit mein Dank gilt, deutlich länger ist.

Danke an Prof. Dr.-Ing. Christoph Stiller für die Möglichkeit am Institut für Mess- und Regelungstechnik am KIT lernen, lehren und forschen zu dürfen. Ebenfalls danke für viel fachlichen Input auf unseren internen wissenschaftlichen Seminaren und in persönlichen Gesprächen. Gleiches gilt für alle Mitglieder unserer Gruppenleiterebene, stellvertretend seien Dr. rer. nat. Martin Lauer und Dr. Carlos Fernández López genannt.

Danke an Prof. Dr.-Ing. Steven Peters, dass er das Korreferat für meine Arbeit übernommen hat und an Prof. Dr.-Ing. Marcus Geimer für die Übernahme des Vorsitzes des Promotionsprüfungsausschusses.

Thanks to the representatives of the industrial partners supporting my research work for many years. Representatively, Jeff Ota, Maria Soledad Elli, Ignacio Alvarez and Fabian Oboril be highlighted.

Stellvertretend für die vielen, die meine Arbeit in Teilen probegelesen haben, danke an Ricarda, Eike, Şahin und Frank, die sich meine komplette Arbeit vor der Einreichung zu Gemüte geführt haben. Danke an Jan, der sich tapfer selbst den komplizierteren Formeln angenommen hat.

Danke an die vielen Abschlussarbeiter, die mit ihrer Arbeit die Richtung meiner Forschung beeinflusst haben. Stellvertretend seien hier aufgrund der zeitlichen und inhaltlichen Nähe zu meiner Doktorarbeit Alexander Scheid und Julian Wadephul genannt.

Danke für die vielen fachlichen Diskussionen rund um Prädiktion und den hilfreichen Input an Eike, Jannik, Şahin, Carlos, Sascha, Thomas Gilles und

Fabian, stellvertretend auch für quasi alle aktuellen und ehemaligen Kollegen.

Natürlich möchte ich auch denen danken, von denen ich als gelernter Festkörpermechaniker und Strömungssimulant in die Welt des Programmierens, des maschinellen Lernens und des automatischen Fahrens mit all seinen Facetten eingeführt wurde. Das betrifft vor allem die (Dienst-)älteren Kollegen, die mir als (Dienst-)jüngerem hier unter die Arme gegriffen haben und die fachliche Betreuung im Maschinenraum der Programmierung am MRT übernommen haben: Marc, Sascha, Ole, Pio, Fabian, Johannes, Eike, Jan, Hendrik, Eduardo.

Generell danke an alle, die Hilfsbereitschaft und Kollegialität am MRT an den Tag legen, und damit das Institut für Kollegen und Studenten zu einem geeigneten Ort machen, an dem neue Ideen entwickelt, ausgearbeitet und direkt im Straßenverkehr erprobt werden können.

Mein Dank gilt auch meiner Familie und meinen Freunden, die mich durch die Tiefs der Promotionszeit getragen haben und die Interesse und Unterstützung für meine Forschungsarbeit und meine wissenschaftlichen Interessensgebiete gezeigt haben.

Karlsruhe im August 2023

Florian Josef Wirth

Zusammenfassung

Solange Verkehrsteilnehmer ihre Manöverabsicht und ihre geplante Trajektorie automatischen Fahrzeugen nicht mitteilen können, ist eine Verhaltensvorhersage für alle beteiligten Verkehrsteilnehmer erforderlich. Mit einer solchen Vorhersage kann das Verhalten eines automatischen Fahrzeugs vorausschauend generiert und damit komfortabler und energieeffizienter gemacht werden, was den Verkehrsfluss verbessert.

Es wird ein künstliches neuronales Netz für Graphen (GNN) vorgestellt, das verschiedene probabilistische Positionsvorhersagen für interagierende Agenten zur Analyse bereitstellt. Das vorliegende Anwendungsbeispiel ist die Verkehrssituationsanalyse für das automatische Fahren, für welches ein diskretisierter Vorhersagezeitraum von einigen Sekunden als relevant angesehen wird. Das GNN propagiert einen vollvernetzten, gerichteten Agentengraphen probabilistisch durch einen dünnvernetzten, gerichteten Kartengraphen. Merkmale des Agentengraphen, der aus Verkehrsteilnehmern und deren Beziehungen besteht, sowie Merkmale des Kartengraphen, der aus Fahrbahnstücken und deren geometrischer, sowie verkehrsregelbezogenen Verbindungen besteht, können für die Vorhersage verwertet werden.

Das Modell prädiziert für jeden Agenten zu jedem Prädiktionszeitpunkt eine diskrete Aufenthaltswahrscheinlichkeitsverteilung über alle Fahrbahnstücke des Kartengraphen. Eine solche Prädiktion ist in der wissenschaftlichen Literatur zwar üblich, setzt aber für deren stochastische Interpretierbarkeit und damit Anwendbarkeit statistische Unabhängigkeit des zukünftigen Verhaltens der Verkehrsteilnehmer voraus. Da diese Annahme bei interagierenden Agenten als unzulässig erachtet wird, prädiziert das Modell darüber hinaus für alle Agentenpaare diskrete Verbundwahrscheinlichkeitsverteilungen. Aus diesen können bedingte Prädiktionen gegeben möglicher zukünftiger Positionen einer der beiden Agenten berechnet werden.

In der Evaluierung werden gängige Metriken für den vorliegenden Fall angepasst und verschiedene Modellierungstiefen einander gegenübergestellt. Sowohl die individuelle Prädiktion als auch die bedingte Prädiktion werden erfolgreich auf Genauigkeit und statistischer Zuverlässigkeit untersucht.

Abstract

As long as traffic participants (TPs) cannot share information about their maneuver intention or their planned trajectory with automated vehicles (AVs), behavior prediction for these TPs is required. Such a prediction enables the proactive generation of AV behavior making driving more comfortable and energy-efficient, besides improving traffic flow.

An artificial graph neural network (GNN) is presented which provides probabilistic position predictions for interacting agents as basis for further analysis. The present use case is analysis of traffic situation for automated driving. A discretized prediction period of several seconds is considered relevant. The GNN probabilistically propagates a fully-connected, directed agent graph through a sparsely-connected, directed map graph. Features of the agent graph, consisting of agents and their relationships, as well as features of the map graph, consisting of lane tiles and their geometric as well as traffic rule-related connections, can be utilized for prediction generation.

The model predicts a discrete occupancy probability distribution over all lane tiles of the map graph. While this type of prediction is common in scientific publications, it assumes statistical independence of all agents' future behavior for its stochastic interpretability and thus applicability. Since this assumption is considered invalid for interacting agents, the model further predicts discrete joint probability distributions for pairs of agents. This allows conditional predictions given possible future position of either of both agents.

In the evaluation, common metrics are adapted to the discrete output at hand and different modeling depths are juxtaposed. Both individual prediction and conditional prediction are successfully examined for accuracy and statistical reliability.

Table of Content

Acronyms

ADAS	advanced driver assistance system
ADE	average displacement error
AD	automated driving
AI	artificial intelligence
ANN	artificial neural network
AV	automated vehicle
CNN	convolutional neural network
FDE	final displacement error
GCN	graph convolutional network
GNN	graph neural network
GPU	graphical processing unit
GRU	gated recurrent unit
GT	ground truth
HD	high definition
IoU	intersection over union
KF	Kalman filter
LHT	left-hand traffic
LSTM	long short-term memory
MLP	multi-layer perceptron
MRT	Institut für Mess- und Regelungstechnik
RHT	right-hand traffic
RNN	recurrent neural network
SpAGNN	Spacially-Aware Graph Neural Network (competing approach)
TP	traffic participant
VRU	vulnerable road user

Operators and Symbols

a	Scalar.
$\mathbf{a}$	Vector with entries a_i.

$\mathbf{A}$	Matrix with entries $\mathrm{A}_{i,j}$.
$\boldsymbol{A}, \boldsymbol{\mathcal{A}}$	Tensor with entries $A_{i,j,\ldots,k}$.
$\mathbb{A}$	Sets and tuples.
$\mathcal{L}, \mathcal{C}$	Scalar loss and tensorial cross-entropy.
$n_{(\cdot)}$	Size of a set finite set.
$\mathbf{a}^{(\cdot)}$	Variable $\mathbf{a}$ refers to expression in brackets. Usually a layer index λ. Unlike for upper indices without brackets, sizes of elements may differ.
$\varphi(\cdot)$	Function for processing $(\cdot)$ with defined input and output. Usually a small fully-connected feed-forward artificial neural network (ANN) for which no specific architecture is defined. Can have an arbitrary activation function, number of layers, neurons, etc.
$\alpha(\cdot)$	Aggregator function (or *aggregator*) that reduces a set of arbitrary size with values of same type – usually feature vectors – to one value of the same type. Order invariant operation if not defined otherwise. Does not contain trainable weights.
$\equiv$	Denotes equality of two terms with different notations in a formula: $\mathbf{x} \equiv \mathrm{x}_i$.
$\tilde{(\cdot)}$	Processed value of same shape as $(\cdot)$. Placeholder.
$(\cdot)'$	Derivative.
$[\ldots, \ldots]$	Concatenation along the non-shared dimension, while all other dimensions o are shared. e.g. $[\mathbf{A}, \mathbf{B}] \in \mathbb{R}^{n+m \times o} \forall \mathbf{A} \in \mathbb{R}^{n \times o}, \mathbf{B} \in \mathbb{R}^{m \times o}$.
$[\ldots]$	Concatenation along indices, e.g. $[\mathbf{a}^i] = \mathbf{A} \in \mathbb{R}^{n_\mathrm{A} \times n_\mathrm{I}}, \mathbf{a}^i \in \mathbb{R}^{n_\mathrm{A}}$
$\odot$	Entry-wise multiplication between two tensors of same shape.

$\underset{i}{\text{norm}}$	Normalize through dividing each element by the sum along dimension i.
$\lvert\mathbf{x}\rvert$	Absolute/unsigned value.
$\lvert\lvert\mathbf{x}\rvert\rvert_1$	The euclidean 1-norm, sum over absolutes of vector entries $\sum_i \lvert x_i \rvert$.
$\lvert\lvert\mathbf{x}\rvert\rvert_2$	The euclidean 2-norm, sum over squares of vector entries $\sum_i x_i^2$.

Naming Indices

A	Agent.
AE	Agent edge.
Cand	Candidates for position.
Joint	Joint prediction.
Conf	Conflict (Identifier).
Final	Final prediction of statistically independent positions with interaction.
L	Lane tile.
LE	Lane tile connection (edge of the map graph).
Map	Map-based prediction without interaction.
Mess	Message (Aggregator).
NEN	Node-edge-node.
Notif	Notification (Extractor).
Pos	Corresponding to a position.
ph	Prediction horizon.
R	Receiving lane tile.

S	Sending lane tile.
SelfMess	Self-Message (Extractor).
$\succ$	Merging position conflicts.
$\times$	Crossing position conflicts.

Glossary

a	Agent index.
b	Agent index.
c	Conflict intensity between two agents.
$\mathbf{c}$	Vector describing the conflict between and agent w.r.t. another agent.
d	Scalar edge direction feature.
$\mathbf{e}$	Edge feature element.
$\mathbf{f}$	Vector of features, further defined through upper and/or lower indices..
$\mathbf{f}^a$	Agent feature vector.
$\mathbf{f}^{a,b}$	Agent edge feature vector.
$\mathbf{f}^l$	Lane tile feature vector.
$\mathbf{f}^{l_\mathrm{R},l_\mathrm{S}}$	lane tile connection (edge) feature vector.
$\mathbf{g}$	Graph feature element.
i	Index for dimension not further specified. Usually a feature index.
j	Index for dimension not further specified. Usually a feature index.
k	Lane tile index.
l	Lane tile index.
l_R	Receiving or “to” lane tile index.
l_S	Sending or “from” lane tile index.
m	Index for dimension not further specified. Usually a feature index.
$\mathbf{m}$	Message vector.
n	Integer number, size of a set..
n_A	Number of agents.
n_AE	Number of agent edges.
n_Data	Number of data pairs in a dataset.
n_E	Number of edges in a graph.

n_{F}	Number of features. If no further index is given, a general feature number may be given, or the number varies depending on the context.
$n_{\mathrm{F,A}}$	Number of agent features.
$n_{\mathrm{F,AE}}$	Number of agent edge features.
$n_{\mathrm{F,Conf}}$	Number of conflict features.
$n_{\mathrm{F,E}}$	Number of edge features.
$n_{\mathrm{F,L}}$	Number of lane tile features.
$n_{\mathrm{F,LE}}$	Number of lane tile edge features.
$n_{\mathrm{F,Mess}}$	Number of message features.
$n_{\mathrm{F,Notif}}$	Number of notification features.
$n_{\mathrm{F,S}}$	Number of state features.
$n_{\mathrm{F,SelfMess}}$	Number of self messaging features.
$n_{\mathrm{F,Y}}$	Number of output features.
n_{L}	Number of lane tiles.
n_{LE}	Number of lane tiles connections/map graph edges.
n_{N}	Number of nodes.
n_{T}	Number of time steps.
n_{λ}	Number of layers.
$\mathbf{n}$	Node feature element.
$\mathbf{o}$	Output state of a LSTM cell.
p	Entry of position vector.
$\mathbf{p}$	Vector of positions.
$\mathbf{p}_{\mathrm{Cand}}$	Contains all lane tiles the front axle rectangle of an agent a has an intersection over union (IoU) with. Identifies, that an agent can potentially be located at precisely that lane tile of the map subgraph.
$\mathbf{p}^{a}_{\mathrm{LTM}}$	Positional vector that contains the probabilities of agent a being currently driving on the respective lane tiles.
$\mathbf{p}_{\mathrm{GT}}$	Vector of ground truth (GT) positions.
$\mathbf{s}$	State variable.
t	Time step index.
t_{ph}	Time point in the future the prediction refers to.
Δt	Time difference between two consecutive prediction steps.
Δt_{ph}	Time horizon of the prediction.

$\mathbf{t}$	Vector of target values.
$\tilde{\mathbf{t}}^{a,l}$	Non-redundant transformation features from $\mathbf{T}^{a,l}$ between agent a and lane tile l.
$\mathbf{u}$	Notification vector.
v	Speed.
$\dot{v}$	Acceleration.
$\mathbf{v}$	Eigen vector.
$\mathbf{w}$	Vector of weights.
$x^{a,l}$	The lateral translation between agent a and lane tile l.
x	Entry of input vector, further defined through upper and/or lower indices. See Section 2.1.2.
$\mathbf{x}$	Vector of input features.
$\mathbf{x}^{(\lambda)}$	Vector of input features.
$y^{a,l}$	The longitudinal translation between agent a and lane tile l.
y	Entry of output vector, further defined through upper and/or lower indices. See Section 2.1.2.
$\mathbf{y}$	Output vector.
z	Time index.
A	Adjacency matrix entry.
$\mathbf{A}$	Adjacency matrix.
C	Conflicts map entry between two agents.
$\mathbf{C}$	Conflict map between two agents.
$\mathbf{C}_{\succ}$	Merging conflicts map between two agents.
$\mathbf{C}_{\times}$	Crossing conflicts map between two agents.
C	Conflicts map entry between all agents.
$\boldsymbol{C}_{\succ}$	Merging conflicts map between all agents.
$\boldsymbol{C}_{\times}$	Crossing conflicts map between all agents.
$\boldsymbol{C}$	Conflicts map between all agents.
D	Degree matrix entry.
$\mathbf{D}$	Degree matrix.
E	Refers to an edge.
$\mathbf{F}$	Feature matrix.
$\mathbf{F}_{\mathrm{L}}$	Lane tile feature matrix.
$\mathbf{F}_{\mathrm{A}}$	Agent feature tensor.
$\boldsymbol{F}$	Feature tensor.

$\boldsymbol{F}_{\mathrm{AE}}$	Agent edge tensor matrix.
$\boldsymbol{F}_{\mathrm{LE}}$	Lane tile connection (edge) feature tensor.
$\mathbf{h}$	Hidden states in vector form.
$\boldsymbol{H}$	Hidden states in tensor form.
I	Entry of an exemplary matrix for general transitions from one lane tile to the other.
$\mathbf{I}$	Exemplary transition matrix for general transitions from one lane tile to the other.
$\mathbf{L}$	Transition matrix from one lane tile to the lane tile on its left hand.
$\mathbf{M}$	Message Matrix.
N	Refers to a node.
$\boldsymbol{O}$	Output states in tensor form.
P	Entry of position matrix.
$\mathbf{P}$	Vector of matrix.
$\mathbf{P}_{\mathrm{Cand}}$	Contains all lane tiles the front axle rectangle of an agent has an IoU with. Identifies, that an agent can potentially be located at precisely that lane tile of the map subgraph.
$\mathbf{P}_{\mathrm{Map}}$	Contains probabilities if an agent is on a certain lane tile at a certain time step in the future, before allowing agent interaction.
P	Probability that an agent is on a certain lane tile at a certain time step in the future.
$\boldsymbol{P}$	Contains probability that an agent is on a certain lane tile at a certain time step in the future.
$\boldsymbol{P}_{\mathrm{Final}}$	Contains probabilities if an agent is on a certain lane tile at a certain time step in the future, with agent interaction.
$\boldsymbol{P}_{\mathrm{GT}}$	Contains ground truth if an agent is on a certain lane tile at a certain time step in the future.
$\boldsymbol{P}_{\mathrm{Map}}$	Contains probabilities if an agent is on a certain lane tile at a certain time step in the future, before allowing agent interaction.
$\mathbf{Q}$	Laplacian matrix.
$\mathbf{R}$	Transition matrix from one lane tile to the lane tile on its right hand.
$\mathbf{S}$	Self message matrix.

$\mathbf{T}$	Transformation matrix.
$\mathbf{T}^{a,l}$	Transformation matrix from agent a to lane tile l.
$\mathbf{V}$	Matrix of eigen vectors.
W	Weight matrix entry, further defined through upper and/or lower indices. See Section 2.1.2.
$\mathbf{W}$	Weight matrix.
X	Input feature matrix entry, further defined through upper and/or lower indices. See Section 2.1.2.
$\mathbf{X}$	Input feature matrix.
Y	Output feature matrix entry, further defined through upper and/or lower indices. See Section 2.1.2.
$\mathbf{Y}$	Output feature matrix.
α	Aggregator function.
$\boldsymbol{\delta}$	Learning error.
$\boldsymbol{\delta}^{(\lambda)}$	Learning error of layer λ.
ϵ	Eigen value.
η	Learning rate.
λ	Layer index in an ANN.
μ	Mask.
μ_{C}	Conflict mask. 0 on its trace where agent a equals agent b, 1 otherwise.
μ_{Joint}	Joint probability loss mask.
μ_{TME}	Transition motion mask. 1 if edge from l_{S} to l_{R} exists, 0 otherwise.
μ_{Pos}	Position loss mask.
θ	Step function, Heaviside function .
ϑ	An angle.
$\vartheta^{a\rightarrow l}$	The angle between agent a's coordinate frame and lane tile l's coordinate frame.
φ	Artificial neural network module, i.e. function with trainable weights.
φ_{A}	Network module.
φ_{b}	Network module.
φ_{Joint}	Joint position probability generator, network module.
φ_{Conf}	Conflict Identifier, network module.
φ_{LTM}	Lane Tile Matcher, network module.

φ_{Mess}	Message Extractor, network module.
φ_{N}	Artificial neural network module, that processes on node-level.
φ_{NEN}	Artificial neural network module, that processes on a node-edge-node-level.
φ_{Notif}	Notification Extractor, network module.
$\varphi_{\text{S,t=1}}$	First state generator, network module.
$\varphi_{\text{S,t>1}}$	Other states generator, network module.
$\varphi_{\text{SelfMess}}$	Self-Message Extractor, network module.
φ_{TME}	Transition Motion Estimator, network module.
φ_{y}	Output module of a model.
σ	Activation function, non-linear if not stated otherwise.
τ	Transition probability from one to another lane tile.
$\tau_{\text{State},t}$	Transition probability for latent states from one to another lane tile.
$\tilde{\tau}_{\text{State},t}$	Preliminary transition probability for latent states from one to another lane tile.
$\tau_{\text{Pos},t}$	Transition probability for positional prob. mass from one to another lane tile.
$\tilde{\tau}_{\text{Pos},t}$	Preliminary transition probability for positional prob. mass from one to another lane tile.
ξ	Joint position probability in scalar form.
ψ	The conditional probability that an agent is on a certain lane tile at a certain time step given another agent on another lane tile, in scalar form.
ω	Turning rate.
Γ	Set of all lane tiles.
Θ	Set of all time steps.
Λ	Set of all agents.
$\mathbf{\Pi}$	Crossing matrix indicating spatial overlaps of two lane tiles.
Π	Crossing matrix entry indicating spatial overlaps of two lane tiles.
Υ	Set of lane tiles of a trajectory.
$\Phi_{\mathcal{M}}$	A prediction for a measurand.
Ξ	Joint position probability in tensor entry form.

$\boldsymbol{\Xi}$	Joint position probability in tensor form.
$\boldsymbol{\Xi}_{\text{GT}}$	Joint position probability ground truth in tensor form.
Ψ	Contains conditional probabilities that an agent is on a certain lane tile at a certain time step given another agent on another lane tile, in tensor entry form.
$\boldsymbol{\Psi}$	Contains conditional probabilities that an agent is on a certain lane tile at a certain time step given another agent on another lane tile, in tensor form.
Ω	An event. Accordingly, $\mathcal{P}(\Omega)$ notes the probability that the event is happening.
$\mathbf{0}$	Null tensor of equvalent shape (usually a vector).
$\mathbf{1}$	Tensor of equivalent shape filled with 1 (usually a vector).
$\mathbb{1}$	Unity matrix.
$\mathbb{D}$	Set of data pairs.
$\mathbb{N}$	Set of all natural numbers.
$\mathbb{R}$	Set of all real numbers.
$\mathbb{T}$	Set of transitions between one lane tile to another.
$\mathbb{Z}$	Set of all integers.
$\mathcal{E}$	Set of edges in a graph.
$\mathcal{E}_i$	Set of edges connected to node i.
$\boldsymbol{\mathcal{C}}_{\text{Joint}}$	Cross entropy tensor for joint probability.
$\mathcal{C}_{\text{Pos}}$	Cross entropy scalar for positional probability.
$\mathcal{G}$	Graph, consisting of vertices and edges connecting those vertices.
$\mathcal{L}$	Loss.
$\mathcal{L}_{\text{Pos}}$	Probabilistic position cross entropy loss.
$\mathcal{L}_{\text{Pos,Map}}$	Probabilistic position cross entropy loss of map-based prediction.
$\mathcal{L}_{\text{Pos,Final}}$	Probabilistic position cross entropy loss of final prediction.
$\mathcal{L}_{\text{Joint}}$	Joint probabilistic position cross entropy loss.
$\mathcal{L}_{\text{Sum,End}}$	Sum of losses of final position and joint position.
$\mathcal{L}_{\text{Sum,All}}$	Sum of losses of map-based position, final position and joint position.
$\mathcal{M}$	A measurand.

$\mathcal{N}$	Set of neighboring nodes of a node i.
$\mathcal{O}$	Computational complexity.
$\mathcal{P}$	Probability for a given event.
$\mathcal{T}$	A planned trajectory.
$\mathcal{V}$	Set of vertices in a graph.

1 Introduction

First, motivation for the presented research is described in Section 1.1. The goal of this work is described by defining requirements on traffic prediction Section 1.2, the contributions are summarized in Section 1.3 and prerequisites for the proposed prediction approach are formulated in Section 1.4. Last, an overview over chapters of this work is given in Section 1.5.

1.1 Motivation

Automated driving (AD) is expected to disrupt individual mobility and road traffic. The hypothetical benefits both for the individual traffic participant (TP) and society are manifold and are surveyed e.g. by Milakis *et al.* [76]. With advancing technology, advanced driver assistance systems (ADASs) and automated vehicles (AVs) will slowly penetrate the market with the result of challenging mixed human and robot traffic for years, maybe decades [81]. As long as there is mixed human and automated traffic, an AV requires some form of behavior prediction regarding other drivers' intentions in order to plan a trajectory for the AV, that allows it to fit in with human traffic[1]. The precise requirements towards behavior prediction are still subject to discussion among both planning and prediction researchers.

Automated systems consist of a complex sequence of functional modules. The output of a certain module is determined by the input requirements of the subsequent module. A module predicting TPs incorporates detections from perception modules, i.e. detected, tracked, and classified digital twins of TPs, that are called *agents*. Additionally, prediction requires a road topology and traffic rules that apply to the AV and other TPs which are included in a locally generated or globally available high definition (HD) map.

[1] In a fully automated environment AVs can share their trajectories or directly optimize them jointly.

The output of a prediction module serves as input to the two downstream stages of behavior generation and trajectory planning. In the former, high-level behavior decisions are generated, e.g. *let pedestrian cross the road, then drive on* or *change lane to the left before truck passes*. In the latter, an ego trajectory is created that breaks down the high-level behavior decision to a drivable trajectory. For generating reasonable behavior, estimations for future actions of other TPs are necessary, ideally conditioned on the ego vehicle's own possible actions.

The gap between perception and planning modules is filled by an extensive amount of prediction approaches for the field of traffic agent prediction in AD, many of which have been developed in recent years[2]. Before proposing a new solution, a collection of requirements for prediction in the context of automated driving is presented.

1.2 Requirements for Prediction

The content of this thesis was developed in close collaboration with the motion planning research team at Institut für Mess- und Regelungstechnik (MRT), which imposed the following requirement upon this work.

Uncertainty Estimation

Measurements are only complete and useful if an expressive uncertainty is added to the estimation [32]. The true value of the measurand $\mathcal{M}$, the *ground truth* (GT), is determined at time t_0 of measuring and remains unknown.

For prediction, this is not the case. At t_0, a prediction $\Phi_{\mathcal{M}}$ with prediction horizon Δt_{ph} is required referring to time $t_{\text{ph}} = t_0 + \Delta t_{\text{ph}}$. The prediction of a dynamic system is required Δt_{ph} before the state of a dynamic system is determined, and before a measurement can take place. During Δt_{ph}, arbitrary combinations of events or actions can happen that influence the system whose process variable is to be predicted. The longer the desired prediction horizon Δt_{ph} is, the more a – non-converging – system can potentially change. At t_{ph}, a measurement for $\mathcal{M}$ can be conducted with an uncertainty orders of magnitudes lower – meaning much more accurate – than what is achievable with a prediction $\Phi_{\mathcal{M}}$ at t_0.

[2] See Section 3.2.

One could argue that for developing prediction methods in AD, datasets are available for which GT was generated by recording traffic over time in the real world. Nevertheless, during inference in the field, the predictor has to generalize from those datasets in order to solve a prediction problem of a non-deterministic sociotechnical system and is therefore not regarded as a classical estimation problem. The generation of a meaningful uncertainty while improving accuracy for a position prediction is therefore regarded as the essential problem of prediction (see Section 2.4).

Multi-Modality

In order to increase accuracy while capture uncertainties, probability distributions with more than one local maximum need to be feasible. In traffic, there are often mutual exclusive maneuver options that are likely for an agent in a certain scenario. Consider e.g. a yielding scenario between two agents as in Fig. 1.1.

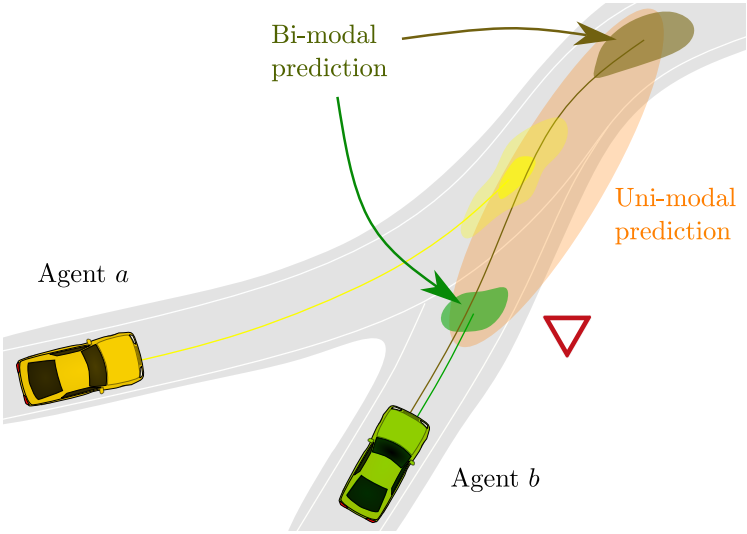


Figure 1.1: Agent a in a yielding scenario with agent b and their maneuver options. Since agent a is on the priority road, it will most likely choose to drive on. Agent b can choose between accelerating and merging before agent a passes, or brake and wait until agent a passed in order to merge behind it. If agent b choses a maneuver in between, it would risk a crash that agent a can avoid by resigning from its preferred behavior choice. Best viewed in color.

While the likely option of agent a on the priority road is driving on as desired, the yielding agent b can choose between two options: The first option is accelerating in order to pass the conflict zone without inducing a risk for agent a. The second option is to brake and continue merging after agent a. Of course, there are several reasonable trajectories that refer to each of those maneuvers. However, in between those two modes there are options of unlawfully cutting in agent b can take. Any of those options either requires a reaction of prioritized agent a in order to avoid a crash or leads to an unsafe situation[3].

Consider a uni-modal Gaussian position prediction with a single local maximum that is supposed to be shaped to cover both possible scenarios of agent b. Its maximum must be located between the two individual maneuvers modes (green and brown) and have a larger uncertainty, making the prediction misleading and uninformative for planning. In conclusion, a multi-modal representation for predictions is required.

Ability to Handle Different Traffic Scenes and Generalize Accordingly

In the great majority of traffic scenarios, vehicles follow lanes. Thus, an important prediction prior is given by the HD map, more precisely by the road markings and traffic rules it contains. A flexible approach is expected to handle an arbitrary number of map segments forming an arbitrary road topology, as well as an arbitrary number of interacting agents in the scene. For practical application, the number of agents covered by the sensor range of the AV forms the limit of what a model is required to robustly cover.

Also, an approach is expected to generalize from the limited set of available training examples to unknown intersection topologies in the test set and the real world.

Parameter Parsimony

Closely related to the ability to generalize is a parameter efficient model. If few examples are available for training, artificial neural networks (ANNs) with many parameters and especially convolutional neural networks (CNNs) working with highly redundant grid data are prone to overfit. High-level

[3] According to the definition of an unsafe situation in Shalev-Shwartz *et al.* [99].

features lead to a more compact representation, which in turn reduces the required number of trainable parameters in the model.

Computational Efficiency

For real time applications, computational efficiency needs to be considered when developing ANNs. Especially for hyperparameter tuning, models could theoretically be trained with infinitely many settings in order to fine-tune architectures. Conducting an extensive parameter study requires access to huge computational and therefore financial resources. By lowering the number of hyperparameters, it becomes realistic to test a reasonable amount of combinations in order to achieve satisfying results. In an automated mobile platform the computational resources are strictly limited compared to offline training clusters, especially since many modules of a software pipeline have to share this already limited amount of computational power and storage.

The challenging task therefore is to develop a model that can be fine-tuned to a reasonable state. Simultaneously, it shall be efficient enough for deployment on the limited resources of a mobile platform with an inference time small enough considering it being applicable in the AD processing pipeline.

Complete Input Information

There is no consensus in the research community regarding information necessary to produce a useful prediction, besides having more input information available leads to a more profound result than having less. Even though there are recent approaches that do not use a map topology at all [16, 68], it is assumed that prediction can greatly be improved with the use of maps.

Flexible Prediction Format

Most competing prediction approaches produce a position prediction in Cartesian coordinates that is independent of the map. While this is a useful format for trajectory planning, probabilities for future maneuvers are required for decision-making. A trivial solution for getting maneuver probabilities from e.g. a Gaussian mixture is integrating the distribution over regions that belong to a certain maneuver option. Such a region could be a parallel lane to identify lane changes or an intersection branch so identify turns. However, integration

might lead to degrading in probability calibration and therefore misleading uncertainties.

The output of the prediction model therefore has to bridge the gap between representing high-level behavior-classes for decision-making and representing future position that can be used for trajectory optimization.

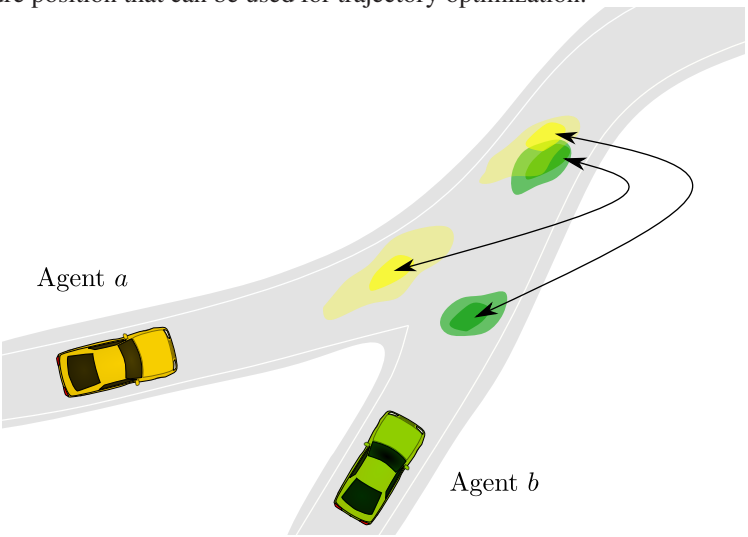


Figure 1.2: Agent b in a merging scenario with agent a and their two maneuver options: accelerating, or braking. The realized trajectories of both agents are not statistically independent. Each option of agent a can be conditioned to the contrary option of agent b and vice versa. In reality, the probability of a crash (both agents accelerate) should be much smaller than 25% (product of probabilities of the two modes corresponding to the acceleration maneuvers).

Conditional Prediction

Predictions are claimed to be usually multi-modal. Assume a merging situation symmetric with respect to priority as illustrated in Fig. 1.2. The acceleration maneuvers options – that are equally likely as the alternative braking maneuver – of two agents a and b are roughly at the same position. The probability for a crash is roughly $25\% = 50\% \cdot 50\%$ if statistical independence between the bi-modal prediction for each agent is assumed. However, in a scenario that requires interaction it must not be assumed that driving maneuvers are statistically independent without further investigation. Instead, the accelerating

maneuver of one agent likely corresponds to the braking maneuver of the other agent and vice versa. Two or more such corresponding maneuvers form an interaction mode. Predicting agent behavior conditioned on behavior others is called conditional prediction[4]. A conditional prediction is assumed to provide a strong advantage over predictions of statistically independent positions, since it allows planners to analyze likely future maneuvers of other agents conditioned on hypothetical future ego maneuvers.

1.3 Goal and Contributions

The goal of the presented approach is to implement a solution that fulfills the requirements listed above.

The main contributions of this work w.r.t. existing approaches are as follows:

1. **Explicit modeling of conflicts:** Information flow is strictly modeled in our network, so the network is trained to identify interpretable conflicts for each pair of agents. It is trained to solve those conflicts afterwards by exchanging interaction messages (see Section 5.2.3 and Section 5.2.5 for details). No other work was found that introduces interpretable conflicts with a comparable modeling depth (see related work of interaction modeling in Section 3.2.2).

2. **Traceable conditional predictions:** Besides the common predictions of statistically independent positions, the proposed model outputs conditional predictions that allow easy analysis of future behavior options for hypothetical future behavior of a selected vehicle, e.g. for the ego AV (see Section 3.2.3 and Section 5.2.8 for details).

3. **Architecture for predicting an agent graph on a topology graph:** A novel model architecture is presented that provides a solution for all tasks between perception and behavior generation in an AV stack. Furthermore, it is data and parameter efficient, real-time capable, and designed for real world application in structured road traffic. Even though it is designed for AD, it forms a general solution for probabilistic position prediction of moving primitives on a static graph topology.

[4] See Section 3.2.3 for conditional prediction approaches found in literature.

1.4 Prerequisites

In order to generate the best possible prediction, the following input data is a prerequisite of this work. For the presented approach to work as intended, availability of a HD map is assumed. It can either be generated during (mapless) driving or made accessible via a shared map regularly updated, e.g. by a fleet. This HD map must contain lane geometry and traffic rules. Furthermore, TPs whose behavior is to be predicted need to be detected, tracked, classified and localized in 2D relative to the HD map. The more accurate those prerequisites are fulfilled, the better the prediction will be.

1.5 Overview

In Chapter 2, fundamentals essential for understanding this work are briefly introduced with focus on the application of TP behavior prediction. The rise of graph neural networks (GNNs) in various research fields and recent contributions in the field of TP prediction are delineated in Chapter 3. Chapter 4 introduces basic design concepts utilized for designing the proposed model and gives an overview of the model's interfaces. In Chapter 5, the modules of the proposed approach and how they interact is described in detail. The model is evaluated in Chapter 6 w.r.t. prediction accuracy, probabilistic reliability, and functional claims formulated in the chapters before. The work is concluded in Chapter 7 by estimating the degree of fulfillment of the aforementioned requirements, limitations of the presented approach are discussed and recommendations are given on how to further develop the model.

2 Fundamentals

The theoretical fundamentals essential for understanding this work and deliberations it contains are presented in this section. First, notation conventions are described in Section 2.1. Graphs are introduced in Section 2.2, fundamentals of artificial neural networks (ANNs) and design concepts of graph neural networks (GNNs) are derived in Section 2.3. Last, fundamentals of forecasting research is outlined in Section 2.4.

2.1 Notation

Since notation varies, the use of references and footnotes, the mathematical notation as well as relevant technical terms are defined first.

2.1.1 References and Footnotes

Regarding references, the surname of the first author of a publication and surnames of coauthors are mentioned if equal contribution with the first author is claimed. If more than one work is referenced, or only exemplary work is referenced, naming the authors is omitted for better readability.

Footnotes are used for cross-references, for further explanation of statements the reader might not directly agree with, and for design recommendations for practical application.

2.1.2 Mathematical Notation

In this work, tensors are used up to the 5^{th} order. For better comprehension of an efficient model implementation, several mathematical notations are used. In the index notation – written in regular math mode x_i –, the lower indices denote the *inner* dimensions of this tensor. In this notation according to Einstein summation, indices occurring more than once denote multiplying and summing up, as if there was a sum symbol in front of the respective term

summing up over those indices. The opposite is the symbolic notation – written in bold math mode $\mathbf{x}$ –, where inner indices are not noted.

Also, element-wise notation is distinguished, where an individual element of the tensor is processed – written in letters referring to the element's order $\mathbf{x}^a$ –, and tensor notation, where all existing elements are concatenated into one tensor – written in letters referring to the tensor's order $\mathbf{X} = [\mathbf{x}^a]$. In the element-wise notation, upper indices denote the *outer* dimension, i.e. processing $\mathbf{x}^a$ is done for each agent a, so n_A times, if n_A is the number of agents.

An agent feature *tensor* with feature size n_F and number of agents n_A can be written in two ways $\mathbf{X} \equiv \mathrm{X}_{i,a} \in \mathbb{R}^{n_\mathrm{F} \times n_\mathrm{A}}$, the feature *element* vector of an individual agent can also be written in two ways $\mathbf{x}^a \equiv \mathrm{x}_i^a \in \mathbb{R}^{n_\mathrm{F}}$. Switching between Einstein notation and symbolic notation is marked with the equivalent sign $\equiv$. If not stated otherwise, the element-wise notation implicates "for all a", $\forall a \in \{1, ..., n_\mathrm{A}\}$. The identical mathematical operation – multiplication with weight matrix $\mathbf{W} \in \mathbb{R}^{n_{\mathrm{F,Y}} \times n_\mathrm{F}}$ – can be written in tensor formulation as

$$\mathrm{Y}_{j,a} = \sigma(\mathrm{W}_{j,i}\mathrm{X}_{i,a}) \equiv \mathbf{Y} = \sigma(\mathbf{W}\mathbf{X})$$

and in element-wise notation as

$$\mathrm{y}_j^a = \sigma(\mathrm{W}_{j,i}\mathrm{x}_i^a) \equiv \mathbf{y}^a = \sigma(\mathbf{W}\mathbf{x}^a).$$

On the left side of both equations, Einstein summation was used, on the right side, symbolic notation was used. In symbolic notation, the lower index is used for naming variables. For example a matrix referring to an output may be called $\mathbf{Y}_\mathrm{Output} \equiv \mathrm{Y}_{i,j}$. This naming index is removed when switching to Einstein notation, so Einstein notation is only used for better understanding of mathematical operations for tensors of higher orders.

Speaking upper and lower indices are used, so a, b refer to agent dimensions, l, k refer to lane tile dimensions, t, z refer to time step dimensions, and i, j, m refer to other – usually feature – dimensions. All kinds of index switching of tensors are indicated with $(\cdot)^\mathrm{T}$. The complete information on *how* indices have been switched is represented through speaking indices.

Unlike in physics, the word *tensor* is used without implying properties like rotation invariance for its features w.r.t. a coordinate frame, but it is used generally for an ordered mathematical tabular object of arbitrary dimension. By doing so, this work follows common nomenclature in computer science.

2.1.3 Naming Convention

An agent aims to reach a certain *destination* by driving a certain *route*, e.g. given from an automotive navigation system that is not lane precise. While following the route, the *path* contains precise future positions without knowing at which time the agent will pass a certain point. The path changes frequently depending on the *behavior* of other agents, while the route only changes due to large distortions like traffic jams. A *trajectory* connects path points with time stamps. An *intention* is a *maneuver* each agent has to estimate for other vehicles on a higher abstraction level compared to a trajectory. The feedback loop between intention estimation and behavior generation is called *interaction*. A behavior *mode* combines consistent intention options or maneuver options, respectively, for the ego vehicle.

People or robots participating in real world traffic are called *traffic participant* while they are called *agents* for the scientific problems within automated driving and its subfields.

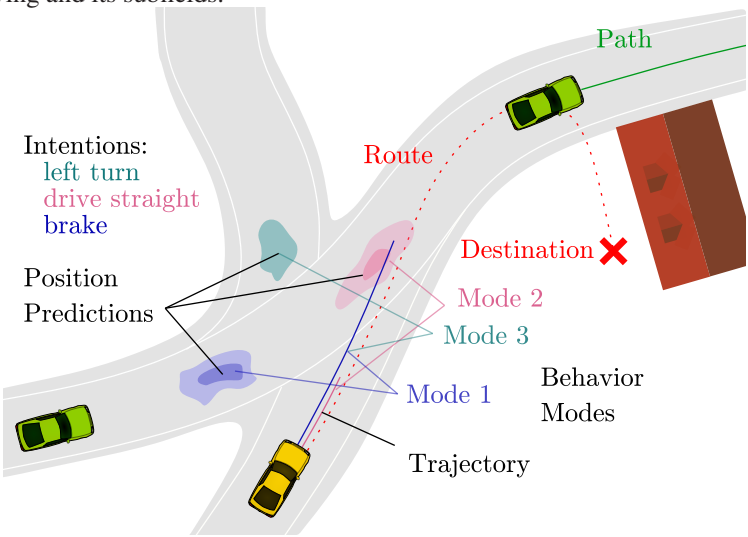


Figure 2.1: An example to illustrate the naming convention. The ego vehicle can choose between two trajectories blue and pink that both refer to a mode. The position predictions of other agents that are interacting with the ego vehicle correspond to one of these modes. High-level intentions can be assigned to position prediction modes. A route is the path to a destination. A trajectory consists of future positions corresponding to time steps. A path only consists of future positions.

2.2 Graph Theory

Fundamentals regarding graph theory can be found for example in Brandes [12], Tittmann [104] or Tutte [106].

An edge $\mathbf{e}^{i,j}$ connects exactly two nodes $\mathbf{n}^i$ and $\mathbf{n}^j$. Edges can be directed $\mathbf{e}^{i,j} \neq \mathbf{e}^{j,i}$ or undirected $\mathbf{e}_{i,j} = \mathbf{e}_{j,i}$. A graph is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $\mathcal{V} = \{\mathbf{n}^i\}, |\mathcal{V}| = n_\mathrm{N}$ denotes a set of n_N node feature elements $\mathbf{n}^i$ and $\mathcal{E} = \{\mathbf{e}^{i,j}\}, |\mathcal{E}| = n_\mathrm{E}$, denotes a set of n_E edge feature elements $\mathbf{e}^{i,j}$. For more expressive identification, edges are referred to by the indices of the nodes they connect. The word *element* refers to an arbitrary tuple of features that is usually written in vector or matrix form in context of ANNs.

In this work, two semantically different graphs – an agent graph and a map graph – are introduced. All agents form a set of agent graph nodes Λ, all lane tiles form a set of map graph nodes Γ. The agent graph is fully-connected the number of agent edges n_AE is equal to the squared number of agents $n_\mathrm{AE} = n_\mathrm{A}^2$. The directed map graph is sparse $n_\mathrm{LE} \ll n_\mathrm{L}^2$, so the number of lane segment connections n_LE is far smaller than the number of lane segments n_L^2.

All nodes that are connected to node i form a set $\mathcal{N} \subseteq \mathcal{V}$ and are called neighbors of node i. All edges that are connected to node i form a set $\mathcal{E}_i \subseteq \mathcal{E}$ and are called connections of node i. For directed graphs, the edges towards, or away from, or both towards and away from node i can be part of $\mathcal{E}_i$. This is further specified in the respective context.

Connections within a graph can be expressed as an adjacency matrix $\mathbf{A} \in \mathbb{Z}^{n_\mathrm{N} \times n_\mathrm{N}}$ where $\mathrm{A}_{i,j} \neq 0$ represents an edge from node j to i. In this work, $\mathbf{A} \in \{0, 1\}^{n_\mathrm{L} \times n_\mathrm{L}}$ holds.

2.3 Artificial Neural Networks

Fundamentals of ANNs can be found for example in Bishop *et al.* [10], LeCun *et al.* [56] or Hastie *et al.* [43].

2.3.1 Fundamentals

ANNs were created as a method to simulate neural processes similar to those in the human brain [67, 90]. The underlying idea is inspired the activation of a brain cell resulting from activations of other brain cells with a linear mapping of input activations $\mathbf{x}$ with weights $\mathbf{w}$. A non-linear function σ called *activation*

function is applied to the resulting scalar and the output y is called *activation* of the corresponding *neuron* and is calculated with

$$\mathrm{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}). \tag{2.1}$$

Fundamental properties of ANNs and how to optimize their weights by introducing the multi-layer perceptron (MLP) as the "universal approximator" [46] are briefly described. A MLP is the most generic type of ANN and is able to approximate every Borel measurable function arbitrarily accurate, if the number of hidden units is sufficiently large, as Hornik *et al.* [46] ascertained.

Multi-Layer Perceptron

If several neurons j with outputs $\mathrm{x}_j^{(\lambda)}$ are introduced, together, they build a *fully-connected layer* because every input scalar $\mathrm{x}_i^{(\lambda-1)}$ contributes to every output activation $\mathrm{x}_j^{(\lambda)}$. This layer can be written in compact vector form as

$$\mathbf{x}^{(\lambda)}(\mathbf{x}^{(\lambda-1)}) \equiv \mathrm{x}_j^{(\lambda)} = \sigma(\mathrm{W}_{j,i}\mathrm{x}_i^{(\lambda-1)}) \equiv \sigma(\underbrace{\mathbf{W}\mathbf{x}^{(\lambda-1)}}_{\tilde{\mathbf{x}}^{(\lambda-1)}}) \tag{2.2}$$

where the output vector $\mathbf{x}^{(\lambda)}$ is a function of the input vector $\mathbf{x}^{(\lambda-1)}$. The matrix product of weights and input is cumulated to $\tilde{\mathbf{x}}^{(\lambda-1)}$ and used later. It is possible to stack several layers to form a *model* that approximates the relationship of input and output data. A model consists of several *modules* and modules combine multiple *layers*. However, none of those terms are strictly defined. In this work, modules are logical units with a defined purpose.

A model formed from several fully-connected layers is called multi-layer perceptron (MLP) [43, 79]. Input and output are indexed with $\lambda \in [1, ..., n_\lambda]$. The result $\mathbf{x}^{(\lambda)}$ of such a neural model or ANN that is supposed to solve a certain task, can then be compared to a target vector $\mathbf{t}$ of same size, in this example with

$$\mathcal{L} = \frac{1}{2}(\mathbf{x}^{(\lambda)}(\mathbf{x}^{(\lambda-1)}) - \mathbf{t})^2 \tag{2.3}$$

resulting in a scalar *loss* term $\mathcal{L}$. Note that the layer's output is always a function of the layer's input; for simplicity, only the last dependency is explicitly noted. The goal of such a model is to minimize this loss given a data set $\{\mathbb{D}\}_{1:n_{\mathrm{Data}}}$ of pairs $\mathbb{D} = \{\mathbf{x}, \mathbf{t}\}$. The adjustment of weights with the aim of minimizing the loss is done with a local optimization algorithm called *error backpropagation*.

Error Backpropagation

The weights of a model are adjusted in a data-driven optimization process, often called *training*, with a weight difference $\Delta\mathbf{W}$ in the direction of loss decrease by

$$\Delta\mathbf{W} = -\eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}}. \tag{2.4}$$

The scalar η is called *learning rate* and is a hyperparameter for adjusting and stabilizing the learning process. According to the chain rule of differentiation first used for a neural network by Werbos [110] and for a deep neural network by Rumelhart *et al.* [92], the derivative can be split in

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{W}} \tag{2.5}$$

where the first fraction is called *error*

$$\boldsymbol{\delta} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}}. \tag{2.6}$$

For a fully-connected layer,

$$\frac{\partial \tilde{\mathbf{x}}^{(\lambda)}}{\partial \mathbf{W}^{(\lambda)}} = \frac{\partial \mathbf{W}^{(\lambda)} \mathbf{x}^{(\lambda)}}{\partial \mathbf{W}^{(\lambda)}} = \mathbf{x}^{(\lambda)\mathrm{T}} \tag{2.7}$$

holds and the error $\boldsymbol{\delta}$ is

$$\begin{aligned}\boldsymbol{\delta}^{(\lambda)} &= \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}^{(\lambda)}} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}^{(\lambda+1)}} \frac{\partial \tilde{\mathbf{x}}^{(\lambda+1)}}{\partial \tilde{\mathbf{x}}^{(\lambda)}} \equiv \frac{\partial \mathcal{L}}{\partial \tilde{\mathrm{x}}_m^{(\lambda+1)}} \frac{\partial \tilde{\mathrm{x}}_m^{(\lambda+1)}}{\partial \tilde{\mathrm{x}}_{\underline{j}}^{(\lambda)}} = \delta_m^{(\lambda+1)} \mathrm{W}_{m\underline{j}}^{(\lambda+1)} \frac{\partial \sigma\left(\tilde{\mathrm{x}}_{\underline{j}}^{(\lambda)}\right)}{\partial \tilde{\mathrm{x}}_{\underline{j}}^{(\lambda)}} \\ &= \delta_m^{(\lambda+1)} \mathrm{W}_{m,\underline{j}}^{(\lambda+1)} \sigma'_{\underline{j}}\left(\tilde{\mathrm{x}}_{\underline{j}}^{(\lambda)}\right) \equiv \sigma'\left(\tilde{\mathbf{x}}^{(\lambda)}\right) \odot \left(\mathbf{W}^{(\lambda+1)\mathrm{T}} \boldsymbol{\delta}^{(\lambda+1)}\right).\end{aligned} \tag{2.8}$$

Underlining an index – which is only used here – in the Einstein notation means there is no sum over that index, only multiplication. Hence, $\boldsymbol{\delta}^{(\lambda)}$ depends on the error of the consecutive layer $\boldsymbol{\delta}^{(\lambda+1)}$. For the weights of the last trainable layer before the loss function Eq. (2.3), the loss gradient can be calculated with

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(\lambda)}} \frac{\partial \mathbf{x}^{(\lambda)}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(\lambda)}} \frac{\partial \sigma\left(\mathbf{W}\mathbf{x}^{(\lambda-1)}\right)}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(\lambda)}} \frac{\partial \sigma\left(\mathbf{W}\mathbf{x}^{(\lambda-1)}\right)}{\partial \mathbf{W}}$$
$$= \underbrace{\left(\mathbf{x}^{(\lambda)} - \mathbf{t}\right) \odot \sigma'\left(\tilde{\mathbf{x}}^{(\lambda-1)}\right)}_{\boldsymbol{\delta}^{(n_\lambda)}} \mathbf{x}^{(\lambda-1)\mathrm{T}} \tag{2.9}$$

In conclusion, the weights of a hidden layer are adjusted with

$$\Delta \mathbf{W}^{(\lambda)} = -\eta \left(\sigma'\left(\tilde{\mathbf{x}}^{(\lambda)}\right) \odot \mathbf{W}^{(\lambda+1)\mathrm{T}} \boldsymbol{\delta}^{(\lambda+1)}\right) \mathbf{x}^{(\lambda)\mathrm{T}}. \tag{2.10}$$

The result in Eq. (2.10) has the following implications:

- The weights of layer λ can be optimized if the error $\boldsymbol{\delta}^{(\lambda+1)}$ of the subsequent layer is known. An optimization run is therefore conducted starting from the last layer and ending at the first. Therefore, the process is called error backpropagation.
- The inputs $\mathbf{x}$ and the derivatives $\sigma'(\tilde{\mathbf{x}})$ needed for optimization need to be stored during inference for backpropagation, making training of ANNs expensive w.r.t. memory.
- For MLPs, the matrix equation is comparably simple. However, matrix form of input and output is not required for optimizing a model with error backpropagation.
- There are mainly two requirements to make error backpropagation applicable. First, mathematical operations used to construct layers and modules must be differentiable w.r.t. their input and their weights. That also holds for activation functions σ[1]. Second, each of the following conditions leads to vanishing weight change and therefore stop the training process:
  - Inputs $\mathbf{x}$ are zero. Usually, this is not a problem, since training data is diverse and non-zero.

[1] If an activation function is discontinuous at some point – e.g. the famous ReLU function [64] –, desired derivatives can be set manually.

- Weights $\mathbf{W}$ are zero. It can be avoided by initializing weights randomly. Sometimes measures have to be taken to avoid zero weights during training.
- Derivative of activation function σ' is zero. This is avoided by using activation functions with a gradient of 1 for large intervals[2]. Also, initializing weights so that activations are in a region of large gradients is a convenient counter measure.
- The error obtained from the subsequent layer $\boldsymbol{\delta}^{(\lambda)}$ is zero. This is dependent on later layers w.r.t. the conditions listed above and can therefore accumulate throughout backpropagation for $-1 < \boldsymbol{\delta}^{(\lambda)} < 1$. This is called the "vanishing gradient problem" and is subject of analyses, especially for recurrent neural networks (RNNs) [44].

The downside of a MLP is the large number of weights per layer, making its optimization costly and prone to overfitting for a small dataset.

Recurrent Neural Networks

Unlike MLPs, neurons of a recurrent neural networks (RNNs) have an internal memory aside the current input which has shown to be useful especially for sequential models[3]. In the simplest configuration, a neuron can access its previous output

$$\mathbf{y}_{t+1} = \varphi(\mathbf{x}, \mathbf{y}_t). \tag{2.11}$$

However, this simple model has shown to be prone to the vanishing gradient problem which is why more sophisticated models have been promposed. Prominent examples are the long short-term memory (LSTM)-cell [44] and the gated recurrent unit (GRU)-cell [24].

[2] Most popular activation functions fulfill $\sigma(x) = 1 \forall x > 0$, see e.g. Kiliçarslan *et al.* [51] for a good overview, or the comprehensive analysis by Lederer [57].

[3] Some of the first applications of RNNs were developed by Servan-Schreiber *et al.* [98] and Kamijo *et al.* [49].

The LSTM cell used in this work has the following form

$$\begin{aligned}
\mathbf{i}_t &= \varphi_{\text{i,SIG}}(\mathbf{x}_t, \mathbf{o}_{t-1}, \mathbf{h}_{t-1}) \\
\mathbf{f}_t &= \varphi_{\text{f,SIG}}(\mathbf{x}_t, \mathbf{o}_{t-1}, \mathbf{h}_{t-1}) \\
\mathbf{g}_t &= \varphi_{\text{g,SIG}}(\mathbf{x}_t, \mathbf{o}_{t-1}, \mathbf{h}_{t-1}) \\
\tilde{\mathbf{h}}_t &= \varphi_{\text{IN}}(\mathbf{x}_t, \mathbf{o}_{t-1}, \mathbf{h}_{t-1}) \\
\mathbf{h}_t &= \sigma_{\text{UPDATE,SIG}}(\mathbf{f}_t \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{h}}_t) \\
\mathbf{o}_t &= \sigma_{\text{OUT}}(\mathbf{h}_t) \odot \mathbf{g}_t .
\end{aligned} \tag{2.12}$$

It consists of four MLPs producing activations for the input- $\mathbf{i}_t$, the forget- $\mathbf{f}_t$, the output-gate $\mathbf{g}_t$ and potential storing information $\tilde{\mathbf{h}}_t$. The gates' final activation functions are sigmoids $\text{SIG} : \mathbb{R} \to [0, 1]$, indicated by the lower index SIG. Each of those gates can learn to activate (open) or deactivate (close) – including all states in between – information flow towards or from the states. The LSTM cell produces a hidden cell state $\mathbf{h}_t$ and an output state $\mathbf{o}_t$ for each inference step.

2.3.2 Graph Neural Networks

The history of GNN applications is described in Section 3.1 where some milestone publications are listed. The last of those forms the basis of this subsection, a tutorial article by Battaglia *et al.* [7]. It describes a framework called "the graph network" and is claimed to be flexible enough to harmonize at least seven existing GNN architectures. Since the concept of GNNs is not well-defined, rules noted in this work may not apply for every single specialized network that is called GNN by the respective authors. However, those rules are assumed to hold for the majority of existing and near future architectures.

While the input and output size is fixed for common MLPs and convolutional neural networks (CNNs), most recent GNNs are able to process an arbitrary graph w.r.t. the number of nodes and the number of edges. Unlike RNNs that can also handle sequences of arbitrary length, GNNs are order invariant w.r.t. the graph entities $\mathbf{f} \in \{\mathbf{n}^i, \mathbf{e}^{i,j}, \mathbf{g}\}$ with $\mathbf{g}$ being graph features. A GNN includes several ANNs, that run on node, edge or graph feature level, or on a combination of those, thus making the input feature size for each of those ANNs static. Battaglia *et al.* propose to introduce processing functions φ and

the input combinations they expect in order to describe an architecture. One of the simplest processing functions is

$$\mathbf{y}^i = \varphi_{\mathrm{N}}(\mathbf{n}^i) \tag{2.13}$$

that processes every node. One of the most common processing functions for feature combinations is

$$\mathbf{y}^{i,j} = \varphi_{\mathrm{NEN}}(\mathbf{n}^i, \mathbf{e}^{i,j}, \mathbf{n}^j) \tag{2.14}$$

that individually processes every pair of nodes $\mathbf{n}^i, \mathbf{n}^j$ that is connected by an edge $\mathbf{e}^{i,j}$.

Most of the modules of the model proposed in this work are processing functions defined on a set of agent-node, agent-edge, map-node, map-edge entities, or a combination of those. The focus of Section 5.2 where the proposed model is described therefore is on understanding the operation level of the described functions, and not on how precisely the trainable architecture of each module is built. The architecture of modules and their hyperparameters are described later in Appendix A.3.

The second function class Battaglia *et al.* [7] mention are aggregator functions α or simply *aggregators*. Within a GNN, the number of processed outputs $\mathbf{y}$ varies depending on the current graph. Since the number of trainable weights must not depend on the number of nodes or edges, there has to be a possibility of reducing an arbitrary number of features to a feature entity of known size. The output of such a function is expected to be invariant w.r.t. the order of the input graph entities. Two of the most common aggregators are the sum over neighboring (hidden)[4] node features

$$\mathbf{y}^i = \sum_{\mathbf{n}^j \in \mathcal{E}_i} \mathbf{n}^j, \tag{2.15}$$

and the mean over all (hidden) node features

$$\mathbf{y} = \frac{1}{n_{\mathrm{N}}} \sum_{i=1}^{n_{\mathrm{N}}} \mathbf{n}^i. \tag{2.16}$$

[4] It is more common to add up hidden node features than directly adding the input features of a node.

While the former is here formulated to keep information at the node level, the latter reduces all (hidden) node features to a graph-level feature. Aggregators consist of an arbitrary combination of order invariant and differentiable functions and do not have trainable weights[5]. In this work, max-aggregators are used in the Notification Extractor (Section 5.2.4) and the Self-Message Extractor (Section 5.2.6), sum-aggregators are used in the Message Aggregator (Section 5.2.5), and weighted mean-aggregators are used for propagating states in the Predictor (Section 5.2.7).

The last important design element are connection matrices such as the adjacency matrix, Laplacian matrix or matrix of eigenvectors of the Laplacian matrix. Those quantities can be used as separate inputs for the network. They can be processed with e.g. positional one-hot vectors $\mathbf{p} \in \{0,1\}^{n_\mathrm{N}}, \sum_j \mathrm{p}_j = 1$ for node i. With the adjacency matrix $\mathbf{A} \in \{0,1\}^{n_\mathrm{N} \times n_\mathrm{N}}$, e.g. neighboring nodes can be identified through a many-hot vector $\mathbf{p}_{\mathcal{E}_i} \in \{0,1\}^{n_\mathrm{N}}$ with

$$\mathbf{p}_{\mathcal{E}_i} = \theta(\mathbf{A}\mathbf{p}^i). \tag{2.17}$$

This many-hot vector can then be used as a mask to identify only the node features of the neighbors of node i.

In conclusion, GNNs allow to process non-equidistantly structured data of arbitrary size in an order invariant manner. With those design patterns at hand, the proposed model and its modules can be created.

2.4 Prediction Theory

One of the oldest scientific fields with application of prediction is meteorology. Getting in touch with short-term – weather – forecasts cannot be avoided when consuming daily news, and after decades of increasing significance, long-term – climate – forecasts receive broad media attention as well.

Consequently, the earliest scientists active in the field of prediction as a demarcated subject within statistics used meteorological forecast as their primary example of application.

In one of those early publications about prediction, Murphy and Winkler [78] test subjective uncertainty of human forecasters for reliability with a dataset

[5] Of course one can directly include an update function running on the aggregator's output, but that is considered a separate function and not part of the aggregator.

collected after the event subject to prediction had occurred, by means of which ground truth (GT) was created. In consequence, Dawid [29] states the necessity of a prediction to be *well-calibrated* as a minimal desirable property. After many earlier publications of his research group, Gneiting together with Katzfuss [38] published summarizing work that can be seen as a general basis for probabilistic forecasting and its evaluation. They conclude their work with a set of universal rules for predictions:

- "Probabilistic forecasts aim to maximize their sharpness, subject to calibration. Calibration concerns the statistical compatibility between the probabilistic forecasts and the realized observations; sharpness refers to the concentration of the predictive distributions and thus is a property exclusive to the forecasts." [38]
- Specialized error functions which they call *scoring rules* assess both calibration and sharpness.

Despite it being a separate field of research, scoring rules as introduced by Gneiting and his group are not commonly used in automated driving. During research regarding prediction and prediction metrics, the conclusion is drawn, that there are significant flaws regarding evaluation of predictions [48, 111]. The requirements for a good prediction – denoted by Gneiting and his group, and summarized above – are more expressive than the prediction metrics commonly used for automated driving; a conclusion Zernetsch *et al.* [118, 119] share.

Consequently, metrics for evaluation are presented in Chapter 6 that pay respect to this paradigm.

3 Related Work

This chapter focuses on explaining the design concepts of graph neural networks (GNNs) that demarcate them from other popular artificial neural network (ANN) model classes by briefly tracing back to its origins in Section 3.1. Afterwards, recent competing approaches for traffic participant (TP) prediction are summarized with respect to key design features in Section 3.2.

3.1 Graph Neural Networks and their Evolution in Applied Research

After the success of deep artificial neural networks (ANNs) for technical application with high-dimensional structured data, the concept of graph neural networks (GNNs) has recently regained attention. Like many terms used self-evidently in the field of machine learning, the term GNN dates back over a decade, when it was first used by Gori *et al.* [41]. Prior to this, recursive neural networks encoding graph structures had been developed for directed acyclic graphs, e.g. by Goller and Küchler [39], Starita *et al.* [102], or Gori *et al.* [40]. Bianchini *et al.* [8] propose a method for processing cyclic graphs recursively with a stop condition. The architecture of Gori *et al.* [41] is the first to process arbitrary graphs without simplification. Therefore, their work can be seen as the foundation of today's graph neural network architectures.

Their proposed architecture consists of two layer types, a neighborhood processing layer φ_N and an output layer φ_y. First, φ_N defines an update function for every node i's state by

$$\mathbf{s}_{t+1}^{i} = \varphi_\mathrm{N}(\mathbf{n}^{i}, \mathbf{s}_{t}^{\mathcal{E}_i}, \mathbf{n}^{\mathcal{E}_i}) := \sigma\left(\sum_{\mathbf{n}^{j} \in \mathcal{E}_i} \varphi_\mathrm{A}(\mathbf{n}^{i}, \mathbf{n}^{j}) \mathbf{s}_{t}^{j} + \varphi_\mathrm{b}(\mathbf{n}^{i}) \right) \tag{3.1}$$

which creates the next state $\mathbf{s}_{t+1}^{i}$ for every node i from given node features of this node $\mathbf{n}^{i}$, the previous states $\mathbf{s}_{t}^{\mathcal{E}_i}$ of neighboring nodes, and the features $\mathbf{n}_{\mathcal{E}_i}$ of neighboring nodes. It is proposed to be implemented as a sum over linear

combinations of neighboring nodes though a pair-wise node processing layer φ_A and the neighbor's state $\mathbf{s}_j$ biased with a multi-layer perceptron (MLP) φ_b processing the features of the node of interest. Second, φ_y defined by

$$\mathbf{y}_{t+1}^i = \varphi_y(\mathbf{n}^i, \mathbf{s}_{t+1}^i) \tag{3.2}$$

produces a node-wise output $\mathbf{y}_{t+1}^i$ from the node's features $\mathbf{n}^i$ and the updated state $\mathbf{s}_{t+1}^i$ from Eq. (3.1). As indicated by t and $t + 1$, respectively, in the formulas, the network's inference consists of processing Eq. (3.1) and Eq. (3.2) iteratively until the states of the nodes converge.

Finally, the quadratic error between the output features $\mathbf{y}^i$ and a target feature vector $\mathbf{t}^i$ for each node i is minimized. Gori *et al.* also propose to retrieve a graph-wise output instead of the described node-wise output by defining the output of a dedicated node as the output of the whole graph.

In this first simple GNN, several properties can be identified that still hold for many GNNs including our proposed model:

- Unlike multi-layer perceptrons, graph neural networks operate on related inputs and outputs of arbitrary size, e.g. on graphs or subgraphs with a variable number of nodes and edges. Prediction of chemical properties of molecules [37] is an example for processing individual, self-contained graphs. For traffic prediction, however, intersections used for training are subgraphs of the worldwide road network (see Section 3.2.1 for further explanation).
- A GNN consists of several ANNs that operate on quantities called nodes and edges of fixed size. The number of quantities, however, can be variable. The dimensions of weight tensors may only depend on fixed feature sizes, not on the number of nodes or edges.
- Functions are required that aggregate a feature sequence of variable length to a feature of known size. This resulting feature must be invariant regarding the order of the sequence. Such a function is called *aggregator* (see Section 2.3.2). It is implemented in Gori *et al.* [41] as the sum over all neighboring "contributions" in Eq. (3.1).
- Gori *et al.* mention the possibility to model directed graphs by adding the edges' direction $d^j \in \{0, 1\}, j \in \mathcal{E}_i$ to the neighbors' node features. Later Scarselli *et al.* [96] published a more general approach which takes

into account edge labels $\mathbf{e}^{j,i}$ as separate features changing Eq. (3.1) to Eq. (3.3)

$$\mathbf{s}_{t+1}^{i} = \varphi_{\mathrm{N}}(\mathbf{n}^{i}, \mathbf{s}_{t}^{\mathcal{E}_i}, \mathbf{n}^{\mathcal{E}_i}, \mathbf{e}^{\mathcal{E}_i}) := \sigma\left(\sum_{\mathbf{e}^{j,i} \in \mathcal{E}_i} \sum_{\mathbf{n}^{j} \in \mathcal{E}_i} \varphi_{\mathrm{A}}(\mathbf{n}^{i}, \mathbf{e}^{j,i}, \mathbf{n}^{j}) \mathbf{s}_{t}^{j} + \varphi_{\mathrm{b}}(\mathbf{n}^{i}) \right) \tag{3.3}$$

with φ_{A}, $\mathbf{e}^{\mathcal{E}_i}$ and $\mathbf{e}^{j,i}$, respectively, being the edge features from node i to its neighbors $\mathbf{n}^{j} \in \mathcal{E}_i$ and edge feature size $n_{\mathrm{F,E}}$.

The Revival

The work of Scarelli *et al.* did not find the attention their ideas deserved. Their numbers of citation took off after being mentioned within the recent revival phase of GNNs.

The work of Bruna *et al.* [13] and Defferrard *et al.* [30] extends processing graphs with neural networks to the spectral domain, making use of physics-motivated properties of a graph[1]. A good overview on the family of spectral graph neural networks can be found in Chen [23]. Despite strong efforts to make the parameter complexity independent of the number of nodes in the graph, Bruna *et al.* and Defferrard *et al.* restrict their problems to graphs of fixed input size, ignoring a major advantage of GNNs.

Battaglia *et al.* [6] and Kipf *et al.* [54] revive the idea of a flexible graph size, that is proposed in Scarelli *et al.* [96] and Micheli [75]. Battaglia *et al.* use a graph approach to learn dynamics of a set of geometric primitives of arbitrary size in the 2D space.

Kipf *et al.* present a layer model called graph convolutional network (GCN) defined by

$$\mathbf{X}^{(\lambda+1)} = \sigma(\underbrace{\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{\frac{1}{2}}}_{\mathbf{Q}} \mathbf{X}^{(\lambda)} \mathbf{W}) \tag{3.4}$$

with $\mathbf{X}^{(\lambda)} \in \mathbb{R}^{n_{\mathrm{N}} \times n_{\mathrm{F}}^{(\lambda)}}$, $W \in \mathbb{R}^{n_{\mathrm{F}}^{(\lambda)} \times n_{\mathrm{F}}^{(\lambda+1)}}$, λ being the layer index and $n_{\mathrm{F}}^{(\lambda)}$ the corresponding number of hidden features. Here, $\tilde{\mathbf{A}} = \mathbb{1} + \mathbf{A}$ and $\tilde{\mathrm{D}}_{i,i} = \sum_j \tilde{\mathrm{A}}_{i,j}$ are supposed to avoid close-to-zero eigenvalues compared to the strictly mathematically motivated form. Equation (3.4) allows information flow with the

[1] See paragraph on spectral domain of graphs in Appendix A.2.

same range as in Eq. (3.1), namely from one node to its neighbors per GCN layer. The authors evaluate their model for classifying nodes in a sparsely annotated graph. The annotated part of the graph can be seen as a subgraph. Their method works well under the assumption that connected nodes in a graph are likely to share the same label.

After Kipf *et al.* published their layer model, GNNs finally found a large audience in the pattern recognition community. Their key contributions compared to earlier work are the following:

- Equation (3.4) describes an extremely simple model whose weights are shared across all nodes.
- The model is directly defined on matrix level without relying on a domain-shift into the frequency domain as in Bruna *et al.* [13] and Defferrard *et al.* [30].
- They use an efficient sparse matrix multiplication, so the computational complexity shrinks from $\mathcal{O}(n_{\mathrm{N}}^2)$ to $\mathcal{O}(n_{\mathrm{E}})$.
- They show their model's data efficiency for large graphs by using – in the most minimalist case – only one labeled node per class.

From here on, many publications introduced a variety of GNN layers and modules for solving numerous practical and theoretical problems.

Up to this point, the term GNN was used for every node- and edge-based differentiable model whose parameters are optimized with backpropagation. Hence unlike the term convolutional neural network (CNN) that stands for weight-sharing filters on large equidistant grid data, GNNs denote a large variety of models for non-equidistant structured data.

Battaglia *et al.* [7] propose a unified framework for understanding, constructing, and describing recent and future GNN architectures. It formulates a high-level framework that is flexible enough to describe about eight prior GNN research articles of various fields, most of which are publications whose authors contributed as coauthors to Battaglia *et al.* [7].

We see this publication as a fundamental work for GNNs design.

3.2 Competing Prediction Approaches

There are two earlier surveys on motion prediction by Lefèvre *et al.* [59] and Sivaraman *et al.* [100], three recent surveys from Mozaffari *et al.* [77], Rudenko *et al.* [91], and Huang *et al.* [47]. Besides, Varadarajan *et al.* [107] contains

an excellent literature review comparable to survey papers. Since they already form an overview more complete as could be achieved within a couple of pages in this work, reference is made only to work closely related to specific comparable features of our approach.

In Section 1.2, the main requirements a traffic prediction approach has to fulfill are listed. From those, a subset of features is chosen that is seen as fundamental for designing a good prediction approach. As design features for comparison, map encoding and modeling, interaction modeling, and prediction modeling with focus on conditional prediction are chosen. For each design feature, competing approaches are briefly named and conclusion w.r.t. to the proposed approach are drawn.

3.2.1 Map Modeling

As mentioned in Section 1.2, there is barely any doubt in the research community that a map is highly beneficial for prediction. However, the methods of including maps in models spead widely and are summarized below.

Grid Map. Many reseachers utilize rasterized (top-view) grid maps for each map entity [9,18,21,27,28,28,34,45,52,58,66,82,83,94,101,103,115], close-to-sensor occupancy grid maps [14,20,88,109], or both [19,63,84,89,93,117] to represent environmental features. Li *et al.* [60] add a front camera image.

Khandelwal and Qi *et al.* [50] name disadvantages that match our concerns regarding map rasterization: First, the number of network parameters explodes when an early map encoder has to extract incorporated map information that is usually sparse. This results in increased demand for training examples in order to ensure generalization. Second, compared to maps represented as a graph, rasterized data is highly ambiguous which leads to additional storage consumption. Third, the output is not lane related, but needs to be matched to individual driving options or maneuvers in a post-processing step. Fourth, rasterization omits information of lane connectivity which forms an important prediction prior as Zeng *et al.* [116] correctly state.

Polylines. A less ambiguous way of representing map information is the use of map features as line string or polylines. Each polyline consists of a variable number of points in an ordered list as e.g. Khandelwal and Qi *et al.* [50] propose. A point feature includes at least the point's coordinates but can also contain further, e.g. semantic point features. Varadarajan *et al.* [107] propose that at least start point, end point and semantic class are required. Gau, Sun *et al.* [33], Gu *et al.* [42], and Zhao and Gao *et al.* [123] utilize vector sets with

semantic information, the former has proposed this idea initially. A polyline in this work is represented as several vectors – also with start point, end point, and semantic information – sharing the same polyline ID. The main disadvantage is that it requires a huge amount of training data in order to generalize[2]. Also, they do not encode relations between individual polylines, thus the network has to learn all relations within the map implicitly.

Large Lane Segments. Instead of basic geometrical primitives, a topological map can be utilized. Many popular automated driving (AD) datasets [11, 15, 22, 55, 120] provide a semantic map with relational information as proposed by Poggenhans *et al.* [85] with the Lanelet2 format.

Liang *et al.* [61] propose a lane graph representation similar to our own proposal that was adopted by Zeng *et al.* [116] and Deo *et al.* [31]. They define straight lane centerline segments as nodes. If a segment is accessible directly from a neighboring segment, both are connected with an edge and the edge belongs to one of four classes: predecessor, successor, left neighbor, right neighbor. Positions of agents and lane segments are defined in Cartesian coordinates. The matching of agent and lane segment is represented as a limited neighborhood where an agent is only connected to lanes within a certain radius. The edges encode two map features: the distance between neighboring lane segments and the connection class from the list above.

Gilles *et al.* [35, 36] utilize lane segments long enough to contain a curve. As in the approaches above, they utilize the relational information between those lane segments. In [35], they predict a discrete rectangular occupancy probability distribution for every lane segment and warp this individual fixed-size occupancy grid to each lane segment. In [36], the encoded graph features are subsequently decoded in a heat raster over the whole map.

Small Lane Segments. A compromise between using a Cartesian grid of the whole environment and using lane segments is a 1D rasterization along the centerline of a lane segment.

Ye *et al.* [114] compress the map by sampling points along the centerlines and process their spatial relationship with a network originally designed for point clouds.

[2] Gau, Sun *et al.* [33] note a performance decrease when switching from their internal dataset at Google to the publicly available Argoverse [22] dataset.

Discussion

Using lane segments as they are extracted directly from an high definition (HD) map has several advantages. It is computationally efficient and therefore resource-efficient. Most importantly, actions that are given as trajectories relative to a lane segment coordinate frame can often be associated with a high-level maneuver. For grid-based output or coordinates in a global coordinate frame, this association cannot be achieved immediately.

In this work, a compromise is applied between using lane segments as they are in the original map and dividing the world in an equidistant grid. The map graph is cut in small pieces called *lane tiles*. This way, finer nuances regarding different maneuver options can easily be extracted even if lane segments are comparably large (see Section 4.1).

3.2.2 Depth of Interaction Modeling

Varadarajan *et al.* [107] classified recent work regarding a set of reoccurring network modules. An excerpt of that list is taken and extended with recent articles in Table 3.1.

Table 3.1: Comparison of prediction model architectures. The upper part is an except from Varadarajan *et al.* [107]. The lower part is an own completion. raster=information in grid map, conv.=convolution, attn.=attention-based, Enc.=Encorder, Dec.=Decoder, transf.=transformer, msg.=message.

Method	Motion Enc.	Interactions	Decoder
CAM [82]	LSTM	attn.	LSTM
CoverNet [83]	raster	conv	lookup
DESIRE [58]	GRU	spatial pooling	GRU
DKM [27]	raster	CNN	CNN
IntentNet [19]	raster	CNN	conv
LaneGCN [62]	1D conv	GNN	MLP
MANTRA [66]	GRU	–	GRU
MFP [103]	GRU	RNN+attn.	GRU
MTP [31]	raster	CNN	conv
MultiPath [69]	raster	CNN	MLP

Multipath++ [107]	LSTM+MCG	LSTM + MCG	MCG
PLOP [14]	LSTM	CNN	MLP
PRANK [9]	raster	CNN	lookup
PRECOG [89]	GRU	multi-agent sim.	GRU
R2P2 [88]	GRU	–	GRU
RoadRules [45]	raster	CNN	LSTM
SpAGNN [17]	raster	GNN	MLP
TNT [124]	polyline	maxpool, attn.	MLP
Trajectron++ [94]	LSTM	RNNs+attn.	GRU
VectorNet [33]	polyline	maxpool,attn.	MLP
WIMP [50]	LSTM	GNN+attn.	LSTM
AgentFormer [115]	conv. AutoEnc.	agent-aware attn.	AutoDec.
DenseTNT [42]	polylines	maxpool,attn.	MLP
DSDNet [117]	raster	msg.-passing	MLP
GOHOME [35]	GNN	attn.	MLP
ILVM [18]	[3]	SpAGNN	[4]
LaneRCNN [116]	GNN	CNN & msg.-passing	MLP
LookOut [26]	VectorNet	msg.-passing	SpAGNN & MLP
SceneTransf. [80]	self-attn	transf. cross-attn.	MLP & self-attn.
THOMAS [36]	GNN	attn.	MLP
TPCN [114]	Voxel & Points	CNN	CNN

[3] CNN & MLP & GRU & msg.-passing
[4] MLP & SpAGNN & MLP

Table 3.2: Comparison of interaction modeling depth divided into three groups with increasing modeling effort from top to bottom.

Method	Depth of Interaction modeling
VectorNet [33]	GNN on vectorized input
TNT [124]	same as [33]
DenseTNT [42]	same as [33]
MTP [31]	CNN on raster input
IntentNet [19]	CNN on raster input
MultiPath [69]	CNN on raster input
PRANK [9]	CNN on raster input
CoverNet [83]	CNN on raster input
LookOut [26]	CNN on raster input
RoadRules [45]	CNN on raster input
ILVM [18]	CNN on raster input
Trajectron++ [94]	attention between agents
SceneTransf. [80]	factorized self-attention
PRECOG [89]	state sharing
MFP [103]	state sharing during RNN rollout
AgentFormer [115]	agent-aware attention.
TPCN [114]	PointNet++-based neighborhood processing [86]
WIMP [50]	social graph attention
Multipath++ [107]	agents can access neighbors' states
SpAGNN [17]	msg. passing from neighbors' states
DESIRE [58]	spatial pooling (comp. SocialLSTM [2])
DSDNet [117]	energy weighted msg.-passing
GOHOME [35]	self-attention among agents
THOMAS [36]	self-attention among agents
CAM [82]	different types of attention
LaneGCN [62]	map-based graph attention

LaneRCNN [116]	map-based graph attention

Only few recent architectures do not explicit model interaction [14, 66, 88]. In many architectures, information can potentially flow from any source to any target in the network. A good example is VectorNet [33], in which trajectory tiles and map entities are both encoded as vectors, followed by a GNN that runs on vector-level and has to learn all relations by itself. The user does not have a chance to deeply analyze individual modules and test for their proper functionality. This architecture is basically a black box from the user's point of view. It is refered to as *low modeling depth*.

In contrast, the term *high modeling depth* is introduced, for cases when information is already directed by the explicit model design. As an example, GOHOME [35] consists of individual modules designated for a certain task through its location within the model's information flow. As an example, an Agent2Agent-module established self-attention among agents on the scene in order to encode agent interactions[5].

Based on publications that have already been analyzed by the authors and our subjective assessment, recent work is sorted with respect to the modeling depth used in the interaction module in Table 3.2, with the least modeling depth at the top.

After sorting prediction approaches by interaction modeling depth, the resulting list naturally divided itself in three groups: The authors of the upper group do not explicitly model interaction. They create networks and rely on the networks to learn all patterns required for prediction of TP that are present in the given data.

The middle group executes interaction modeling to some extent. By design, those architectures do not grant access to other agents' states equally, but it is at least weighted, e.g. by the social distance.

In the bottom group, many design features direct the flow of information, especially regarding the agents' neighborhood on the map graph.

Discussion

Several advantages in modularizing the network architecture and particularly utilizing a high interaction modelling depth can be recognized:

[5] Of course, for the reader of an article, assessing the modeling depth to some extent is subjective.

- It is possible to strictly limit the information flow to make a module solve a distinct task within the prediction pipeline due to lack of distracting information. With that, an information processing pipeline can be designed following the human way of predicting as archetype.
- Often, ANNs are compared to a "black box" whose behavior cannot be properly analyzed[6]. A modularized architecture enables analysis of the correct execution within a distinct module.
- Modularization usually results in less trainable parameters, lowering data and computational requirements. Consequently, it leads to more robust results given the same amount of data.

This work aims to utilize a large modelling depth that tries to replicate the human way of predicting traffic scenarios (see Section 5.1). With this, interfaces in the model are revealed that allow for functional analysis of the model, if a dataset e.g. with labeled interaction classes or labeled conflict classes was given.

3.2.3 Cooccurrence Modeling

The last chosen key design feature focuses on the model output. As explained in Section 1.2, the assumption of predicted positions to be statistically independent is strongly violated for many challenging scenarios with interaction.

Predicting Agents Independently. Most competing approaches produce agent-centric predictions that cannot be related to each other, so it is unknown which trajectory $\mathcal{T}^a$ of agent a corresponds to trajectory $\mathcal{T}^b$ of agent b [9, 14, 17, 19, 21, 27, 28, 33–35, 45, 58, 62, 66, 83, 94, 107, 114–116, 124].

Despite producing agent-wise predictions, Khandelwal and Qi *et al.* [50] highlight the necessity to reason about hypothetical scenarios and evaluate their prediction model with artificially created agents and states that are inserted in the ego vehicle's desired path.

Ego Trajectory as Input. Salzmann, Ivanovic *et al.* [94] directly include the ego vehicle's planned trajectory into their model input. This way, a model has the chance to generalize the behavior of other agents conditioned on the ego vehicle's path. Similarily, Tolstaya *et al.* [105] propose a model called

[6] Explainable artificial intelligence (XAI) is a catchword that summarizes the research around making results of ANNs interpretable [95].

Conditional Behavior Prediction that considers a pair of two agents where the trajectory of one of the agents, the robot, is fixed, and the corresponding trajectory of the other vehicle is predicted.

Consistency in Behavior Modes. A recent development in traffic scene prediction is to not only predict position distributions for individual agents, but to predict a consistent evolution of the whole traffic scene, sometimes called *modes* [18, 82]. These describe all the distinct possible maneuvers of an agent, that usually strongly rely on the maneuvers of others, see Fig. 1.1 for an example of two merging agents.

Gilles *et al.* [36] propose a model that produces an occupancy heatmap for each agent and samples a fixed number of possible trajectory end points. A consecutive network then learns to produce consistent combinations of those end points as possible future scene evolutions and assigns a probability to each evolution.

Conditional Prediction. Rhinehart *et al.* [89] claim to have developed the first conditional forecasting model for AD. They developed a single agent generative model based on Rhinehart *et al.* [88] that predicts several agents' behavior given their earlier trajectories and a grid map. They do so by predicting all agents' behavior with a joint generative model, whose input is not only the past trajectories and the grid map, but also a random state sample from a constant Gaussian for each agent. By finding a certain set of states that represent the robot's desired future trajectory, the network can be fed again with Gaussian noise for the other agents' states while the robot's trajectory is given by the found states. The authors formulate an optimization problem that leads to the robot's states.

With a different method, Tang *et al.* [103] publish a generative model that is also able to produce different trajectory "rollouts" even for multiple given trajectories. By rolling out a joint recurrent neural network (RNN) whose output state are the future positions at the next time step, the output of the model is created. Since the output state is also fed back into the model, the state can be set manually during rollout for as many agents as desired. Furthermore, this method directly learns to assign class labels to the outcoming trajectories without the need of providing ground truth.

Extending Casas, Culino, Suo *et al.* [18], Cui, Casas, Sadet *et al.* [26] directly sample consistent behavior modes from their model similarly to Gilles *et al.* [36], and additionally propose a planning method that is able to handle their output.

The approach of Ngiam, Caine *et al.* [80] expects an arbitrary set of conditional trajectory waypoints and also output different modes with assigned probabilities consistent with those given waypoints.

Discussion

The academic focus of prediction is clearly shifting from an individual field of research with the objective of minimizing the average displacement error between prediction and ground truth (GT), to prediction output that can be put into perspective in the corresponding scenario regarding other agents' reactions. The focus now is on general interpretability and more specifically on usability for the successive planning module. While behavior modes are an important element for interpretable prediction, not all agents of a traffic scene are dependent on all other agents, therefore the number of required modes to catch all major behavior combinations might be a limiting factor already for a few agents.

In the proposed model, bivariate discrete position predictions are produces that allow for pairwise and conditional analysis of corresponding behavior between two agents (see Section 4.3). With this pairwise output, the behavior of more than two interacting agents can be analyzed regarding behavior consistency. The absence of behavior consistency is an indicator for a not trustworthy prediction and should result in conservative behavior generation.

4 Design Patterns and Model Interface

In this chapter, key design features of the proposed spatially discrete modeling approach are explained, and the interfaces – input and output – of the proposed model are defined and explained. The model itself is described in Chapter 5. In total, 12 input and 3 output variables illustrated in Fig. 4.1 will be introduced that are lucidly summarized in Appendix A.5.

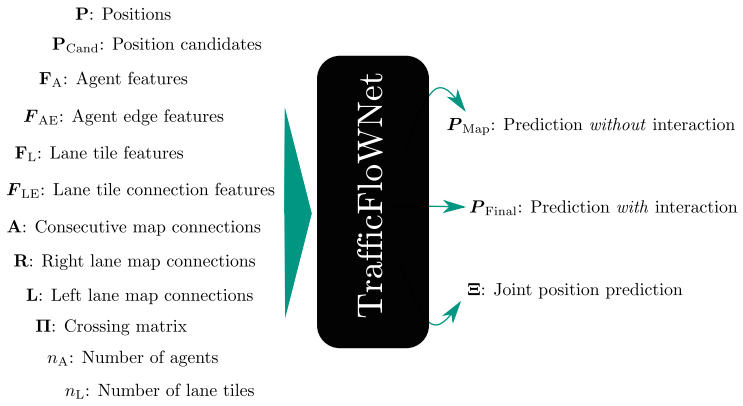


Figure 4.1: Input and output overview of the proposed model. Each variable is explained in the following.

Two Connected Graphs

One of the challenges the proposed approach tackles, is to combine two semantically different graphs in a differentiable and therefore trainable way: The

map subgraph[1] is extracted from the high definition (HD) map consisting of lane segments and their connections, and an interaction graph consisting of agents and their relations that evolve to interactions within the model.

A core design feature of the proposed model is the discrete spatial sectioning of the map graph in small *lane tiles*, so agents, time steps, and lane tiles can be represented in tensor form. This implies that core design features can be realized efficiently by tensor operations as explained later. The tensor indices $a, b \in [1, ..., n_{\mathrm{A}}]$ are used for agents, $l, k \in [1, ..., n_{\mathrm{L}}]$ are used for lane tiles, and $t, z \in [1, ..., n_{\mathrm{T}}]$ are used for time steps. Unlike lanelets, lane tiles have a minimum and maximum length. While lanelets can potentially have a length of 10 cm, e.g. as an artifact of automated map generation, or 10 km, e.g. on a highway, without violating model assumptions, the lane tiles have roughly the spatial dimensions that one car occupies laterally and longitudinally. Those bounds will be defined more precisely in Section 4.1. Agents are always bound to lane tiles in the model, unlike many other publications, where both agents and their prediction means are point masses in Cartesian coordinates. This discrete model approach allows efficient conflict representation between agents (see Section 4.1.1 and Section 4.1.4) and pairwise joint probability estimation (see Section 4.3) explained in Section 5.2.3 and in Section 5.2.8 in detail.

4.1 The Map Graph & Positioning of Agents

A high definition (HD) map in the context of automated driving consists of road entities such as road markings, traffic signs, traffic lights and the traffic rules they imply, both for lane-bound vehicles and vulnerable road users (VRUs). A map is the static model of an infrastructural environment. The environment changes over time, so the map has to be adapted to those changes through updates frequently. The map outdates e.g. by construction sites, police or fire brigade changing traffic rules or the road course. Such an incident happens either very slowly compared to overall traffic scene change, or it directly results in complete closure of a lane, road, or intersection. Those changes are considered as corner cases and their coverage is not the primary goal of

[1] As explained in Section 2.2, the worldwide road network can be seen as one large graph. From that graph only subgraphs can be used for training due to availability and performance reasons. Nevertheless, the term "graph" is used for simplicity when mentioning the map subgraph utilized in the model.

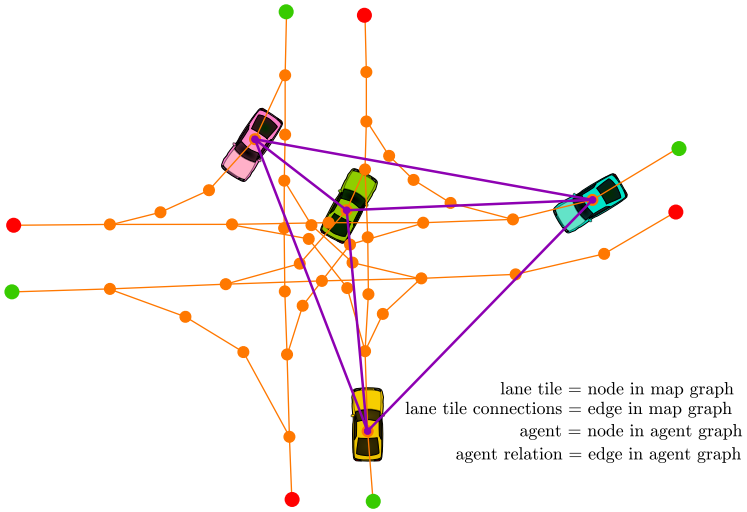


Figure 4.2: Visualization of how the model perceives agents and the map: The map graph is segmented into lane tiles. An agent's position on the map graph is defined by a probabilistic position vector over all lane tiles in the scene. In this image, the position vector for each agent is a one-hot vector assigning the agent to exactly one lane tile. The positional probability mass of agents can only be propagated along the map subgraph (orange) and leave the scene where the map graph was cut: At entry lane tiles (green) and exit lane tiles (red).

this work. However, most of them can be covered and solved by adjusting the map graph accordingly. Conclusively, the map can be assumed static from the viewpoint of a traffic participant at the time they pass through it and especially for the time span of Δt_{ph} without loss of generality.

In this work, availability of a lane-based topological map for automated driving is assumed where the lane width corresponds approximately to the lateral space a lane-bound vehicle covers, e.g. the framework Lanelet2 by Poggenhans *et al.* [85]. This map can be either given as a static road map, or a road topology generated while driving, e.g. [72–74]. In Lanelet2, a lanelet is defined as a lane segment between a left and a right road boundary, often directly given by a physical bound such as a curb stone or a road marking. Road infrastructure, e.g. signs or stop lines, can have a certain relation to individual lanelets, inducing traffic rules.

For the current use case, lanelets are split longitudinally in segments in such a way that one agent can roughly occupy one lane tile both longitudinally and laterally. Due to the variety of map topologies, a fixed size is not considered useful. This edit was done manually for ten intersections available in public drone datasets that are used for training and testing (see Section 6.1), for real world application it was automated. The resulting centerlines of lane tiles are between 2 m and 8 m of length, the majority between 3 m to 4 m. The original width is retained as provided by the map An example on how Lanelet2 maps look like for real world road topologies compared to the lane tiles is illustrated in Fig. 4.3.

4.1.1 Agent Positioning in the Map with a Position Vector

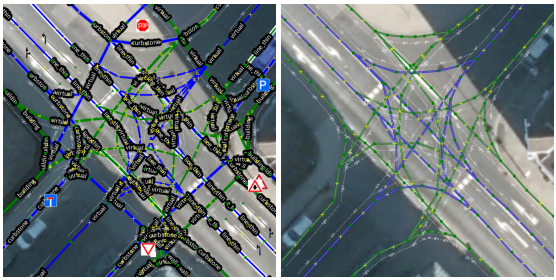
In order to establish a connection between agent and map graph, a probabilistic position vector $\mathbf{p}^{a,t} \in [0,1]^{n_\mathrm{L}}, \sum_l \mathrm{p}_l^{a,t} = 1$ is used to locate agent a at a certain time step t in the map subgraph on lane tile l. Regarding input and ground truth, a position vector entry $\mathrm{p}_l^{a,t}$ is 1 if a vehicle is located in the corresponding lane tile with absolute certainty l, and 0 if not. Since it is often ambiguous which lane tile is currently occupied by a certain agent, it is explained in Section 4.1.2 how input positions and ground truth (GT) are identified. Regarding the input of the proposed model, assuming perfect positioning of agents to lane tiles in a preceded perception module is unnecessary. A module is proposed later that assigns agents to lane tiles probabilistically which can be trained end-to-end with the rest of the model. Entries of a position vector within the model can be interpreted as a discrete occupancy probability distribution over lane tiles making the model's output representation a position vector with discrete probabilistic character.

Positional Conflict Identification through Matrix Multiplication

Next, the position vector's property to identify and represent conflicts is presented. Multiplying $\mathbf{p}^{a,t}$ with $\mathbf{p}^{b,t}$

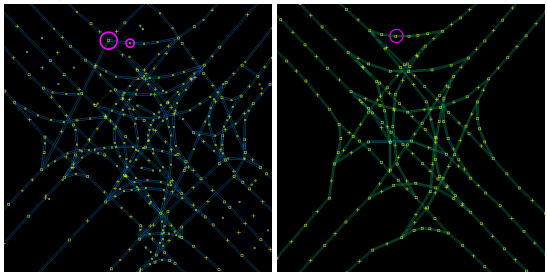
$$c^{a,b,t} = \mathrm{p}_l^{a,t}\mathrm{p}_l^{b,t} \equiv \mathbf{p}^{a,t\mathrm{T}}\mathbf{p}^{b,t} \tag{4.1}$$

results in $c^{a,b,t} \in [0,1]$ where $c^{a,b,t}$ denotes the probability of a position conflict between agent a and b at *the same* time step t assuming the predicted positions of agent a and b are statistically independent. In real world traffic,

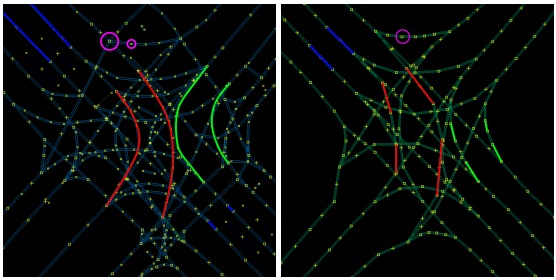


(a) Lanelet2 maps. While the original map contains rich features that also include sidewalks, polyline annotations and positions of road signs, only routing information, stop lines and priority signs are utilized for the needs of the proposed lane tile map.

(b) Aerial image of the exemplary intersection inD location1 by Bock *et al.* [11].



(c) Geometric map features. Lane tile boundaries (right) are given with a start and an end point (large squares). Lanet boundaries (left) can additionally have middle points (small squares) to form polylines.



(d) Left: Exemplary lanelets. Boundaries can be curved. Right: Exemplary lane tiles. Each lane tile is a quadrilateral.

Figure 4.3: An exemplary original Lanelet2 map (left) compared to a map with lane tiles (right). All common paths within an intersection are represented in the map. Source of aerial images: Esri World Imagery.

neither the actual behavior, nor position predictions of two agents with future position conflicts are statistically independent[2]. As illustrated in Fig. 1.2, overlapping positional predictions of different agents usually do not belong to the same behavior mode. The potential probabilistic character of $c^{a,b,t}$ therefore is only mentioned for completeness here but not further used or assumed in this work. Note that perfect predictions that are represented by one-hot position vectors with Eq. (4.1) rarely result in conflicts, since agents usually do not occupy the same lane tile[3]. Thus, the uncertainty of predictions would require a certain spread in order to identify conflicts at the same time step t.

As a result, it is claimed that combining position predictions of different agents *at two different* time steps t and z leads to a useful representation for identifying and interpreting position conflicts and their temporal relation for a pair of agents.

For this purpose, position vectors $\mathbf{p}^{a,t}$ of an agent a can be concatenated along the time dimension t leading to $\mathbf{P}^a \in [0,1]^{n_\mathrm{T} \times n_\mathrm{L}}$. Multiplying $\mathbf{P}^a$ with the position matrix of another agent b

$$\mathbf{C}_{\succ}^{a,b} \equiv \mathrm{C}_{t,z}^{a,b} = \mathrm{P}_{t,l}^{a} \mathrm{P}_{l,z}^{b} \equiv \mathbf{P}^a \mathbf{P}^{b\mathrm{T}} \tag{4.2}$$

results in conflict matrix $\mathbf{C}_{\succ}^{a,b} \in [0,1]^{n_\mathrm{T} \times n_\mathrm{T}}$ where $\mathrm{C}_{t,z}^{a,b}$ is the probability for a position conflict of agent a at time step t with agent b at time step z. This conflict matrix only identifies conflicts that happen on the same lane tile, e.g. because of merging or close following. This is indicated with the symbol $\succ$ which resembles two merging lanes. Another type of conflict can arise from the spatial crossing of two different lane tiles and is introduced in Section 4.1.4.

The different deliberations introduced in Eq. (4.1) and Eq. (4.2) are related through $c^{a,b,t} = \mathrm{C}_{t,t}^{a,b}$, so the first contains the trace entries of the second conflict object. The two different time indices indicate the hypothetical information incorporated in this tensor, unlike in Eq. (4.1), where behavior at the same time step is compared. After introducing the highly related crossing conflicts and the crossing matrix in Section 4.1.4, illustrative examples for basic conflicts will be given.

[2] Ideally, predictions are conditioned on (hypothetical) behavior. See Section 3.2.3 for details.

[3] A line tile was defined to be roughly the region an agent longitudinally and laterally can occupy.

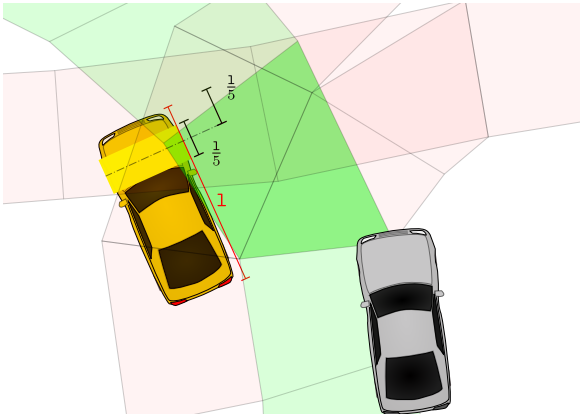


Figure 4.4: Identifying the correct lane tile an agent is currently occupying: First, route planning is conducted from the lane tile an agent a has entered the scene to the lane tile an agent leaves the scene. The resulting route lane tiles Υ^a form a track mask (light green). Current GT (intense green) is the lane tile with the largest IoU with the front axle rectangle (yellow) of an agent (orange). Only lane tiles that are part of the route are considered.

4.1.2 Position Identification and Training Target

Vehicles that are given in Cartesian space must be matched to discrete lane tiles. On the real world asphalted road surface, a vehicle can move independently of lane markings, traffic rules and other agents. The map graph that is used in this work is structured and can only be passed in certain directions, resulting in potential modeling mistakes, if agents are not located on the correct part of the map graph from the start. So for receiving accurate input positions and receiving GT future positions, the correct lane tiles need to be found for each agent at each time.

Generating Track Mask

The process of identifying GT lane tiles is illustrated in Fig. 4.4. First, the light green track of occupied lane tiles Υ^a is identified with route planning from a scene entry lane tile to a scene exit lane tile for each agent. Scene entry and exit lane tiles are identified for each agent by use of nearest neighbor assignment with the first and last position of an agent's trajectory in the dataset. This track is represented as a binary mask over all lane tiles. Only the lane

tiles that are part of that track are considered as input position or future GT position at a certain time step[4]. It is possible to construct examples in which route planning fails to create a good track mask[5]. Such events have not been specifically investigated, but it was not observed to be a frequent problem in the great amount of manually inspected scenarios. Also, despite a smooth training process being disturbed by such examples, the resulting predictions during inference are still useful. Since agents frequently leave the scene within the prediction horizon Δt_{ph}, it is generally necessary to handle vanishing GT position vectors $\mathbf{p}^{a,t} = \mathbf{0}$ correctly for certain a and t. The counter measures taken to handle both regular and rare cases of vanishing GT are explained in Section 5.4.1 and Section 5.4.2.

Assigning Agent to exactly one Lane Tile

If a track is identified for an agent, it is necessary to assign the correct lane tile at each time step for which the agent was recorded. Unlike other researchers and benchmarks, the center of a vehicle is chosen when reducing the bounding box of a vehicle to a point mass. Instead, the approximated position of the front axle center is used which is depicted yellow in Fig. 4.4, since it guides the movement of the vehicle along a trajectory[6]. It is assumed to be located at 20% length behind the front bound of the vehicle's bounding box[7], and laterally in the middle. The lane tile with the largest intersection over union (IoU) with the front axle rectangle located around the front axle center point with width w and length $\frac{\text{length}}{5}$ is chosen. By choosing a comparably short but wide bounding box, matches are created even if an agent drives at the very edge of a lane tile in order to pass a potentially blocked road. The front axle center point might then already be located outside the lane tile. Position vectors $\mathbf{p}^{a,t}$ of

[4] On roads with more than one lane in the same direction, all parallel lane tiles are part of that track mask.

[5] A U-turn is not covered that way since it is not considered for route planning. An agent going through a roundabout with a turn of more than a whole circuit also drives on lane tiles not included in the track mask since the track only contains the shortest path.

[6] The foremost point of a long vehicle might already be located outside the asphalted road in tight turns, being misleading for track identification. The front axle center point is guaranteed to be located on the drivable road.

[7] Smart EQ fortwo: 15.7% [70] Audi A1 (GB): 19.6% [3], Audi A3 (8Y): 20.2% [5], Audi A6 (C8): 18.6% [4], Mercedes-Benz S-Class (V223): 17.2% [71], VW Multivan (T7): 18.4% to 19.1% [108], MAN Lion's City 12: 22.8% [65]

all agents a can be concatenated to position matrices at a certain time step t to $[\mathbf{p}^{a,t}] = \mathbf{P}^t \in \mathbb{R}^{n_\mathrm{L} \times n_\mathrm{A}}$. And those matrices can be concatenated to position tensors $[\mathbf{P}^t] = \boldsymbol{P} \in \mathbb{R}^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A}}$. $\boldsymbol{P}_\mathrm{GT} \equiv P_{t,l,a}$ equals 1 for lane tile l with maximal IoU with front axle rectangle of agent a at time step t and lane tile $l \in \Upsilon^a$.

Mask and Features for Lane Tile Matching

Next, the position candidates matrix $\mathbf{P}_\mathrm{Cand} \in \{0,1\}^{n_\mathrm{L} \times n_\mathrm{A}}$ is introduced. An entry is 1, if the front axle rectangle (yellow) of agent a has a non-zero IoU with lane tile l, see Fig. 4.5. It is used as a mask to erase unlikely position matches of the Lane Tile Matcher introduced in Section 5.2.1.

With a local 2D coordinate frame for each agent front axle point and for each lane tile centerline start and end point, transformation matrices

$$\mathbf{T}^{a,l} = \begin{pmatrix} \cos(\vartheta^{a \to l}) & -\sin(\vartheta^{a \to l}) & x^{a,l} \\ \sin(\vartheta^{a \to l}) & \cos(\vartheta^{a \to l}) & y^{a,l} \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.3}$$

can be calculated. Transformation feature vectors

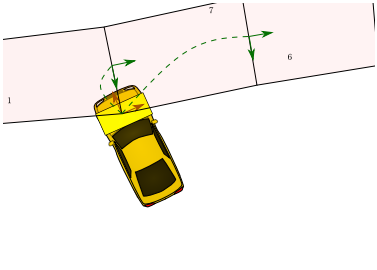
$$\tilde{\mathbf{t}}^{a,l} := [\cos(\vartheta^{a \to l}), \sin(\vartheta^{a \to l}), x^{a,l}_\mathrm{Start}, y^{a,l}_\mathrm{Start}, x^{a,l}_\mathrm{End}, y^{a,l}_\mathrm{End}] \tag{4.4}$$

that contain the information from Eq. (4.3)[8] can be extracted and are used as a further input feature for the model. With the transformation information and the position candidates mask, it is assumed that agents can be matched to lane tiles they are currently located on.

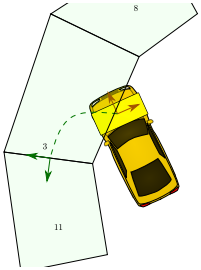
4.1.3 Modeling Movement Options along the Map Subgraph with Transition Matrices

In order to allow movement of agents with a position given by position vectors within the map subgraph, transition matrices are introduced. Each *transition matrix* $\mathbf{I} \in \mathbb{T} \subset \{0,1\}^{n_\mathrm{L} \times n_\mathrm{L}}$ represents a transition class leading from one lane tile to another. A *transition class* is the high-level movement option

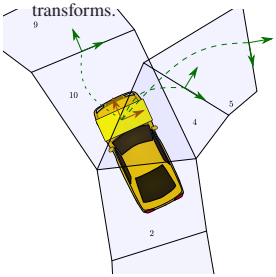
[8] Transformations to centerline start and end point have the same rotation, but different translational components.



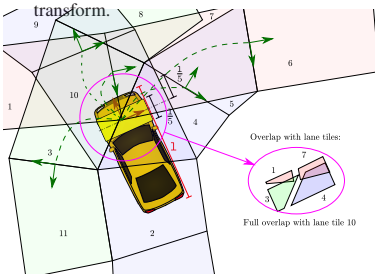
(a) Crossing lane tiles and their relative coordinate transforms.



(b) Opposite lane tiles and their relative coordinate transform.



(c) Ego lane tiles and their relative coordinate transforms.



(d) All lane tiles of a scene and overlapping areas with front axle rectangle.

Figure 4.5: Position candidates matrix is created by checking the overlap between the front axle rectangle of an agent and the area of the lane tile. Each lane tile with an overlap is considered as positional candidate. For better clarity, the ego lane tiles (blue), the opposite lane tiles (green) and the crossing lane tiles (red) are illustrated separately. The agent's coordinate frame is shown in brown, the lane tile end point coordinate frames are shown in green and the transform between those are indicated by dashed green lines. The overlapping regions are highlighted separately in the pink excerpt. For clarity, the lane tile start point coordinate frames are not visualized.

leading from one lane tile to another with the same behavior. E.g. the entries $I_{k,l}$ of the transition class "lane change to the left" connects lane tile k that is reachable through a lane change to the left starting from lane tile l. The resulting transition matrices are sparse. The transition classes between two lane tiles can be directly extracted from the map. The transition class from l

to k is never the same as from k to l. Lane tiles are connected with transition matrices, if they lead into the same direction[9].

Entry $A_{k,l}$ of matrix $\mathbf{A} \in \mathbb{T}$ denotes whether lane tile k is *directly consecutive* to lane tile l, meaning that an agent follows its current lane. The symbol $\mathbf{A}$ refers to the adjacency matrix used in graph theory, but unlike the original definition of adjacency matrices, each transition class has its own transition matrix in the proposed approach. Each lane tile can have several directly consecutive lane tiles creating motion options towards different directions. However, most have exactly one.

The purpose of a transition matrix $\mathbf{I}$ is to propagate positions through the directed road graph via the corresponding transition class by multiplying it with a position vector $\mathbf{p}^{a,t}$, e.g. the output

$$\mathbf{p}^{a,t+1} = \mathbf{I}\mathbf{p}^{a,t} \tag{4.5}$$

is a position vector after one discrete motion step through a certain transition class $\mathbf{I}$. In case of $\mathbf{I} := \mathbf{A}$, each occupancy probability would be transferred one lane tile forward in driving direction in the graph. Probability mass transitions are later[10] normalized over all transition options for each lane tile, so prob. mass[11] cannot leak the scene.

Besides driving to the consecutive lane tile, agents have the option to stay on their current lane tile. Therefore, the identity matrix $\mathbb{1}$ denotes connections of lane tiles with themselves. Each lane tile l can have up to one connected lane tile on the right and on the left that are reachable via lane change. Those lane changes are represented by transition matrices, to the right by $\mathbf{R}$ and to the left by $\mathbf{L}$. For many road topologies, $\mathbf{R} = \mathbf{L}^{\mathrm{T}}$ holds. However, there are map elements that allow lane change in one direction only, e.g. lane change from but not onto an acceleration lane when entering a highway. Note that all turns are included in transition class $\mathbf{A}$, so $\mathbf{R}$ and $\mathbf{L}$ only contain lane changes of parallel lanes in the same direction.

[9] However, the approach would allow to introduce a U-turn transition class whose matrix connects all pairs of lane tiles that share their left lane boundary (in right-hand traffic (RHT), in left-hand traffic (LHT) vice versa). Also, the approach is not strictly limited to automated driving (AD): Generally, it is the developer's task to introduce a meaningful set of transition classes and corresponding transition matrices $\mathbb{T}$ for each specific occupancy prediction problem on graphs.

[10] See Section 5.2.2 for details.

[11] Since the term *probability mass* is used a lot, the abbreviation *prob. mass* is utilized.

An agent on lane tile l can reach lane tile k through exactly one transition class[12]. In order to represent different velocities, especially if lane tiles are chosen to be smaller than the distance traveled with average speed Δt, combinations of the presented transition matrices are useful. In the present case of traffic scene prediction, the matrix products $\mathbf{LA}$, $\mathbf{RA}$ and $\mathbf{AA}$ are introduced as separate transition options. Driving in reverse can be represented with $\mathbf{A}^{\mathrm{T}}$, so in total eight transition classes are used in this work: $\mathbf{A}$, $\mathbb{1}$, $\mathbf{R}$, $\mathbf{L}$, $\mathbf{AA}$, $\mathbf{RA}$, $\mathbf{LA}$, $\mathbf{A}^{\mathrm{T}}$.

Similar to the beginning of this subsection, $\mathbf{I}$ is utilized subsequently as a placeholder for any of the eight previously introduced options. If $\mathbf{I}$ is used in a formula, in practice the formula is evaluated with all transition matrices.

4.1.4 Crossing Matrix for Spatial Overlap of Different Lane Tiles

Since Eq. (4.2) only produces conflicts for agents driving on the exact same lanelet, another matrix is required in order to complete the modeling of potential conflicts. Crossing matrix $\mathbf{\Pi} \in \{0,1\}^{n_{\mathrm{L}} \times n_{\mathrm{L}}}$ contains overlapping lane tiles where $\Pi_{k,l}$ denotes the IoU of lane tile l and k and therefore is symmetrical[13]. Multiplying the position matrix twice with the crossing matrix

$$\mathbf{C}_{\times}^{a,b} \equiv C_{t,z}^{a,b} = \mathrm{P}_{t,k}^{a} \Pi_{k,l} \mathrm{P}_{l,z}^{b} \equiv \mathbf{P}^{a\mathrm{T}} \mathbf{\Pi} \mathbf{P}^{b} \tag{4.6}$$

results in a crossing conflict matrix $\mathbf{C}_{\times}^{a,b} \in \mathbb{R}^{n_{\mathrm{T}} \times n_{\mathrm{T}}}$ between agent a and agent b.

Similar to the transition matrices in Section 4.1.3, more than one kind of crossing conflict can be introduced. It might be beneficial to handle crossing conflicts between cars separately from conflicts between cars and pedestrians. Since VRUs are not subject of this work, only one crossing conflict matrix is used.

[12] Note that most lane tile pairs are not directly connected at all. Many can be reached with sequences of transitions and some are not reachable at all.

[13] The trace entries are $\Pi_{l,l} = 1 \forall l$, so crossing conflicts would include merging conflicts (Section 4.1.1). In order to make distinction between crossing conflicts and merging conflicts simpler during training, the crossing matrix is chosen to be traceless.

4.1.5 Spatial Conflicts

Two types of conflicts have been introduced, merge conflicts in Eq. (4.2) where agents reach the same lane tile and cross conflicts in Eq. (4.6) where agents reach two overlapping lane tiles. A conflict $\mathbf{C} \in \{\mathbf{C}_{\succ}, \mathbf{C}_{\times}\}$ can be visualized as a gray scale image. In Fig. 4.6, $\mathbf{C}$ is visualized for some conflict primitives. They illustrate the interpretability of the proposed conflict representation.

4.1.6 Node and Edge Features of the Map Subgraph

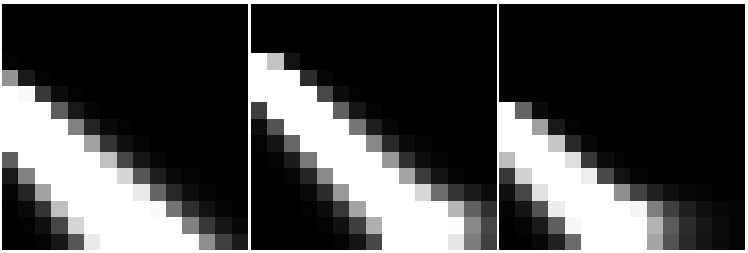
Feature matrices of lane tiles (nodes) $[\mathbf{f}^{l}] = \mathbf{F}_{\mathrm{L}} \in \mathbb{R}^{n_{\mathrm{L}} \times n_{\mathrm{F,L}}}$ and lane tile connections (edges) $[\mathbf{f}^{l_{\mathrm{R}}, l_{\mathrm{S}}}] = \boldsymbol{F}_{\mathrm{LE}} \in \mathbb{R}^{n_{\mathrm{L}} \times n_{\mathrm{L}} \times n_{\mathrm{F,LE}}}$ are given with feature sizes $n_{\mathrm{F,L}}$ for lane tiles and $n_{\mathrm{F,LE}}$ for lane tile connections.

Useful geometric features for lane tiles are shown written in red monospace font in Fig. 4.7. An additional non-geometric lane tile feature is the speed limit.

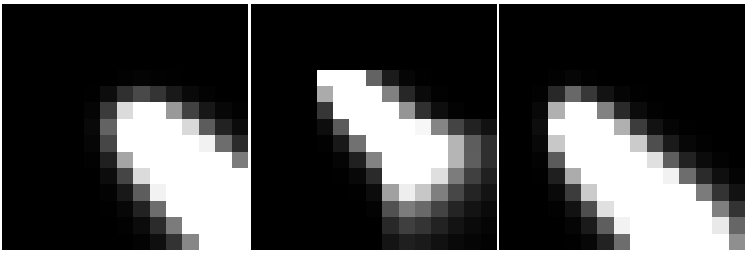
Useful geometric features for lane tile connections are written in green monospace font. Besides geometric features, a flag for identifying self-connection, a priority indicator and a stop line flag are used. The priority indicator has three states: +2 if there is a stop line between two lane tiles, +1 if there is a yield sign between two lane tiles, and 0 if there is no sign related to priority. The stop line flag is +1 if a stop line needs to be passed in order to reach the subsequent lane tile, and 0 if not. The advantage of using two separate indications here is that a priority to the right situation can be derived from the lack of priority signs and the existence of stop lines. All input features regarding nodes and edges of the map graph are listed in Appendix A.4.

One of the problems mentioned later is low diversity of map topologies identified as suitable for this work. If there were more maps available and the problem of overfitting to unique features was decreased, introducing more input features would be useful. For example, feature flags for lanes can be introduced: -2 if the agent is currently on a road where it can only turn left, +2 if the agent is currently on a road, where it can only turn right, ± 1 if also driving straight on is possible, and 0 if only driving straight on is allowed. With such a feature, model requirements regarding exploring the map and storing information during the map-based prediction phase are relaxed.

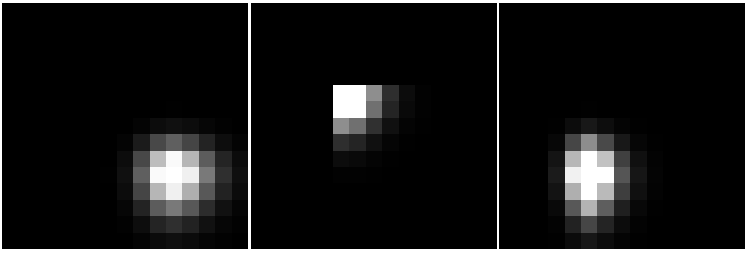
Another useful feature is the lane marking between parallel lanes. This feature is set to 0 if there is no lane marking that has to be crossed when using the corresponding connection. It is 1 if the lane marking is dashed,



(a) Conflict when following on the same lane. Left: slow speed, small initial distance. Middle: increased speed, small initial distance. Right: increased speed, initial distance increased. The temporal distance between two agents results in a shift of the conflict activations. The sharpness of the conflict activation region corresponds to the sharpness of the individual predictions.



(b) Conflict when two agents merge into the same lane. Left: slow speed, same distance to merge point. Middle: increased speed, same distance to merge point. The distribution fades out because the mode of both agents leaks out of the simulated scene. Right: slow speed, unequal distance to merge point. Merging looks like temporally dilated following, because there is a temporal distance from both agents to the conflict region.



(c) Conflict when two agents drive on two crossing roads. Left: slow speed, same distance to crossing point. Middle: increased speed, same distance to crossing point. Right: slow speed, unequal distance to crossing point. The conflict activation climaxes for a certain temporal combination t, z and fades afterwards.

Figure 4.6: Visualization of different basis conflicts $\mathbf{C} \equiv \mathrm{C}_{t,z} \in \mathbb{R}^{n_\mathrm{T} \times n_\mathrm{T}}$ between a pair of agents. The two time dimensions t, z start from the top left corner in each image.

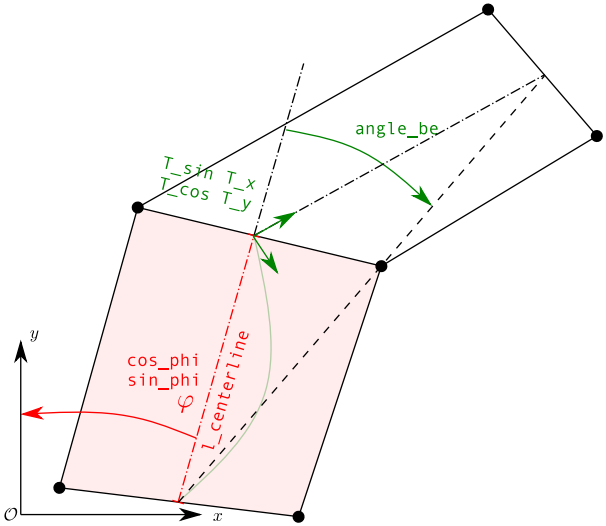


Figure 4.7: Features of lane tiles and lane tile connections. In greek letters, helper variables are visualized, in monospace font, the derived input features are noted. Those features are used not only for consecutive lane tile connections as one might assume from this schematic image but for connections in all transition classes.

which in Germany indicates that overtaking is allowed. It is 2 if it is solid, which indicates that overtaking is currently not allowed. Which this feature, the opposite lane can even be included in the map graph. Additionally, another lane feature flag can indicate whether the current lane tile is driven in the correct direction or in the wrong direction, e.g. during overtaking.

Furthermore, traffic light states can be added as connection features. However, intersections controlled by traffic lights are excluded in this work, because there is no corresponding dataset and because interaction is reduced to a minimum there.

4.2 The Interaction Graph

Agents can move in the scene obeying the map graph topology, its geometric properties and its traffic rules that contain information on who has to yield to whom in case of a conflict. Signs can be introduced through lane tile connection

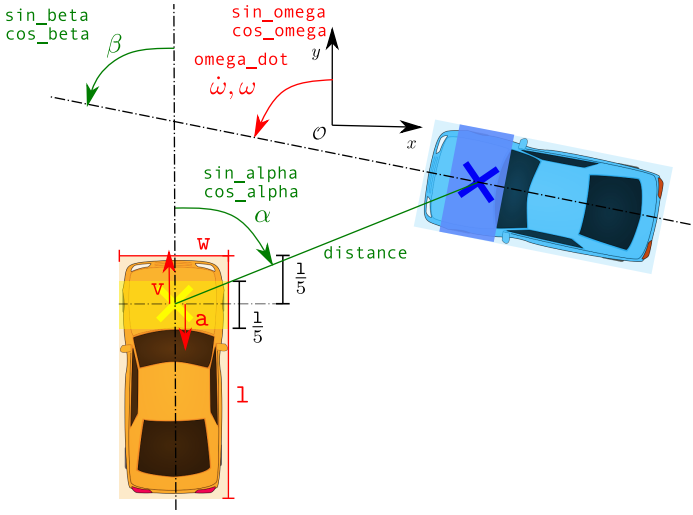


Figure 4.8: Agent node and agent edge features that are used for training and evaluation.

features, speed limits through lane tile features, and common general rules such as priority to the right or 4-way-stop behavior[14] shall be learnable by the model through behavior in the data and the geometric relations between agents at intersections that lack priority signs. Agents interpret those rules while considering other agents in the scene.

In the presented approach, agents are individual entities in the scene that get propagated through the map graph in order to minimize a positional classification loss at different prediction horizons. As described in Section 4.1.1, position vectors connect both graphs by defining the position of an agent on a certain lanelet.

Feature matrices of agents (nodes) $[\mathbf{f}^{a}] = \mathbf{F}_{\mathrm{A}} \in \mathbb{R}^{n_{\mathrm{A}} \times n_{\mathrm{F,A}}}$ and agent relations (edges) $[\mathbf{f}^{a,b}] = \boldsymbol{F}_{\mathrm{AE}} \in \mathbb{R}^{n_{\mathrm{A}} \times n_{\mathrm{A}} \times n_{\mathrm{F,AE}}}$ are given with feature sizes $n_{\mathrm{F,A}}$ for agents and $n_{\mathrm{F,AE}}$ for agent edges.

Useful geometric agent and agent edge features are described in Fig. 4.8. Additionally, the agent's class is indicated by an integer (truck/bus: -1, car/motorcycle: 0, bicycle: $+1$).

[14] Not used in this work, since the work is focused on European traffic.

Valuable further features that are not part of today's datasets is information about the indicator (indicating left: -1, indicator off: 0, indicating right: +1), the current stearing angle which is particularly interesting for agents standing at an intersection and a boolean flag about whether two agents are able to see each other. All features of agent graph nodes and edges are listed in Appendix A.4.

4.3 Output Definition

The network outputs discrete probabilistic position predictions over all lane tiles for each agent at each future time step $\boldsymbol{P} \in [0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A}}$ at two different stages within the model. The first prediction is done *without* allowing communication and interaction between agents, which is called map-based prediction $\boldsymbol{P}_{\mathrm{Map}}$. The second and final prediction is done *with* allowing communication and interaction between agents, which is called final prediction $\boldsymbol{P}_{\mathrm{Final}}$. $\boldsymbol{P}_{\mathrm{Map}}, \boldsymbol{P}_{\mathrm{Final}} \in [0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A}}$ holds. In contrast to most other approaches, the optimization task is formulated as a classification problem over the existing lane tiles for each agent at each future time step. For each agent a at each future time step t, a future position estimate is given by $\mathbf{p}^{a,t} \in \mathbb{R}^{n_\mathrm{L}}$ that fulfills $\sum_{l=1}^{n_\mathrm{L}} \mathrm{p}_l = 1, \mathrm{p}_l \geq 0 \forall l$[15].

With $\boldsymbol{P} \equiv P_{t,l,a}$, an estimate is given about how likely it is that agent a will be on lane tile l at time step t. In probability theory, this is called an *event* Ω. Either this event actually happens, or it does not. The probability of the event $\Omega_{t,l,a}$ is estimated. Thus, the estimate that this event happens is $P_{t,l,a} = \mathcal{P}(\Omega_{t,l,a})$.

Furthermore, the network outputs the conditional probabilistic position estimate for agent a and lane tile l at time step t given that agent b will be on lane tile k at the same time step t as a tensor $\boldsymbol{\Psi} \in [0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A} \times n_\mathrm{L} \times n_\mathrm{A}}$ where $\boldsymbol{\Psi}$ fulfills

$$\sum_{k=1}^{n_\mathrm{L}} \sum_{l=1}^{n_\mathrm{L}} \Psi_{t,l,a,k,b} = 1 \forall ((a,b,t) \wedge (a \neq b)), \tag{4.7}$$

which can be described as

$$\Psi_{t,l,a,k,b} = \mathcal{P}(\Omega_{t,l,a} \mid \Omega_{t,k,b}). \tag{4.8}$$

[15] A scene leaks prob. mass at the exits. Still the distribution is of probabilistic character.

Directly learnable is the joint probability

$$\Xi_{t,l,a,k,b} = \mathcal{P}(\Omega_{t,l,a} \wedge \Omega_{t,k,b}) \tag{4.9}$$

and the conditional probability $\mathcal{P}(\Omega_{t,l,a} | \Omega_{t,k,b})$ can be derived for a desired agent and lane tile specific condition $\mathcal{P}(\Omega_{t,k,b})$ with

$$\mathcal{P}(\Omega_{t,l,a} \mid \Omega_{t,k,b}) = \frac{\mathcal{P}(\Omega_{t,l,a} \wedge \Omega_{t,k,b})}{\mathcal{P}(\Omega_{t,k,b})}. \tag{4.10}$$

Since the result is a probability distribution over lane tiles l, instead of dividing by $\mathcal{P}(\Omega_{t,k,b})$, the l-dimension is normalized for a given agent b. The corresponding formulas leading to this conditional prediction can be found in Section 5.2.8.

5 Spatially Discrete Scene Prediction

The architecture of the proposed graph neural network (GNN) and its modules are explained in this chapter. Section 5.1 describes the model architecture briefly with analogy to the human driver. Section 5.2 describes the model in detail. In Section 5.3, the output of the proposed model is described and recommendations on how to post-process it are given. Section 5.4 the loss functions, their adaptations and different training strategies are discussed.

5.1 Model Overview

The proposed architecture consists of separate modules, each with a distinct task. Those tasks follow the human analogy of predicting other traffic participants (TPs):

1. **Assigning agents to lane tiles:** As explained in Section 4.1.2, it might be ambiguous which lane a TP is currently following in a structured road topology, given only a snapshot of the agent's state. Therefore, the **Lane Tile Matcher** (Section 5.2.1) is trained to assign agents to the lane tiles they currently drive on as a discrete probability distribution over all lane tiles that overlap with the front axle rectangle of the agent. The following modules can only propagate the positional prob. mass of the agents according to these matches.

   An experienced driver can accomplish that assignment without much reasoning based on the past observed states of another TP. Also, this experienced driver can directly identify other drivers for whom lane matching is not possible because their states completely contradicts the experience: for wrong-way drivers.

2. **Hypothetical prediction for exploration of the map:** If a single TP was alone in the world, its speed profile would be well-defined by personal

perception of safety, resource consumption and comfort requirements. So in multi-agent traffic, the main reason for speed adaptations is to resolve spatial conflicts with other TPs. Those conflicts can be solved according to priority indication, e.g. through traffic lights and signs, or spatial conflicts need to be resolved with cooperative behavior, especially in merging scenarios or in a hose[1]. In order to estimate future motion properly for all TPs in a complex traffic scene, position conflicts need to be identified first. In the **Map-based Predictor** (Section 5.2.7), each agent can explore the map graph along the possible drivable paths, storing information e.g. about the traversed route or the agent's current state. As the name implies, the Map-based Predictor can only make use of map features and features of the agent that is to be predicted.

For the analogy with human drivers, this corresponds to a foresighted driving style: Humans need to layout their own desired paths only given road topology and traffic rules.

3. **Identifying positional conflicts:** If the map-based predictions of two TPs occupy the same space at the same time, there is a potential conflict. The more potential conflicts for each pair exist, the more likely the necessity of a conflict solution through interaction becomes. By giving each conflict a numerical form, the **Conflict Identifier** (Section 5.2.3) can learn to identify recurring temporal conflict patterns that may be generalizable for various road topologies.

   A human driver foresees overlapping of the own desired trajectory with a potential trajectory of another TP at certain future time points requiring behavior adaptations in order to avoid risky situations.

4. **Sending gathered information to oneself and others:** After storing information about the passed route and identifying conflicts, the **Notification Generator** (Section 5.2.4) extracts data relevant for others and the **Message Generator** (Section 5.2.5) can produce an individual interaction message for each agent. Also, an agent can send information gathered during map-based prediction through the **Self-Message Generator** (Section 5.2.6) to itself. This information can contain upcoming

[1] In reality, those are not strict contradictions, but cooperative interaction frequently takes place in intersections with clear prioritization, too.

route options or traffic signs indicating traffic rules the agent needs to obey.

A human driver draws conclusions from the foreseen conflicts and the applying traffic rules. The driver then has the chance to adapt the initial motion intention accordingly.

5. **Final prediction with interaction:** Lastly, each agent is predicted along the map graph in the **Predictor** (Section 5.2.2) once more including interaction information the agent received from itself and from other agents.

   A human driver decides on a trajectory that both fits to the driver's *estimated intention of others* and *the path leading to its own destination* that each driver knows for him- or herself.

   Unlike for a human driver, the desired path of each agent is not utilized in the present approach. This introduces a large uncertainty about the evolvement of a scene and separates the research field of prediction from the research field of multi-agent planning.

6. **Conditional prediction:** Besides predicting distributions for statistically independent positions, the **Cooccurrence Estimator** (Section 5.2.8) predicts bivariate joint distributions for each pair of agents. With the definition of conditional probability, those joint distributions allow analyzing the traffic scene evolvement conditioned on hypothetical behavior of both the ego vehicle and other agents.

   A human driver would choose a (re)action and move forward faster if the driver did not have to consider several motion options of others but instead knew about their future paths. Therefore, humans intuitively condition their behavior on actions of others in order to avoid unlawful and risky situations.

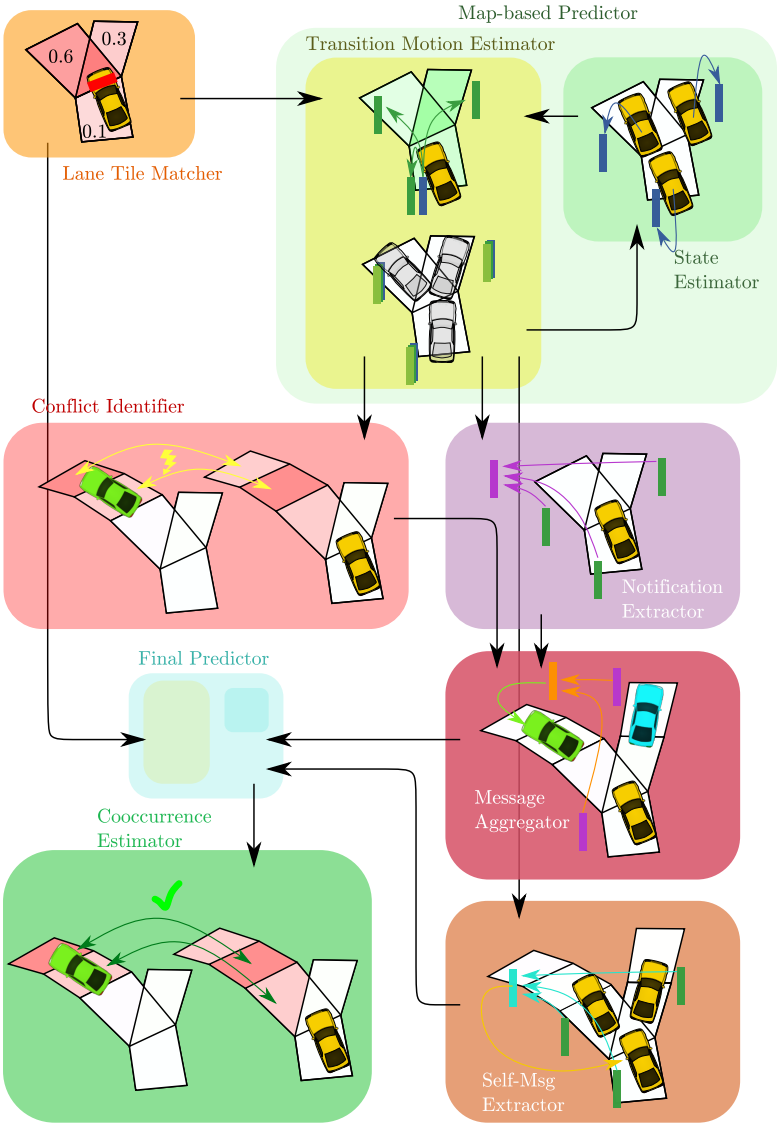

Map-based Predictor
Transition Motion Estimator
0.6
0.3
0.1
Lane Tile Matcher
State
Estimator
Conflict Identifier
Notification
Extractor
Final Predictor
Message
Aggregator
Cooccurrence
Estimator
Self-Msg
Extractor

Figure 5.1: Schematic overview of the proposed model architecture as described briefly in Section 5.1. Each pictogram represents a module described in the corresponding subsection of Section 5.2. An overview with formulas instead of pictograms can be found in Fig. 5.10. Arrows with one head represent information flow, arrows with two heads represent correspondences, vertical bars represent latent variables such as states, notifications and messages.

5.2 Model

The model and its modules are described in detail, leading from the schematic overview in Section 5.1 to the condensed overview with formulas in Fig. 5.10.

5.2.1 Lane Tile Matcher

In Section 4.1.2, difficulty of assessing which route on the map graph an agent is currently choosing was discussed, and on which lane tile it is driving, respectively. Therefore, assigning agents to lane tiles is regarded as a preconnected technical task that requires a separate module estimating the assignments. The *Lane Tile Matcher* is fed with dynamic information of the agent $\tilde{\mathbf{f}}^a = [\omega, v, \dot{v}]$ and its transformations to the lane tiles' centerline start and end point. After creating a local 2D coordinate frame both for each agent and for each lane tile entry, a transformation matrix Eq. (4.3) can be calculated for each pair of agent and lane tile. A multi-layer perceptron (MLP) φ_{LTM} creates a matching probability for all lane tiles $[\mathbf{p}^a_{\text{LTM}}] \in [0,1]^{n_{\text{L}} \times n_{\text{A}}}$

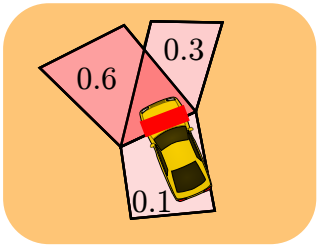


Figure 5.2: The **Lane Tile Matcher** assigns agents' initial positional probabilities to lane tiles.

$$
\begin{aligned}
\mathrm{p}^a_l &= \varphi_{\text{LTM}}\left(\tilde{\mathbf{f}}^a, \tilde{\mathbf{t}}^{a,l}\right), \\
\tilde{\mathbf{p}}^a_{\text{LTM}} &= [\mathrm{p}^a_l]_l, \\
\mathbf{p}^a_{\text{LTM}} &= \underset{l}{\text{norm}}\left(\mathbf{p}^a_{\text{Cand}} \odot \tilde{\mathbf{p}}^a_{\text{LTM}}\right), \qquad (5.1)
\end{aligned}
$$

where $\tilde{\mathbf{t}}^{a,l}$ contains the non-redundant transformation features from Eq. (4.4)[2]. The output of the MLP is multiplied with a mask of position candidates

[2] An illustration of the matching task is given in Fig. 4.5.

$\mathbf{p}^a_{\text{Cand}} \in [0,1]^{n_\text{L}}$ that is 1 if agent a and lane tile l have a non-vanishing intersection over union (IoU) and 0 if not. The remaining activations are positive and normalized, so the result satisfies probabilistic requirements and can be interpreted accordingly.
In Section 5.4.3, several variants of the proposed model and variants of proposed training procedures are explained. The Lane Tile Matcher is an optional module. Alternatively to including it in the proposed model, agent-to-lane tile matching can be done with an external preprocessing algorithm whose probabilistic result $\mathbf{p}^a$ can be fed into the proposed model as an input. Therefore, the Lane Tile Matcher can be regarded as a proposal to solve agent-to-lane tile matching which is directly integrable into the model, is trainable in an end-to-end fashion and lowers the prerequisite w.r.t. the input of the model, since agent-to-lane matching is not required from preceded perception modules.

5.2.2 Predictor

The *Predictor* is a module that is trained to produce a position probability tensor $\boldsymbol{P} \in \mathbb{R}^{n_\text{T} \times n_\text{L} \times n_\text{A}}$ where entry $P_{t,l,a}$ denotes the probability of agent a occupying lane tile l at time step t. Due to the discrete character of the presented method, the forward propagation is divided in n_T motion steps of duration $\Delta t = t_\text{ph}/n_\text{T}$. Two Predictors are used in the proposed model. First, the *Map-based Predictor* is used. It produces a position estimate solely based on map features and on individual agent features. Agents cannot communicate and therefore not interact with each other. With this map-based position prediction, potential conflicts are identified. Based on those conflicts, agents can then interact to solve them. Second, the *Final Predictor* is used. It is built up the same way as the Map-based Predictor, utilizes the messages sent between agents as a separate input and has individual weights.
Both Predictors contain two logical submodules – State Estimator and Transition Predictor – implemented by three artificial neural networks (ANNs): Initial State Estimator, Recurrent State Estimator, and Transition Predictor. Their tasks and their relations are explained in the following.

Initial State Estimator

Latent states $\mathbf{h}^{a,l}_t, \mathbf{o}^{a,l}_t$ are estimated for each combination of agent a and lane tile l. In the MLP $\varphi_{\text{S,t=1}}$, those are estimated utilizing agent features and lane tile features

$$\mathbf{h}_t^{a,l}, \mathbf{o}_t^{a,l} = \varphi_{\mathrm{S,t=1}}(\mathbf{f}^a, \mathbf{f}^l, \tilde{\mathbf{t}}^{a,l}). \tag{5.2}$$

The output state $\mathbf{o}_t^{a,l}$ is only utilized within the corresponding predictor it belongs to. The hidden state $\mathbf{h}_t^{a,l}$ collects information that is important for later modules.
The described states can contain latent information about the particular agent-lane tile combination, or on the other hand lane tile or agent features which the module has identified to be useful for further propagation. These states can also contain information on hypothetical previous or future motion intentions. As for all latent variables, to allow for precise interpretation of a state, their meaning needs to be investigated separately.

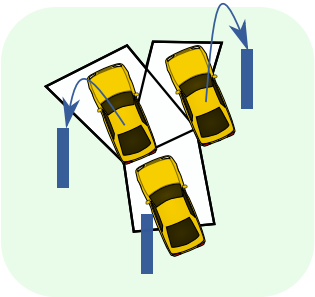


Figure 5.3: The **Initial State Estimator** creates states given agent and lane tile features, the **Recurrent State Estimator** later creates states given earlier states and lane tile features.

Transition Motion Estimator

A consecutive MLP φ_{TME} derives future transition motion probabilities given the output state $\mathbf{o}$ and edge features of the map graph

$$\begin{aligned}
\tilde{\tau}_{\mathrm{State},t}^{l_\mathrm{R},a,l_\mathrm{S}}, \tilde{\tau}_{\mathrm{Pos},t}^{l_\mathrm{R},a,l_\mathrm{S}} &= \mu_{\mathrm{TME}}^{l_\mathrm{R},l_\mathrm{S}} \cdot \varphi_{\mathrm{TME}}(\mathbf{o}_t^{a,l_\mathrm{S}}, \mathbf{f}^{l_\mathrm{R},l_\mathrm{S}}) \\
\tau_{\mathrm{Pos},t}^{l_\mathrm{R},a,l_\mathrm{S}} &= \underset{l_\mathrm{R}}{\mathrm{norm}}(\tilde{\tau}_{\mathrm{Pos},t}^{l_\mathrm{R},a,l_\mathrm{S}}) \\
\tau_{\mathrm{State},t}^{l_\mathrm{R},a,l_\mathrm{S}} &= \underset{l_\mathrm{R}}{\mathrm{norm}}(\tilde{\tau}_{\mathrm{State},t}^{l_\mathrm{R},a,l_\mathrm{S}}).
\end{aligned} \tag{5.3}$$

Generally, an agent a can be predicted from a sending lane tile l_S onto a receiving lane tile l_R via a certain transition motion type $\mathbf{I}$. The transition motion type is determined by the index pair $(l_\mathrm{R}, l_\mathrm{S})$, since two lane tiles are connected by exactly one edge and its corresponding motion type[3]. All prob.

[3] Since distinguishing between sender and receiver lane tile is important in the context of this subsection, l, k are exchanged here with speaking variables.

mass must be propagated from the sender lane tile l_S to one of its receiver lane tiles l_R. Therefore, prob. mass is normalized over the receiving lane tiles l_R, resulting in transition probabilities that can be interpreted as the probability of agents a leaving lane tile l_S towards lane tile l_R choosing the corresponding transition motion type **I**:

$$\tau_{\text{Pos},t}^{l_R,a,l_S} = \mathcal{P}(\Omega_{t+1,l_R,a} \mid \Omega_{t,l_S,a} = 1). \tag{5.4}$$

Two transition probabilities are distinguished. $\tau_{\text{State},t}^{l_R,a,l_S}$ is used for propagating latent states, $\tau_{\text{Pos},t}^{l_R,a,l_S}$ is used to propagate the positional prob. mass. Decoupling the propagation of latent states and position has shown to be useful. While the positions are optimized for estimating ground truth (GT) future positions, the states $\mathbf{h}, \mathbf{o}$ are reused to gather information during map graph exploration. The information in $\mathbf{h}$ is later shared between agents in order to improve the prediction and can therefore contain information gathered independently of the preliminary positions of an agent.

Now the initial occupancy probabilities – either given as input $\mathbf{p}_0^a := \mathbf{p}^a$ or generated by the Lane Tile Matcher $\mathbf{p}_0^a := \mathbf{p}_{\text{LTM}}^a$ – are multiplied with the transition probability $\tau_{\text{Pos},t}^{l_R,a,l_S}$. Finally, summing up positional prob. mass transferred to the same lane tile l_R over all sender lane tiles l_S contributing to l_R leads to

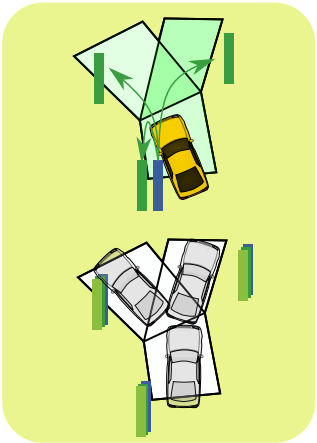


Figure 5.4: The **Transition Motion Estimator** predicts lane tile wise transition probabilities for both states and positional prob. mass given the output states. Transferred states and prob. mass leading to the same lane tile are then added in a weighted sum.

$$\mathbf{p}_t^a \equiv \mathrm{p}_{l_R}^a = \sum_{l_S} \tau_{\text{Pos},t}^{l_R,a,l_S} \cdot \underbrace{\mathrm{p}_{l_S}^a}_{\equiv \mathbf{p}_{t-1}^a}. \tag{5.5}$$

Also, latent states need to be propagated with the generated motion transitions. To do so, the motion transition probabilities $\tau_{\text{State},t}^{l_R,a,l_S}$ serve as weights to transfer shares of $\mathbf{o}_{t-1}^{a,l_S}$ to $\tilde{\mathbf{o}}_t^{a,l_R}$ according to

$$\tilde{\mathbf{o}}_t^{a,l_\mathrm{R}} = \frac{\sum_{l_\mathrm{S}} \tau_{\mathrm{State},t}^{l_\mathrm{R},a,l_\mathrm{S}} \cdot \mathbf{o}_{t-1}^{a,l_\mathrm{S}}}{\sum_{l_\mathrm{S}} \tau_{\mathrm{State},t}^{l_\mathrm{R},a,l_\mathrm{S}}}. \tag{5.6}$$

$\mathbf{h}_{t-1}^{a,l}$ is transferred accordingly. This allows the module to infer states and gather information for agent a on lane tile l_R independently of the probability for this agent being on said lane tile. With that, each new state is a weighted sum of other recent states' contributions and is therefore expected to contain reproducible information that can be further analyzed.
Next, it is described how a full prediction step is processed and how latent states and prob. mass can iteratively be propagated through the map graph.

Recurrent State Estimation and Transition Prediction

For motion steps beyond the first, the last states $\mathbf{o}_{t-1}^{a,l}, \mathbf{h}_{t-1}^{a,l}$ are utilized as input features instead of the agents' features for state estimation by $\varphi_{\mathrm{S,t>1}}$. A long short-term memory (LSTM)-cell called the *Recurrent State Estimator* $\varphi_{\mathrm{S,t>1}}$ can modify and propagate diverse states that all represent different combinations of motions options leading to a motion state.
Unlike the states of agents, map features are static, so they can be reused as input

$$\mathbf{h}_t^{a,l}, \mathbf{o}_t^{a,l} = \varphi_{\mathrm{S,t>1}}(\tilde{\mathbf{h}}_t^{a,l}, \tilde{\mathbf{o}}_t^{a,l}, \mathbf{f}^l). \tag{5.7}$$

Equation (5.3) to Eq. (5.7) can now be applied iteratively in order to obtain position matrices for all future time steps t.
After n_T time steps, all position vectors $\mathbf{p}^{a,l}$ can be concatenated resulting in the map-based position probability tensor $\boldsymbol{P}_\mathrm{Map} := [\mathbf{p}^{a,l}] \in [0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A}}$ described at the beginning of this module. Entry $P^{t,l,a}$ represents the occupancy probability of agent a on lane tile l at time step $t \in [\Delta t, 2\Delta t, ..., \Delta t_\mathrm{ph}]$, where Δt_ph denotes the maximum prediction horizon.
The same can be done with the hidden states $\boldsymbol{H} := [\mathbf{h}_t^{a,l}] \in \mathbb{R}^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A} \times n_\mathrm{F,S}}$ and with the output states $\boldsymbol{O} := [\mathbf{o}_t^{a,l}] \in \mathbb{R}^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A} \times n_\mathrm{F,S}}$.
The results of the Map-based Predictor can be interpreted in many ways. In general, the interpretation is dependent on whether $\boldsymbol{P}_\mathrm{Map}$ was used as a separate term for optimization, or only the final prediction $\boldsymbol{P}_\mathrm{Final}$ was used. If $\boldsymbol{P}_\mathrm{Map}$ is used for optimization, the distribution might be sharper. However, the states,

especially the output state $\boldsymbol{O}$, is then primarily used for predicting positions as precisely as possible. The hidden state $\boldsymbol{H}$, that is used in later modules, is depending on the output state as can be seen in the equation of the utilized LSTM-cell Eq. (2.12). It has to be evaluated experimentally whether those two states can diverge enough, so each serves its individual purpose sufficiently.

5.2.3 Conflict Identifier

Position conflicts and therefore potential collisions happen between agents a and b that have a non-vanishing probability of occupying the same area at the same point in time t. A measure for intensity and kind of position conflict between two agents can be obtained if multiplying the occupancy probabilities for two agents on the same lane tiles l at the same time step.
By multiplying the map-based position matrices $\mathbf{P}^{a}_{\text{Map}}$ of two non-identical agents a, b along their lane tile dimension for all time step combinations t, z according to

$$\mathbf{C}^{a,b}_{\succ} \equiv \mathrm{C}^{a,b}_{t,z} = \mathrm{P}^{a}_{t,l}\mathrm{P}^{b}_{l,z} \equiv \mathbf{P}^{a}_{\text{Map}}{\mathbf{P}^{b}_{\text{Map}}}^{\mathrm{T}} \tag{5.8}$$

a conflict matrix $\mathbf{C}^{a,b}_{\succ} \in [0,1]^{n_{\mathrm{T}} \times n_{\mathrm{T}}}$ is obtained. Each entry can be interpreted as the probability of agent a being on the same lane tile at time step t as agent b at time step z. This has the following implications. First, conflicts are only identified if these two agents are predicted to occupy the same lane tile. Therefore, this conflict matrix only identifies conflicts emerging from a merge conflict, a car-following situation, or a mixture of both, indicated visually by the index $\succ$ that visually resembles two merging lanes. Lane tiles that overlap but are not identical lead to spatial conflicts that are not observable in $\mathbf{C}_{\succ}$. Those conflicts will be handled separately. Second, even if the map-based prediction is as precise as the GT, there will still be non-vanishing conflicts, since predictions are cross-compared over all combinations of t and z for each pair of agents. If only positional conflicts at the same time step t were considered $\mathrm{P}^{a}_{t,l}\mathrm{P}^{b}_{l,t} \in \mathbb{R}$, conflicts would be less intense when predictions are more accurate.

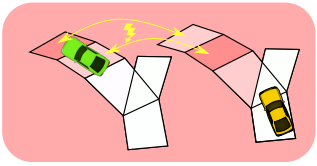


Figure 5.5: Merging and crossing conflicts are identified and classified in the **Conflict Identifier**.

The conflict calculation is implemented as

$$\boldsymbol{C}_{\succ}^{a,b} \equiv C_{t,z}^{a,b} = \mu_{\mathrm{C}}^{a,b} \mathrm{P}_{t,l}^{a} \mathrm{P}_{l,z}^{b} \equiv \mu_{\mathrm{C}}^{a,b} \mathbf{P}_{\mathrm{Map}} \mathbf{P}_{\mathrm{Map}}^{\mathrm{T}}. \tag{5.9}$$

The mask μ_{C} is 1, if $a \neq b$ and 0 otherwise. This way, irrelevant conflicts of an agent with itself are removed.

Often however, agents are not in conflict with each other because they potentially reach the same lane tile. From a graph perspective, they can reach two different lane tiles that occupy the same spatial area in the real world. Therefore, a second kind of conflict, namely crossing conflicts, are introduced. They are calculated by multiplying $\mathbf{P}_{\mathrm{Map}}^{a}$ with the crossing matrix $\boldsymbol{\Pi}$ from both sides along the lane tile axes

$$\mathbf{C}_{\times}^{a,b} \equiv \mathrm{C}_{t,z}^{a,b} = \mu_{\mathrm{C}}^{a,b} \mathrm{P}_{t,l}^{a} \Pi_{l,k} \mathrm{P}_{k,t}^{b} \equiv \mathbf{P}_{\mathrm{Map}}^{a} \boldsymbol{\Pi} \mathbf{P}_{\mathrm{Map}}^{b\ \mathrm{T}} \tag{5.10}$$

and receive the crossing conflict matrix $[\mathbf{C}_{\times}^{a,b}] =: \boldsymbol{C}_{\times} \in [0,1]^{n_{\mathrm{T}} \times n_{\mathrm{A}} \times n_{\mathrm{A}} \times n_{\mathrm{T}}}$ where an entry denotes at which motion steps t, z two agents a and b will have a conflict because they occupy lane tiles that overlap. As for $\boldsymbol{C}_{\succ}$, conflicts of an agent with itself are erased with a mask.

For each agent combination a, b a *conflict activation map* is given by $[\mathbf{C}_{\times}^{a,b}, \mathbf{C}_{\succ}^{a,b}] =: \mathbf{C}^{a,b} \in [0,1]^{n_{\mathrm{T}} \times n_{\mathrm{T}} \times 2}$. Those conflict activation maps can be interpreted visually as illustrated for all conflict primitives in Fig. 4.6[4]. With that, an interpretable and simple pairwise conflict representation is found.

A trainable module is introduced that can learn to interpret these maps and produce a useful conflict activation vector $\mathbf{c}^{a,b} \in \mathbb{R}^{n_{\mathrm{F,Conf}}}$ with

$$\mathbf{c}^{a,b} = \varphi_{\mathrm{Conf}}(\mathbf{C}^{a,b}). \tag{5.11}$$

Since $\mathbf{C}^{a,b}$ can be interpreted as an image with two channels – one for merging conflicts, one for crossing conflicts – with height and width being the time dimensions t and z, the natural choice for designing φ_{Conf} is a simple feature extraction convolutional neural network (CNN). The conflict activation map is symmetric w.r.t. a simultaneous switch of agent and time axes

$$C_{t,z}^{a,b} = C_{z,t}^{b,a} \neq C_{t,z}^{b,a}. \tag{5.12}$$

[4] Since the third dimension of the conflict activation map is small and fixed, it is renounced to use a tilted bold symbol.

Thus, in general, the conflict features are asymmetric $\mathbf{c}^{a,b} \neq \mathbf{c}^{b,a}$. This property is crucial, since conflicts in traffic are rarely symmetric with respect to geometry of the scene, time and priority. With the corresponding asymmetry, the module gains the possibility to learn to solve each conflict from the perspective of each individual agent.

5.2.4 Notification Extractor

Now that conflicts have been identified, information exchange between agents needs to be implemented in order to allow agents to solve their conflicts in the final prediction step. The hidden states $\boldsymbol{H}$ of the Map-based Predictor were already introduced in Section 5.2.2 and the model is trained to fill those states with information relevant in later modules[5]. The *Notification Extractor* utilizes the states to extract information each agent has collected during information propagation on the map graph. With the hidden states of each agent a, a *notification* is created with information that might be important for agent b that potentially has a conflict with agent a. This is implemented by an MLP φ_{Notif}

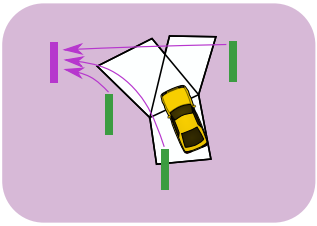


Figure 5.6: For each agent, the **Notification Extractor** produces notifications that contains information important for other agents.

$$\mathbf{u}^a = \max_{t,l}(\varphi_{\text{Notif}}(\boldsymbol{H}^a, \mathbf{P}^a_{\text{Map}})) \tag{5.13}$$

that processes all hidden states and the corresponding occupancy probability $[\boldsymbol{H}, \boldsymbol{P}_{\text{Map}}] \in \mathbb{R}^{n_{\text{A}} \times n_{\text{L}} \times n_{\text{T}} \times n_{\text{F,S}}+1}$ in case it is necessary to weight the relevance of the respective state. Each agent therefore produces a notification $\mathbf{u}^a \in \mathbb{R}^{n_{\text{F,Notif}}}$ with information relevant to others from all information collected over the whole prediction horizon. Since the goal is to condense all information in a single agent-specific vector, max-pooling is used over lane tile and time dimension. Note that those notifications are not addressed to specific agents, but only created from specific agents. Addressing notifications target-specific is executed in the next module.

[5] Unlike the output states $\boldsymbol{O}$, that are directly used within the Predictor and are not further used.

5.2.5 Message Aggregator

Agent communication is based on the previously extracted conflict features and the created notifications. Communication in the proposed model is a synonym for interaction between agents. The temporal information relevant for further description of the conflicts between agents was already done in the Conflict Identifier and is compressed into a directed conflict vector $\mathbf{c}^{a,b}$. The notification $\mathbf{u}^b$ contains information an agent b considers relevant for others, most likely information that helps others to resolve their conflict with agent b in order to allow a better final prediction. The directed edge feature vector between two agents $\mathbf{f}^{a,b}$ contains information that can be extracted from the traffic scenario beforehand and does not need to be learned separately by the model. The *Message Aggregator* then interprets conflict $\mathbf{c}^{a,b}$, information necessary to solve it $\mathbf{u}^b$, and geometric relational features $\mathbf{f}^{a,b}$ for each agent pair a, b and aggregates them over all sender agents b according to

$$\mathbf{m}^a = \sum_b \mu_{\mathrm{C}}^{a,b} \varphi_{\mathrm{Mess}}(\mathbf{c}^{a,b}, \mathbf{u}^b, \mathbf{f}^{a,b}) \quad (5.14)$$

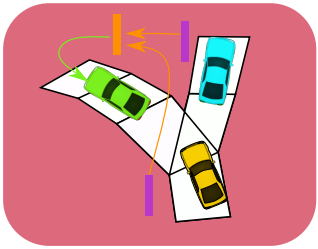


Figure 5.7: For each agent, the **Message Aggregator** combines received notifications, conflict messages and agent edge information to a message that helps the agent produce a better final prediction.

Note that the module processes the inputs individually for agent pairs before aggregating. By doing so, the module can interpret the relevance of the given notification – or information agent b considered useful for everyone to solve its conflicts – for the individual conflict between a and b.
For case $a = b$, a mask is used to permit an agent sending a notification to itself, so $\mu_{\mathrm{C}}^{a,b} = 1$ if $a \neq b$, and 0 otherwise. We now have an additional agent feature matrix $\mathbf{M} = [\mathbf{m}^a] \in \mathbb{R}^{n_{\mathrm{A}} \times n_{\mathrm{F,Mess}}}$ that represents information received from other agents and can therefore be interpreted as interaction messages.
Of course each agent needs the possibility to send itself information about its future path. This kind of ego communication is modeled in the next module.

5.2.6 Self-Message Extractor

As described before in Section 5.2.5, up to now an agent cannot send information gathered during map exploration in the Map-based Predictor as a notification to itself. This issue is solved in the *Self-Message Extractor*. Here, information about the scene collected in the Map-based Predictor the agent identified as being helpful for itself for improving the final prediction is created the same way as in the Notification Generation

$$\mathbf{S} = \max_{t,l}(\varphi_{\text{SelfMess}}(\boldsymbol{H}, \boldsymbol{P}_{\text{Map}})) \qquad (5.15)$$

with $\mathbf{S} \in \mathbb{R}^{n_{\text{A}} \times n_{\text{F,SelfMess}}}$.

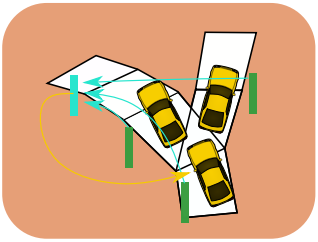


Figure 5.8: Just like the Notification Extractor provides important information for *other* agents, the **Self-Message Extractor** provides information important for the respective agent itself.

5.2.7 Final Prediction Estimation

In the final module, prediction is done iteratively just like in the Map-based Predictor, but instead of using only agent input features, interaction messages from the Message Aggregator and self-messages from the Self-Message Extractor are utilized. The messages of the Message Aggregator implicitly contain information gathered from the agent edge features. So instead of inputting matrix $\mathbf{F}_{\text{A}}$, the following matrices are concatenated $[\mathbf{F}_{\text{A}}, \mathbf{S}, \mathbf{M}] \in \mathbb{R}^{n_{\text{A}} \times n_{\text{F,A}} + n_{\text{F,SelfMess}} + n_{\text{F,Mess}}}$. The final position prediction tensor is called $\boldsymbol{P}_{\text{Final}} \in \mathbb{R}^{n_{\text{T}} \times n_{\text{L}} \times n_{\text{A}}}$.

5.2.8 Cooccurrence Estimator

With $P_{t,l,a} \equiv \boldsymbol{P}_{\text{Final}}$, an estimate is given on how likely it is that agent a will be on lane tile l at time step t. Let such an event be called $\Omega_{t,l,a}$, then $P_{t,l,a} = \mathcal{P}(\Omega_{t,l,a})$. However, what is even more important for analyzing different behavior options between a pair of agents is the conditional probability $\psi^{t,l,a,k,b} := \mathcal{P}(\Omega_{t,l,a} \mid \Omega_{t,k,b})$ between two agents at the same future time step t. In this work, the term *cooccurrence* is used for the set of conditional probabilities between two agents a, b for each time step t and all lane tiles l, k that are of

interest. The term is a neologism between the prefix *co* (latin: *together/with*) and the word *occurrence* and names dependent or independent behavior of the two respective agents relatively to each other.

The Cooccurrence Estimator receives the hidden states $\mathbf{h}_t^{l,a}$ produced from the Final Predictor just like the Notification Generator and the Self-Message Generator receive them from the Map-based Predictor. In case of the Cooccurrence Estimator, both states $\mathbf{h}_t^{l,a}$ and $\mathbf{h}_t^{b,k}$ are required in order to correctly conceive the intentions of both agents and how they are related. The conditional probability cannot be derived from GT since only one condition holds when the tracks of the agents have been recorded. So instead, the joint probabilities $\xi^{t,l,a,k,b} := \mathcal{P}(\Omega_{t,l,a} \wedge \Omega_{t,k,b})$ are predicted

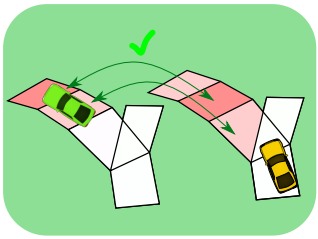


Figure 5.9: The **Cooccurrence Estimator** generates prediction associations by estimating joint distributions for pairs of agents. From those, conditional predictions given hypothetical actions can be derived.

$$\tilde{\xi}^{t,l,a,k,b} = \varphi_{\text{Joint}}(\mathbf{h}_t^{l,a}, \mathbf{h}_t^{k,b}). \tag{5.16}$$

with the result being concatenated to $\tilde{\boldsymbol{\Xi}} = [\tilde{\xi}^{t,l,a,k,b}] \in \mathbb{R}^{n_\text{T} \times n_\text{L} \times n_\text{A} \times n_\text{L} \times n_\text{A}}$. Formally, an output modeling a joint probability fullfils two requirements: It is commutative w.r.t. its individual events[6] and has the form of a probability distribution. With

$$\xi^{t,l,a,k,b} = \underset{l,k}{\text{norm}}(\tilde{\Xi}_{t,k,b,l,a} + \tilde{\Xi}_{t,l,a,k,b}) \tag{5.17}$$

a, l and b, k are permutable and the sum over positional outputs along the two lane tile dimensions l and k equals one, as is expected for an output with probabilistic character. Again, the result can be concatenated to $\boldsymbol{\Xi} = [\xi^{t,l,a,k,b}] \in [0,1]^{n_\text{T} \times n_\text{L} \times n_\text{A} \times n_\text{L} \times n_\text{A}}$.

The Cooccurrence Estimator can be seen as a separate head of the network. As for the Lane Tile Matcher, a major contribution is seen and benefit in adding the Cooccurrence Estimator. However, the model does not require the production of a good position prediction. Therefore, the Cooccurrence

[6] $\mathcal{P}(\Omega_{t,l,a} \wedge \Omega_{t,k,b}) = \mathcal{P}(\Omega_{t,k,b} \wedge \Omega_{t,l,a})$

Estimator is another optional module. The memory consumption is highlighted here, since the number of tensor entries is $2n_{\mathrm{F,S}}n_{\mathrm{L}}^2n_{\mathrm{A}}^2n_{\mathrm{T}}$, so it is a matter of resources whether it shall be added or not.
Considering the latest shift of research focus towards conditional prediction within the community (see Section 3.2.3), instead of using the joint position probability, the conditional position probability of an agent's future behavior given another agent's potential behavior might be more beneficial for application. An overview of the outputs of the model and recommendations about useful postprocessing is given in the following.

5.3 Network Output

Every tensor produced within the modularized model can be seen as an output that can be analyzed during inference in order to check whether they are as meaningful as described during the design process above. However, without putting further effort into output analysis, three directly interpretable outputs can be identified, because they are subject to optimization:

- The positional classification output *without* interaction is called $\boldsymbol{P}_{\mathrm{Map}} \in [0,1]^{n_{\mathrm{T}}\times n_{\mathrm{L}}\times n_{\mathrm{A}}}$ and is produced by the Map-based Predictor in Section 5.2.2. It contains the probability of an agent a being on a certain lane tile l at time step t.
- The positional classification output *with* interaction is called $\boldsymbol{P}_{\mathrm{Final}} \in [0,1]^{n_{\mathrm{T}}\times n_{\mathrm{L}}\times n_{\mathrm{A}}}$. It also contains the probability of an agent a being on a certain lane tile l at time step t. The difference is about which information flow is allowed up to that point in the model where those outputs are created and therefore which information is available in order to produce both position estimates.
- The joint positional classification output $\boldsymbol{\Xi} \in [0,1]^{n_{\mathrm{T}}\times n_{\mathrm{L}}\times n_{\mathrm{A}}\times n_{\mathrm{L}}\times n_{\mathrm{A}}}$ contains the joint probability of agent a being on lane tile l *and* agent b being on lane tile k at the same time step t.

With $\boldsymbol{P}_{\mathrm{Final}}$, it is straight forward to extract the probability for a certain agent choosing a certain maneuver at a certain future time step: The prob. mass of lane tiles representing said maneuver can be summed up. For example, the prob. mass before or beyond a stop line represent the "stop" or the "drive on" maneuvers, respectively. The same can be done for lane changes and different

route options. All examples only require a correspondence between lane tiles and maneuver they represent w.r.t. a certain agent.
Since most other approaches directly predict parametric distributions in Cartesian coordinates, a method for obtaining these from the proposed discrete output shall be proposed. A more sophisticated post-processing option is to fit a parametric distribution – e.g. a univariate Gaussian distribution in Frenet-coordinates for a certain mode – to the discrete distribution that $\boldsymbol{P}_{\text{Final}}$ contains. With that, precise longitudinal position estimates can be derived and used for trajectory optimization if required.
To account for cooccurrence, the joint position probabilities $\boldsymbol{\Xi}$ shall be post-processed. Using the definition of the conditional probability follows

$$\tilde{\psi}^{t,l,a,k,b} = \mathcal{P}(\Omega_{t,l,a} \mid \Omega_{t,k,b}) = \frac{\mathcal{P}(\Omega_{t,k,b} \wedge \Omega_{t,l,a})}{\mathcal{P}(\Omega_{t,k,b})} = \frac{\Xi_{t,l,a,k,b}}{P_{t,k,b}}. \tag{5.18}$$

The choice of indices of the condition t, k, b shall clarify that the latter index pair in $\psi^{t,l,a,k,b}$ refers to the condition.
In practice, normalization over the first lane tile dimension l is done with

$$\psi^{t,l,a,k,b} = \underset{l}{\text{norm}}\left(\Xi_{t,l,a,k,b}\right) \tag{5.19}$$

and all conditional probabilities $\psi^{t,l,a,k,b}$ can be concatenated to $\boldsymbol{\Psi} = [\psi^{t,l,a,k,b}] \in [0,1]^{n_{\text{T}} \times n_{\text{L}} \times n_{\text{A}} \times n_{\text{L}} \times n_{\text{A}}}$ and seen as a separate model output, since it can be derived from $\boldsymbol{\Xi}$ without further knowledge.

5.4 Training

Since many outputs exist, there are many ways of training the proposed model. In Section 5.4.3, their theoretic differences are explained in detail. Before, the loss functions used are described.

5.4.1 Position Loss Function

The positional GT is given by $\boldsymbol{P}_{\text{GT}} \in \{0,1\}^{n_{\text{T}} \times n_{\text{L}} \times n_{\text{A}}}$ and the prediction is $\boldsymbol{P} \in \{\boldsymbol{P}_{\text{Map}}, \boldsymbol{P}_{\text{Final}}\}$. The task is formulated as a classification problem: Each

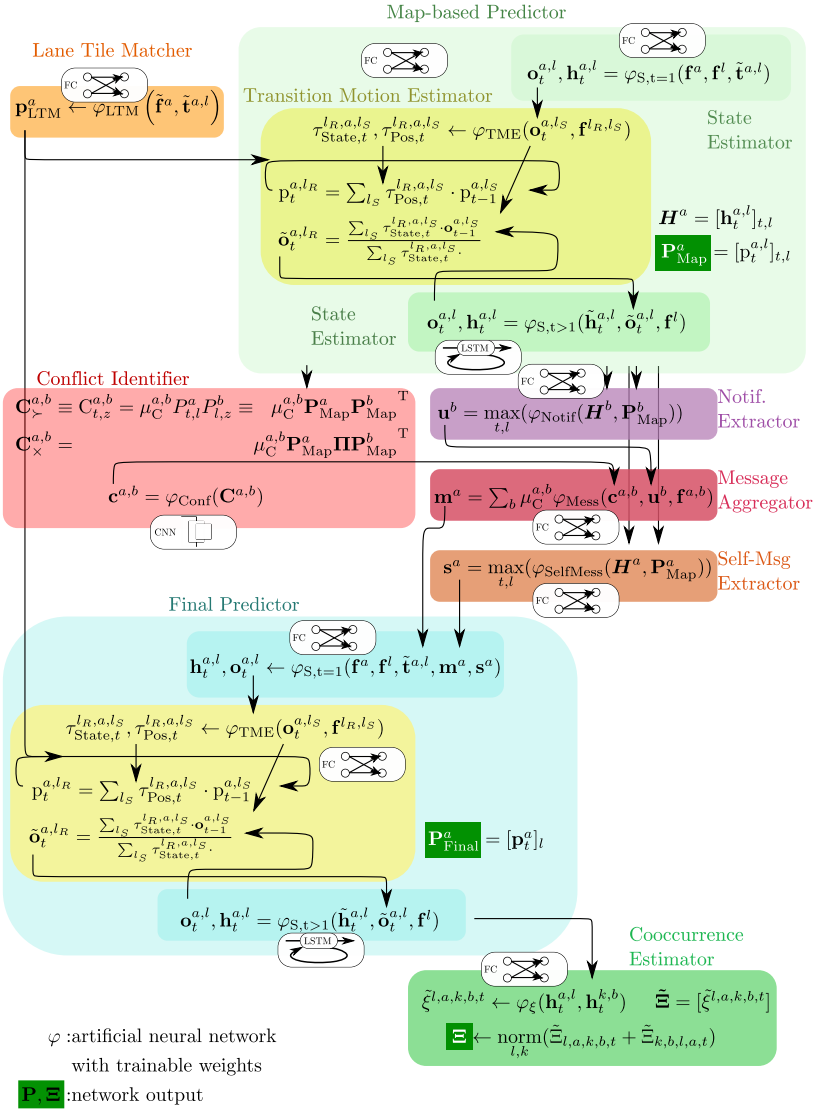


Figure 5.10: Overview of the proposed model and the 12 ANNs it consists of.

agent a is located at exactly one[7] lane tile l at a certain point in time t. The model assigns occupancy probabilities as a discrete distribution over all lane tiles for each agent and time step. The classification problem is therefore solved along the lane tile axis with GT being a one-hot vector $\mathbf{p}_{\mathrm{GT}}^{t,a}$. Since prob. mass can leak at the map subgraph entries and exits, leaked prob. mass $1 - ||\mathbf{p}^{t,a}||_1$ needs to be punished separately as long as GT is still within the scene. If leaks were not punished, the model might learn to propagate prob. mass out of the map graph as fast as possible. The resulting cross entropy loss for each positional vector is

$$\mathcal{C}_{\mathrm{Pos}}^{t,a} = -||\mathbf{p}_{\mathrm{GT}}^{t,a} \log(\mathbf{p}^{t,a}) + (\mathbf{1} - \mathbf{p}_{\mathrm{GT}}^{t,a}) \log(\mathbf{1} - \mathbf{p}^{t,a})||_1 - \log(1 - ||\mathbf{p}^{t,a}||_1), \tag{5.20}$$

so for the lane tile axis, the sum of cost terms together with a term for the leaked prob. mass is used.

Using binary cross entropy is the standard choice for classification problems in machine learning. For the present use case, a disadvantage requires consideration. Binary cross entropy decreases the more prob. mass is assigned to the GT class and increases the more prob. mass is assigned to a non-GT class. In the present example, a cost measure considering the graph distance of non-GT lane tiles w.r.t. the GT lane tile was useful. This way, prob. mass on a lane tile close to the GT lane tile increases cost less than prob. mass on a distant non-GT lane tile. However, a satisfying alternative was not found in literature. Since the proposed model is not optimized to *assign* but rather to *transfer* prob. mass onto a GT lane tile, the distance to the GT lane tile is implicitly considered in the training process. However, an explicit modeling alternative was preferred here.

For optimization, the remaining time and agent axes need to be reduced to a scalar loss. The Adam optimizer [53] introduces a momentum over consecutive training steps. In order so receive gradients in a comparable scale for which such a moving average is beneficial, it is required to average over the agent axis. This way, the model weights are adjusted in a comparable scale, regardless of whether there are 20 agents in the scene or just two. Since the number of time steps is constant, it does not make a change regarding the comparability of

[7] If an agent leaves a scene within the prediction horizons or initial lane tile matching did not succeed for some reason, the case $\mathbf{p}_{\mathrm{GT}}^{t,a} = \mathbf{0}$ needs to be handled.

scale of gradients whether the time step axis is averaged or summed. It was decided to average. Accordingly, the dimension is reduced with

$$\mathcal{L}_{\text{Pos}} = \frac{1}{n_{\text{T}} n_{\text{A}}} \sum_{a \in \Lambda} \sum_{t \in \Theta} \mathcal{C}_{\text{Pos}}^{t,a} \tag{5.21}$$

to obtain a scalar position loss $\mathcal{L}_{\text{Pos}}$.

Resolving Practical Problems during Optimization

Within the prediction horizon Δt_{ph}, new agents might enter the scene agents. Those are not considered in the problem formulation, even though they might have an influence on the behavior of the considered agents during Δt_{ph}. Some agents might leave the scene within the period Δt_{ph}. Also, lane tile matching during GT creation might fail for various reasons[8], so a GT label might be missing for certain prediction time steps. In both cases, there is no lane tile l marked as true for the respective agent a at the respective point in time t. When averaging over the time axis, loss and therefore gradients are overestimated, if GT is missing for easier small prediction horizons, and underestimated, if GT is missing for difficult large prediction horizons. In order to avoid unstable convergence, counter measures are taken.
A mask $\mu_{\text{Pos}}^{a,t}$ is introduced with

$$\mu_{\text{Pos}}^{a,t} = \begin{cases} 1, & \mathbf{p}_{\text{GT}}^{a,t} \exists l : \text{p}_l^{a,t} = 1 \\ 0, & \text{else} \end{cases} . \tag{5.22}$$

With the weighted average for each time step the reduced loss is

$$\mathcal{L}_{\text{Pos}} = \frac{1}{n_{\text{T}}} \sum_{t \in \Theta} \frac{\sum_{a \in \Lambda} \mathcal{C}_{\text{Pos}}^{a,t} \cdot \mu_{\text{Pos}}^{a,t}}{\sum_{a \in \Lambda} \mu_{\text{Pos}}^{a,t}}, \tag{5.23}$$

which requires for each time step t at least one agent a to have a valid ground truth to avoid division by zero.

[8] See Section 4.1.2 for explanation on how GT is obtained.

5.4.2 Joint Position Loss Function

The joint positional GT can be constructed with $\mathcal{P}(\boldsymbol{P}_{\text{GT}} \odot \boldsymbol{P}_{\text{GT}}) \equiv \mathcal{P}(P_{t,l,a} \wedge P_{t,k,b}) = \Xi_{t,l,a,k,b} \equiv \boldsymbol{\Xi}_{\text{GT}} \in \{0,1\}^{n_{\text{T}} \times n_{\text{L}} \times n_{\text{A}} \times n_{\text{L}} \times n_{\text{A}}}$ and the prediction is given by $\mathcal{P}(P_{t,l,a} \wedge P_{t,k,b}) = \Xi_{t,l,a,k,b} \equiv \boldsymbol{\Xi} \in [0,1]^{n_{\text{T}} \times n_{\text{L}} \times n_{\text{A}} \times n_{\text{L}} \times n_{\text{A}}}$. This task is formulated as another classification problem: If agent a is on lane tile l at time step t *and* agent b is on lane tile k at the same time step, then GT equals one, and zero otherwise. The cross entropy is therefore formulated as

$$\boldsymbol{\mathcal{C}}_{\text{Joint}} = -\sum_{l \in \Gamma} \sum_{k \in \Gamma} \boldsymbol{\Xi}_{\text{GT}} \log(\boldsymbol{\Xi}) + (\mathbf{1} - \boldsymbol{\Xi}_{\text{GT}}) \log(\mathbf{1} - \boldsymbol{\Xi}) \tag{5.24}$$

with $\boldsymbol{\mathcal{C}}_{\text{Joint}} \in \mathbb{R}^{n_{\text{T}} \times n_{\text{A}} \times n_{\text{A}}}$ and reduced to

$$\mathcal{L}_{\text{Joint}} = \frac{1}{n_{\text{T}} n_{\text{A}}^2} \sum_{a \in \Lambda} \sum_{b \in \Lambda} \sum_{t \in \Theta} \boldsymbol{\mathcal{C}}_{\text{Joint}}. \tag{5.25}$$

There is no leak in prob. mass this time, since the predicted joint probability distribution was normalized in Section 5.2.8 instead of propagating prob. mass through transition predictions as in the Predictor.
Still, a mask $\mu_{\text{Joint}}^{a,b,t}$ is necessary to ignore loss for agent pairs where at least one of the agents does not have a valid GT.

$$\mu_{\text{Joint}}^{t,a,b} = \begin{cases} 1, & \mathbf{p}_{\text{GT}}^{a,t} \exists l : \mathrm{p}_l^{a,t} = 1 \wedge \mathbf{p}_{\text{GT}}^{b,t} \exists k : \mathrm{p}_k^{b,t} = 1 \\ 0, & \text{else.} \end{cases} \tag{5.26}$$

In order to get gradients in a reproducible scale, the weighted average loss for each time step is

$$\mathcal{L}_{\text{Joint}} = \frac{1}{n_{\text{T}}} \sum_{t \in \Theta} \frac{\sum_{a \in \Lambda} \sum_{b \in \Lambda} \boldsymbol{\mathcal{C}}_{\text{Joint}} \cdot \mu_{\text{Joint}}^{t,a,b}}{\sum_{a \in \Lambda} \sum_{b \in \Lambda} \mu_{\text{Joint}}^{t,a,b}}, \tag{5.27}$$

which requires at least one agent combination (a, b) having a valid ground truth at each t.

5.4.3 Training Strategies

As mentioned in Appendix A.5, there are three main output tensors $\boldsymbol{P}_{\text{Map}}$, $\boldsymbol{P}_{\text{Final}}$ and $\boldsymbol{\Xi}$ with their corresponding losses $\mathcal{L}_{\text{Pos,Map}}$, $\mathcal{L}_{\text{Pos,Final}}$ and $\mathcal{L}_{\text{Joint}}$. Each loss can be minimized separately.

If $\mathcal{L}_{\text{Pos,Map}}$ is minimized, only the weights of the Lane Tile Matcher and the Map-based Predictor are adjusted, since they are the only information connection between input and $\mathcal{L}_{\text{Pos,Map}}$. If $\mathcal{L}_{\text{Pos,Final}}$ is minimized, all weights but the weights within the Cooccurrence Estimator are adjusted. Only if $\mathcal{L}_{\text{Joint}}$ is minimized, all weights of the model are optimized. However, due to the information control between some modules, the gradient flow is highly biased and restricted. For example, if the final loss is minimized, there is no reason why the Map-based Predictor is supposed to produce the best possible non-interactive prediction. The Map-based Transition Estimator still receives gradients due to the connection via the Conflict Identifier, but those are comparably small and lead to slow overall training. On the other hand, there is no reason to strive for a good map-based position prediction as long as the Map-based Predictor generates a prediction that is beneficial in order to minimize the final prediction loss.
Therefore, the first training strategy decision is pre-training the Map-based Predictor separately by minimizing the map-based loss for a couple of steps or training in an end-to-end fashion by only minimizing the final loss.
For the Cooccurrence Predictor, the main strategic question that arises is whether to include it at all. Since it is a separate head of the network and is not required for the position prediction, it is recommended to only use it, if its output can be further processed by later elements in the software pipeline of the automated vehicle. While a major benefit is seen and contribution in the Cooccurrence Predictor compared to related work, a major disadvantage is the large memory consumption of this module and the increase of runtime. The network's overall graphical processing unit (GPU) memory consumption roughly doubles when the Cooccurrence Predictor is added. Another question regarding the training of the Cooccurrence Predictor is whether to optimize all weights of the whole network, or just the one from the Cooccurrence Predictor itself in a late state after the weights of the other modules are frozen. Earlier publications suggest multi-task learning to be beneficial for the tasks-individual performance of a network, if those tasks are semantically related [97, 121, 122]. This holds for the present use case. Therefore, it seems reasonable to directly minimize the sum of the final losses

$$\mathcal{L}_{\text{Sum,End}} = \mathcal{L}_{\text{Pos,Final}} + \mathcal{L}_{\text{Joint}}$$

or directly the sum of all three losses

$$\mathcal{L}_{\text{Sum,All}} = \mathcal{L}_{\text{Pos,Map}} + \mathcal{L}_{\text{Pos,Final}} + \mathcal{L}_{\text{Joint}}.$$

If $\boldsymbol{P}_{\text{Map}}$ is not optimized directly, the map-based position prediction will not be precise. Its hidden states will be optimized through gradients coming from the Notification Generator and the Self-Messages Generator, the map-based position prediction is only optimized through the Conflict Identifier.

6 Evaluation

The used data is described in short in Section 6.1 and in detail in Appendix A.1. The visualization for inspecting and assessing representative examples qualitatively is given in Section 6.3. Then, the Lane Tile Matcher (Section 6.4), the prediction of statistically independent positions (Section 6.5) and the conditional prediction (Section 6.6) are evaluated. Hyperparameters used for evaluation are listed in Appendix A.3.

6.1 Datasets

The presented approach works best if the following requirements regarding the data are fulfilled:

1. Our approach was designed to specifically tackle prediction problems our institute's experimental vehicles face in daily traffic in Karlsruhe, Germany. In Germany, traffic is strongly rule-based and vehicles usually do not share a lane laterally. Therefore, the data should only originate from regions where drivers usually do not share a lane laterally and obey traffic rules to a large extent. From the European perspective, India is a standard example for traffic that is detached from map-derived traffic rules, US traffic also fulfills some criteria, which make only particular scenarios useable for our approach.

2. The approach works on a static number of lane tiles and traffic participants (TPs). This is a problem both for map-centric datasets, where intersections are statically recorded, and agent-centric datasets, where agents driving in the same direction as the ego vehicle are recorded for a long time, while oncoming traffic is tracked only for a short period.

3. A structured map is needed, that contains both the road topology and traffic rules. Ideally the same – namely Lanelet2 – format is used.

For training and evaluation, intersections from INTERACTION dataset [120], rounD [55] and inD [11] are used that fulfill those requirements. The 12

utilized intersections are illustrated in Fig. 6.1. None of the intersections are signalized. Regarding detailed descriptions of the intersection topologies, their surroundings and their traffic rules, we refer to Appendix A.1.
For every single evaluation method, the models are trained with eleven datasets and evaluate on the twelfth. All results illustrated in this work are therefore from evaluation intersections the model has not been trained on.
This is particularly challenging for inD location2, since it is the only intersection with priority to the right in the available datasets. Also, asymmetric merging in interaction CHN_Merging_lower only occurs here. All other topologies – roundabouts (four), symmetric merging (two[1]), priority roads with traffic signs (three[2]), straight multilane road (three[3]) – occur several times. Therefore, the models are expected to generalize properly for those intersection types, but may struggle with the former topologies. Of course, this problem is expected to vanish if more diverse data is available.

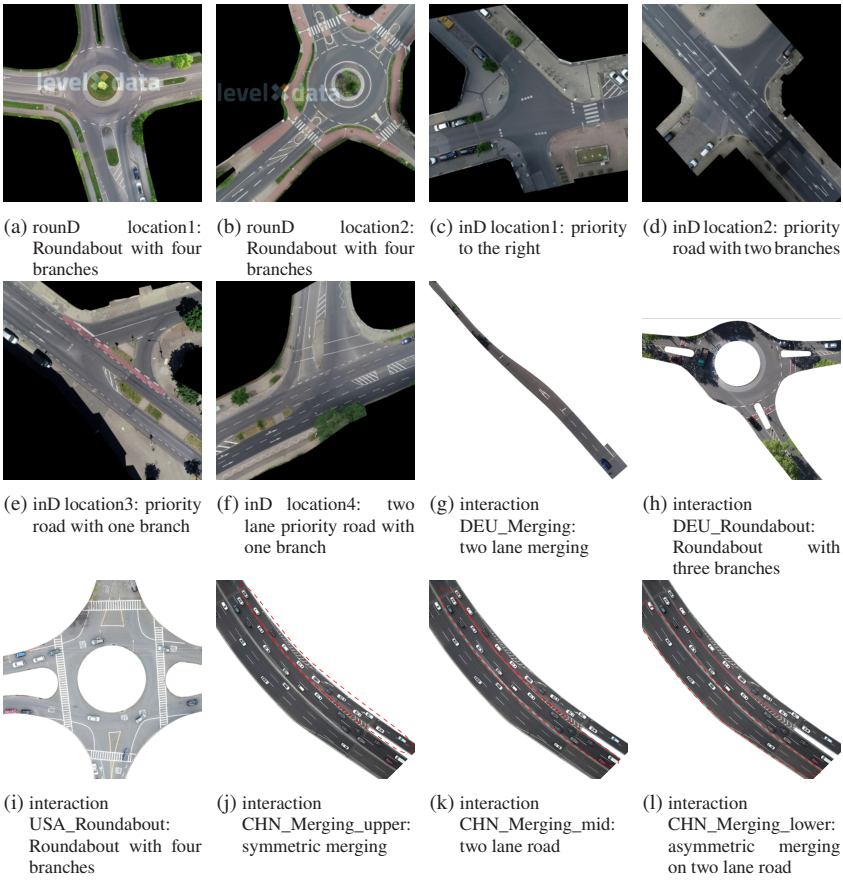
6.2 Runtime

In Fig. 6.2, the runtime is visualized for a model *with* and a model *without* Cooccurrence Estimator. The experiments were conducted on a NVIDIA RTX A6000 with TensorFlow version 2.5 and CUDA 11.0 without putting any additional effort into optimizing the model. For correctly assessing the result, the numbers of lane tiles in the used datasets shall be given in Table 6.1. It is assumed that the runtime is not an application critical factor, if focus is put on relevant agents and relevant map areas around the ego vehicle.

6.3 Visualization

For qualitative evaluation, a visualization is used that is explained in the following. In order to assess the prediction quality, videos were created that loop over the prediction horizon of each agent one after another, so in total, each video has $n_T \cdot n_A$ frames. From those videos, representative frames were extracted and used throughout this work.

[1] Interaction DEU_Merging & CHN_Merging_upper.
[2] inD location 1, 3 & 4.
[3] inD location 4, interaction CHN_Merging_mid & CHN_Merging_lower.



(a) rounD location1: Roundabout with four branches

(b) rounD location2: Roundabout with four branches

(c) inD location1: priority to the right

(d) inD location2: priority road with two branches

(e) inD location3: priority road with one branch

(f) inD location4: two lane priority road with one branch

(g) interaction DEU_Merging: two lane merging

(h) interaction DEU_Roundabout: Roundabout with three branches

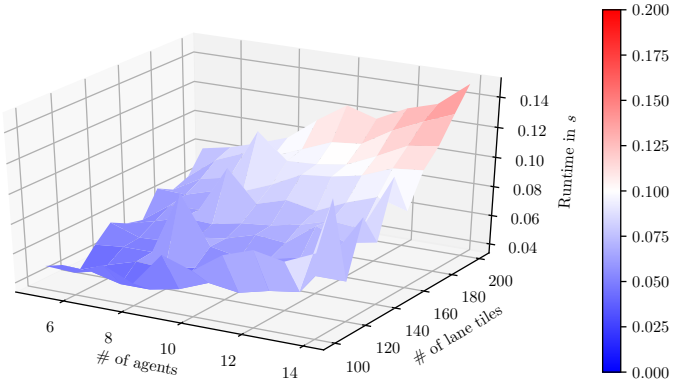
(i) interaction USA_Roundabout: Roundabout with four branches

(j) interaction CHN_Merging_upper: symmetric merging

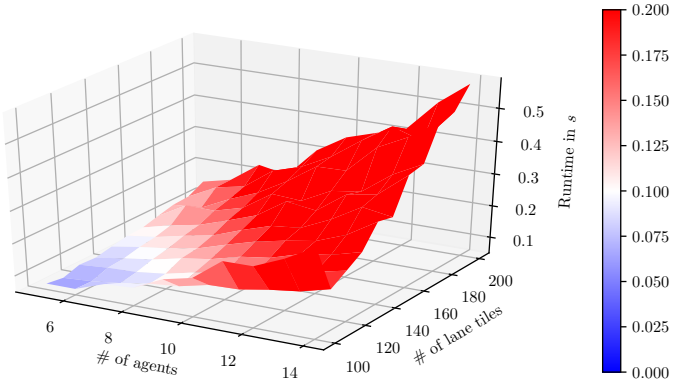
(k) interaction CHN_Merging_mid: two lane road

(l) interaction CHN_Merging_lower: asymmetric merging on two lane road

Figure 6.1: Arial images of the used intersections from rounD, inD and INTERACTION dataset.



(a) Runtime without Cooccurrence Estimator.



(b) Runtime with Cooccurrence Estimator.

Figure 6.2: Runtime depending on number of lane tiles and number of agents in the scene.

Table 6.1: Number of lane tiles in the corresponding scene.

Scene name	# of lane tiles in scene
inD location1	144
inD location2	97
inD location3	84
inD location4	158
rounD location1	120
rounD location2	141
interaction DEU_Merging	51
interaction DEU_Roundabout	77
interaction USA_Roundabout	110
interaction CHN_Merging_upper	39
interaction CHN_Merging_mid	46
CHN_Merging_lower	62

In Fig. 6.3, a simple example scene is given. The scene shows a roundabout in which two agents, one blue and the other red, are approaching two consecutive entries of a roundabout. The road topology is indicated by gray lines. An individual color gets sampled for each agent at each input time step. So in two different input time steps, an agent changes its color. In this example at the current time step, car #30 is red and car #29 is blue. On each agent's input state, its ID as given in the dataset and its momentary final prediction loss is printed. The current prediction horizon is stated in the caption in seconds.
The input state of each agent is visualized with a bounding box as given in the dataset with the agent's color. The GT state of each agent at the current prediction horizon is visualized with a transparent bounding box whose edges match the color of the corresponding input bounding box. In this example, #29 took the first exit of the roundabout and #30 entered the roundabout. Static agents and vulnerable road users (VRUs) that are not predicted are gray, predictions of agents that entered the scene after the current input time step

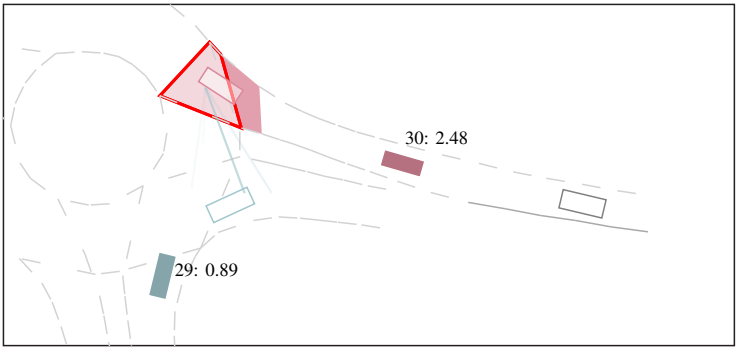


Figure 6.3: 2.4 s: This is the visualization used for qualitative evaluation. Two agents with individual colors and their actual future states are shown with bounding boxes. For each agent, its ID and its loss for the current prediction time step is written above its input state. The prediction of only one car is shown by coloring the lane tiles in the agent's color. The shade of the lane tiles indicates the prob. mass assigned to them. With colored lines, the conditional prediction of an agent given the currently marked ground truth (GT) lane tile is visualized.

are transparent bounding boxes with gray edges (on the right in the example image).

The occupancy prediction of the currently predicted agent #30 is visualized by coloring the lane tiles it was predicted on in the agent's color. If fewer than 1% of prob. mass is assigned to a lane tile, it is not visualized. The shade of each lane tile indicates the prob. mass on said lane tile. For visualization, we normalize the coloring of lane tiles, so the coloring of consecutive prediction time frames cannot be compared. Note that the color is not normalized by the size of the area a lane tile covers, which can lead to an optical bias regarding the actual prob. mass. A bright red edge surrounds the current GT lane tile for the currently predicted agent.

The conditional prediction is visualized with additional colored lines between lane tiles, making some frames visually overcharged. The colored lines connect the conditioned lane tile with the corresponding second agent's predicted lane tile. Again, the shade of the line indicates the prob. mass assigned to the corresponding lane tile. Since only actually occurring events can be properly evaluated, the conditional prediction is conditioned on the red GT lane tile of the currently predicted agent.

In Fig. 6.3, the blue lines indicate the position prediction of #29 given that #30 is on the GT lane tile. So since #30 already entered the roundabout, the corresponding conditional prediction of #29 points towards the first exit. Vice versa it is expected that the conditional prediction for #29, given #30 stops at the stop line before the roundabout, points towards the circuit.
It is claimed that the sum of examples shown in this work represents the overall prediction quality. Subjective impressions on the overall quality was gathered during many hours of manual example inspection while developing the presented approach.
Except for the boxplots of the Lane Tile Matcher (Fig. 6.4), outliers are not visualized. Due to the number of GT points, the number of outliers is often huge and does only create optical load.

6.4 Lane Tile Matcher

The Lane Tile Matcher is evaluated for all sequences[4], time steps and agents of an intersection, in which there is a geometric overlap between the front axle rectangle of the agent and at least two lane tiles of which at least one is not part of the agent's track Υ^a. This means only occasions in which the module can potentially fail at fulfilling its task are evaluated.
We then sum up the prob. mass assigned to lane tiles which are part of the agent's track

$$\mathbf{p}^a_{\text{LTM,Eval}} = \sum_{l \in \Upsilon^a} \mathbf{p}^a_{\text{LTM}}. \tag{6.1}$$

Ideally, $\mathbf{p}^a_{\text{LTM,Eval}} = 1$ holds, in the worst case, it is 0.
In Fig. 6.4, boxplots over all occasions are illustrated for all available locations. Qualitative examples are shown in Fig. 6.5 and Fig. 6.6. Descriptions of those examples can be found directly in the figures' captions.
For roundabouts (rounD location 1&2, interaction USA_Roundabout & DEU_Roundabout), the task of lane tile matching is particularly easy. The circuit and its branches are usually structurally separated, so it rarely occurs that there are geometric intersections of the front axle rectangle to a lane tile

[4] For each intersection, there are several recordings available, each of which probably correspond to one flight with a drone.

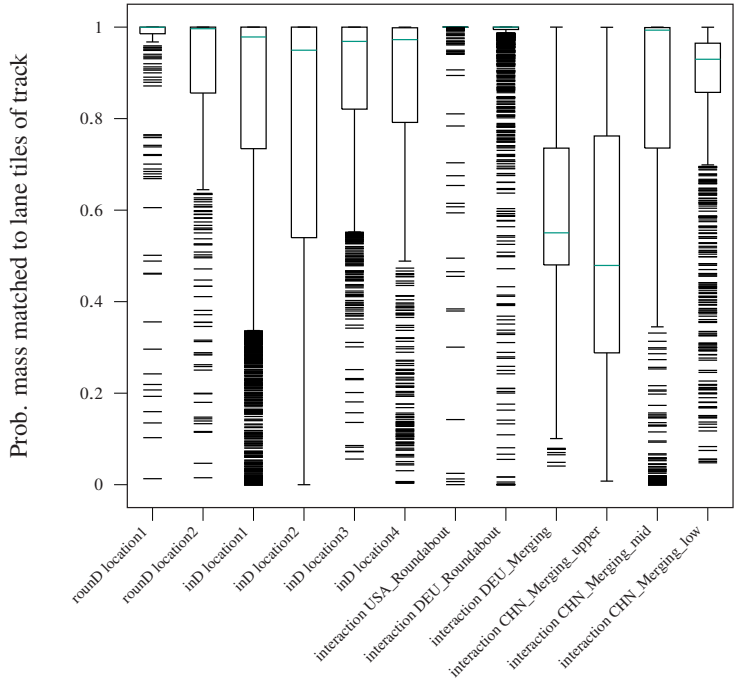
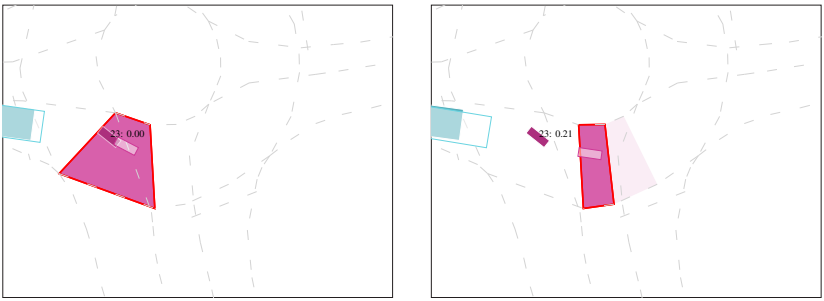
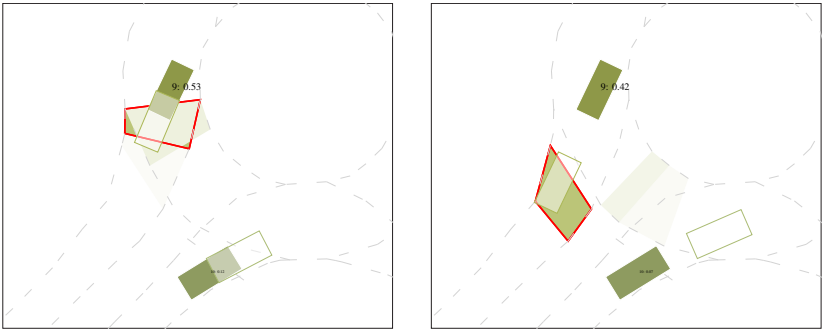


Figure 6.4: Results of Lane Tile Matcher. Share of prob. mass for each agent with geometric overlap of its front axle rectangle with at least two lane tiles of which at least one lane tile has to be outside the agent's track.

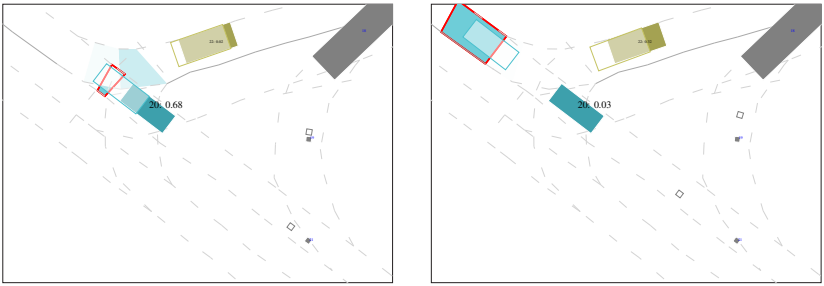
that is not in the agent's track. This only happens at roundabout entries or exits where lane tiles intersect each other (case A). The second option for wrong matches is when entry and exit of the same branch are brought together, usually right after a traffic island where pedestrians can cross (case B). For case A, lane tile matching is rather simple for the network because agents either turn right in order to get into the roundabout, or they turn left because they are already within the roundabout and follow its circuit. Also, the orientation of an agent usually fits to the orientation of the lane tiles. For case B, lane tile matching also is simple since potential lane tiles that are not part of the agent's track are located on the opposite side of the road. Again, the orientation of the agent



(a) rounD location1: Motorcyclist #23 is correctly matched to the circuit lane tile. No prob. mass is propagated towards the closest exit.

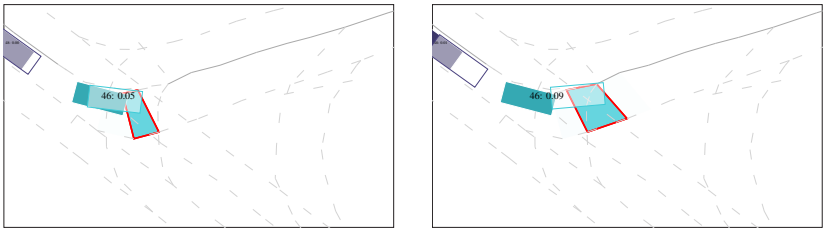


(b) rounD location2: Car #9 is correctly matched to the exiting lane tile. Around 10% of prob. mass was matched to the circuit lane tile.

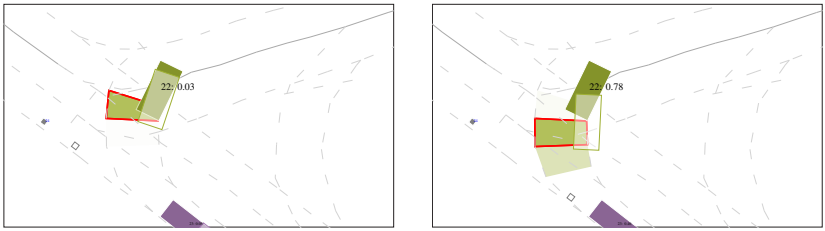


(c) inD location3: Car #20 is correctly matched to the priority lane tile. Around 20% of prob. mass was matched to the merging lane.

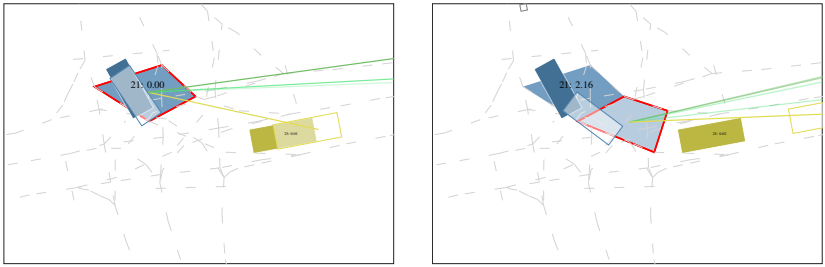
Figure 6.5: Lane Tile Matcher, representative examples 1. Left column: 0.3 s. Right column: 1.2 s.



(a) inD location3: Car #46 is correctly matched to the turning lane despite its front axle rectangle is overlapping with several lane tiles of the opposite road and the lane coming from the branch.



(b) inD location3: Car #22 is correctly matched to the turning lane.



(c) inD location2: Car #21 is correctly matched to the lane turning to the left (to the right from the reader's view). Its front axle rectangle overlaps with many lane tiles coming from and leading towards different directions. It also overlaps with lane tiles that belong to the two alternatives of the vehicle's branch: lane tiles that lead to straight and to the right (to the left from the reader's view).

Figure 6.6: Lane Tile Matcher, representative examples 2. Left column: 0.3 s. Right column: 1.2 s.

w.r.t. the lane tile is a strong prior for this task. In conclusion, the good results for roundabouts are expectable and do not indicate a mistake in the evaluation.
For symmetric merging scenarios (interaction DEU_Merging, CHN_Merging_upper), lane tile matching is difficult, not because the topology is particularly challenging, but because it is hard to tell at which point an agent has actually merged or changed the lane. The road consists of two merging lane tiles that overlap for a long time, even though the TPs already act like the merging has been accomplished and keep driving just as on a one way road. Also, errors during track identification are likely in this scenario. The comparably poor results shall therefore not be overrated, the task of assigning agents and lane tiles is simply not well-defined here.
Similar problems occur for multi-lane roads (interaction CHN_Merging_mid) and asymmetric merging scenarios (interaction CHN_Merging_lower). Automated track identification is not straight forward in these cases, especially when lane changes are conducted that usually require the length of several lane tiles.
In conclusion, the merging scenarios are added for completeness, but it is not recommended to overinterpret their results for lane tile matching.
For urban intersections apart from roundabouts (inD location 1-4), lane tile matching is most important and most challenging. Here, many lane tiles that lead to different directions can intersect. The Lane Tile Matcher produces particularly satisfying results. Since wrongful matching directly results in poor prediction results, the prediction result for intersections of the inD dataset also validated the Lane Tile Matcher's performance.

6.5 Prediction

The statistically independent final predictions generated by the Final Predictor are evaluated next. Metrics for sharpness and calibration are presented, applied, and exemplary results are illustrated.

6.5.1 Metrics

As elaborated in Section 2.4, sharpness and calibration[5] of predictions need to be evaluated. Calibration concerns the statistical compatibility between probabilistic forecasts and GT. Sharpness refers to minimizing the uncertainty of a prediction while calibration is maintained [38].
Popular datasets used for benchmarking[6] expect multiple Cartesian trajectories for a duration of 4 sec (INTERPRET) and 6 sec (NuScenes and Argoverse). The number of trajectories is limited to 6 (Argoverse, INTERPRET) and 25 (NuScenes[7]). All metrics used in popular benchmarks examine sharpness only.

Popular Sharpness Metrics

Through the mentioned datasets, the following prediction metrics have become popular[8]:

- average displacement error (ADE): The Euclidean distance between every point of a trajectory and the GT trajectory. The average over the prediction horizon duration is calculated.
- final displacement error (FDE): The Euclidean distance between the end point of a trajectory and the end point of the GT trajectory.
- Miss Rate: The total share of GT trajectory end points that are not within a 2 m Euclidean distance of at least one of the predicted trajectory end points.

Those metrics can be applied in several ways.

- Minimum: The metrics are evaluated for the trajectory with the smallest error (minADE, minFDE).
- Probabilistic: The metrics are evaluated weighted with the probability that is assigned to the best trajectory proposal (p-minADE, p-minFDE).
- Best/Top i: The metrics are evaluated for the top i trajectories (minADE_Top5, minADE_Top10, MissRate_Top5).

[5] Gneiting *et al.* [38] call it "calibration", Murphy *et al.* [78] and Zernetsch *et al.* [119] call it "reliability", but all refer to statistical compatibility of prediction and observations.
[6] NuScenes [15], INTERACTION/INTERPRET [120], Argoverse [22].
[7] However, for distance evaluation, only the top 5 or 10 trajectories are used.
[8] There are differences in how the metrics are evaluated precisely, but since it is not possible to apply them directly, a coarse overview should do.

Adopted Sharpness Metrics

The proposed model neither predicts trajectories nor does it predict in Cartesian space. Therefore, the following metrics for sharpness evaluation are used:

- **Displacement measure**: The share of prob. mass on the GT lane tile with lane tile distances $d \in \{0, 1, 2\}$. For $d = 0$, only the prob. mass on the GT lane tile is used, for $d = 1$, the prob. mass on the directly connected lane tiles is added, for $d = 2$, the prob. mass on the connected lane tiles after next are added. So beside visualizing the prob. mass on GT lane tiles, the prob. mass on lane tiles directly connected to the GT lane tile is summed up - including the GT lane tile. This is called the GT_1 lane tiles, since all lane tiles are included with a distance of up to one edge on the map graph. Similarly, the GT_2 prob. mass of all lane tiles with a distance of up to two edges to the GT lane tile on the map graph is defined.
- **Miss Rate**: If GT is not on one of i lane tiles with the most prob. mass, it counts as a miss.

Method to Validate Reliability

A method seen as highly valuable and which does not require binning was proposed by Zernetsch *et al.* [119], but it is not applicable to our discrete – "binned" – distribution $\mathbf{p}^{a,t}$. Instead, a method initially proposed by Murphy *et al.* [78] is used:
Lane tiles on which 10% of positional prob. mass was predicted are expected to match the GT lane tile in 10% of the cases if the sample size is large enough. We therefore create nine intervals

$$\mathbb{V}_i :]i \cdot 0.1 - 0.05, i \cdot 0.1 + 0.05[\; \forall \, i \in \{1, 2, ..., 9\} \tag{6.2}$$

and count the number n^i_{GT} of lane tiles l with $\mathbf{p}^{a,t}_{\text{Final}} \cdot \mathbf{p}^{a,t}_{\text{GT}} \in \mathbb{V}_i$ *with* GT being on l and the number $n^i_{\text{not GT}}$ of lane tiles with $\mathrm{p}^{a,t}_l \in \mathbb{V}_i$ *without* GT being on l. Ideally, we expect

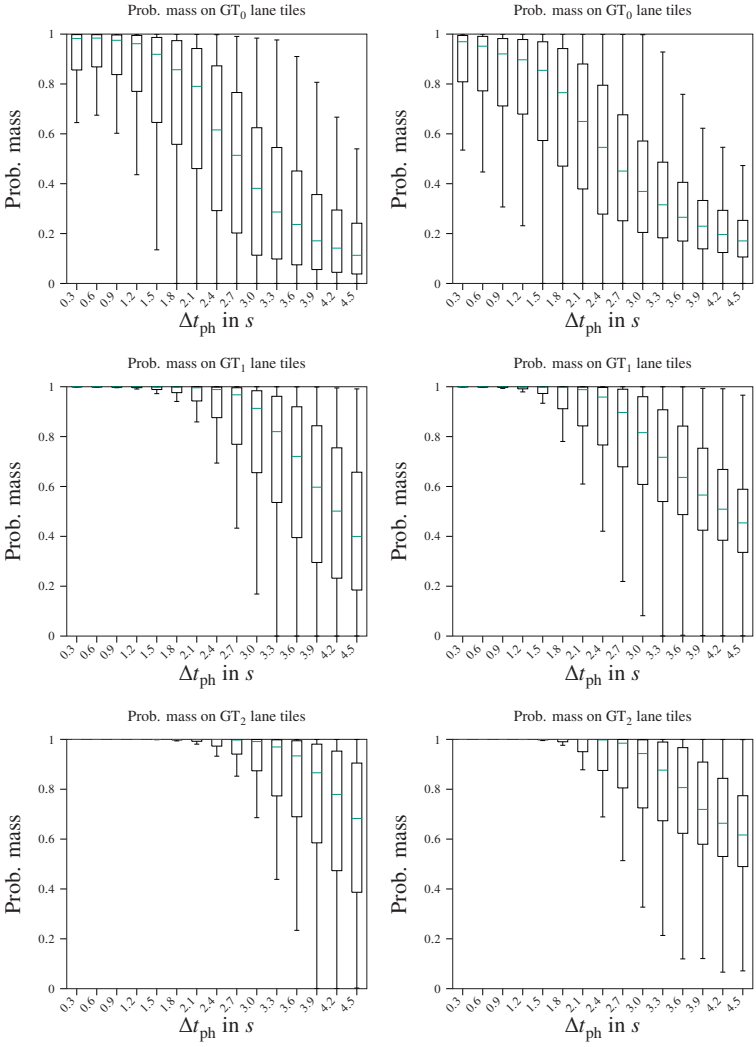
$$\frac{n^i_{\text{GT}}}{n^i_{\text{GT}} + n^i_{\text{not GT}}} = \frac{i}{10} \tag{6.3}$$

to hold for all i.

6.5.2 Quantitative Evaluation of Sharpness

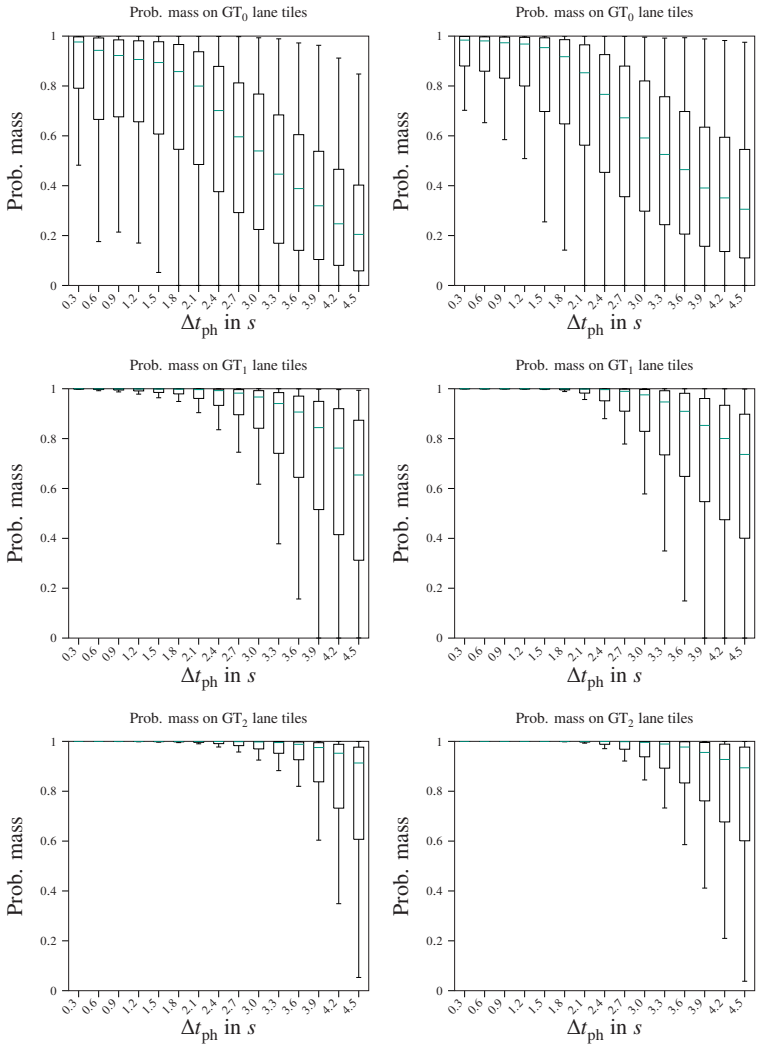
The share of prob. mass on the $GT_i \; \forall \, i \in \{0, 1, 2\}$ lane tiles is visualized for representative locations for rounD in Fig. 6.7, for inD in Fig. 6.8 and for INTERACTION dataset in Fig. 6.9. Due to the dataset size, the large number of outliers below and above the box plots' antennas are not shown.
At first glance, there are some similarities between Fig. 6.7, Fig. 6.8 and Fig. 6.9. For GT_0, the median starts for small prediction horizons around 1 and ends for large prediction horizons between 0.1 and 0.3. In between, there seems to be a logistic transition.
Already for GT_1, when taking into account prob. mass on lane tiles directly connected to the GT lane tile, the median gets close to 1 for progressing prediction horizons around 3.0 s. The median for the final prediction horizon of 4.5 s exceeds 0.4 for all scenarios.
If the prob. mass on the lane tiles after next connected to GT are also considered (GT_2), the median exceeds 0.6 for all scenarios and even exceeds 0.8 for non-roundabouts.
Some phenomena that catch attention are the following.
For GT_1 on interaction DEU_Merging, the confidence interval first shortens until $\Delta t_{\text{ph}} = 1.8\,\text{s}$ and then lengthens again. This might be related to lane tile matching fails, which have already been described in Section 6.4. GT generation was identified as particularly difficult when two lanes overlap directly before merging. Shortly after, the lanes have merged which directly leads to improved prediction result by map design, since there is no alternative for agents other than being in one of the merged lane tiles. So it is expected that this particular phenomenon does not indicate an unknown evaluation artifact but rather an artifact in map or loss function design. Both are side effects of treating a distance problem as a classification task over lane tiles. Later in the qualitative example in Fig. 6.20 it is clearly noticeable that the model does not have a particular problem with prediction during merging scenarios.
The second phenomenon corresponds to the prediction quality decline for roundabouts. Compared to the intersections in inD, agents in roundabouts are predicted less accurately than agents on other intersection types. This most likely stems from the location of the intersections. Roundabouts generally connect three or four branches with roughly the same amount of traffic in each branch. From a road planner standpoint, signed intersections are a better fit in case there are two branches with more traffic than the other branches. The road with higher traffic volume is labeled as priority road and all other branches have

to yield. This also holds for the signed intersections in inD, so the model most likely learns to predict towards those branches instead of predicting someone to turn into a side branch. Especially in inD location1, one side branch leads to a dead end, the other leads to a 30 kph-zone. For inD location3 & 4, there is only one side branch and one priority road. For rounD location2 & 3 and interaction DEU_Roundabout, all branches subjectively have roughly the same amount of traffic.
The miss rates for the five lane tiles with highest prob. mass for each location and prediction horizon can be found in Table 6.2. As already mentioned, inD location2 is particularly difficult, since there are many VRUs that affect the predicted vehicles but are not included in the proposed model. Also, it is the only scene with priority to the right, so the model could not properly anticipate how TPs behave there.
Beside the rather high miss rate for inD location2, the results seem acceptable, especially since the miss rate is extremely low until 3.0 s and only increases significantly for the last 1.5 s.
In Fig. 6.10, Fig. 6.11 and Fig. 6.12, qualitative examples for the final prediction under assumption of statistical independence are illustrated. The chosen scenes contain interaction, since simple non-interactive behavior was observed to be predicted quite well during development. A description of all qualitative examples is directly in the figures' captions. This way, the reader can compare his or her own interpretation of the illustrated scene with the given interpretation without jumping between pages.
Figure 6.10 shows a yield scenario, where agent #42 drives straight on the priority road, agent # 40 yields before turning from the priority road, and agent # 34 enters the intersection last. The left column shows the three agents for a prediction horizon of 1.8 s, the right column shows the scene for a prediction horizon of 3.3 s. The timing of the scene's evolution was not on point. Both agent #40 and #34 are predicted to accelerate faster than they actually did. However, qualitatively, the order of driving maneuvers is predicted accurately. As can be seen for agent #42 in this example, predicting constant velocity driving is can be done accurately by the model.
Figure 6.11 shows a roundabout, where agent #23 is about to leave the roundabout, while agent #24 and agent #26 are about to enter it at two consecutive entries around agent #23. The left column shows the scene at a prediction horizon of 2.4 s, the right column shows a prediction horizon of 3.9 s. The uncertainty on the scene's evolution therefore stem from the different maneuver options regarding entering and leaving the roundabout. Here, the shortcom-



(a) rounD location1 (b) rounD location2

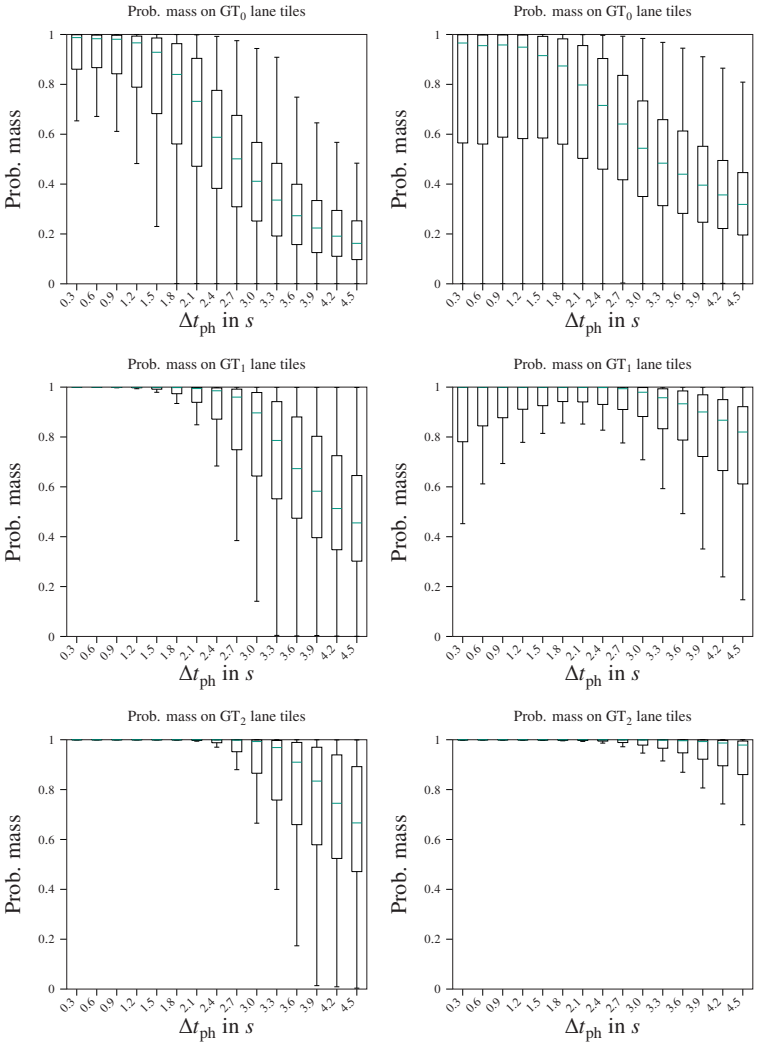
Figure 6.7: Box plots of prediction prob. mass on GT_i for both locations of rounD.



(a) inD location4

(b) inD location3

Figure 6.8: Box plots of prediction prob. mass on GT$_i$ for location 3 & 4 of inD.



(a) interaction DEU_Roundabout

(b) interaction DEU_Merging

Figure 6.9: Box plots of prediction prob. mass on GT_i for DEU_Roundabout and DEU_Merging of the INTERACTION dataset.

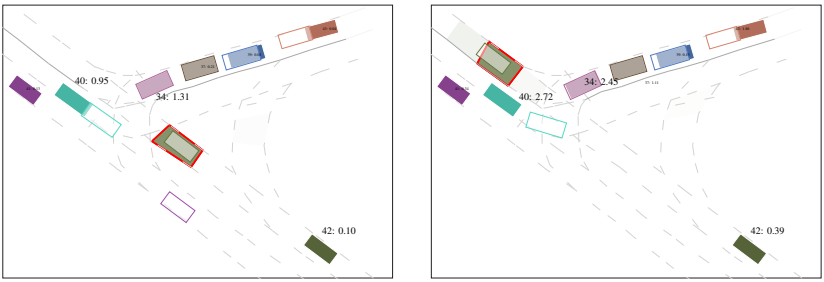
Table 6.2: Miss rates of the top 5 lane tiles in %

location / Δt_{ph}	0.3 s	0.6 s	0.9 s	1.2 s	1.5 s	1.8 s	2.1 s	2.4 s	2.7 s	3.0 s	3.3 s	3.6 s	3.9 s	4.2 s	4.5 s
rounD location1	0.0	0.0	0.0	0.0	0.2	0.2	0.4	0.8	1.4	3.9	5.2	7.9	11.6	16.1	20.7
rounD location2	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.9	1.4	1.9	3.5	4.7	7.4	10.0	13.1
inD location1	0.2	0.3	0.5	1.1	2.4	4.3	6.6	9.2	12.5	15.6	19.3	23.6	27.3	31.0	35.2
inD location2	0.0	0.0	0.1	0.1	0.3	0.8	1.6	2.9	4.6	6.9	9.8	12.4	15.9	19.0	23.0
inD location3	0.0	0.1	0.2	0.2	0.1	0.2	0.5	0.8	1.9	3.1	4.7	6.4	9.0	11.6	13.9
inD location4	0.0	0.0	0.1	0.2	0.3	0.3	0.8	1.4	2.2	3.2	4.8	7.8	11.2	14.9	20.1
interaction USA_Roundabout	0.0	0.0	0.3	0.1	0.1	0.3	0.4	0.7	0.9	1.3	2.0	2.3	3.0	3.7	5.1
interaction DEU_Roundabout	0.0	0.0	0.1	0.1	0.1	0.1	0.3	0.4	0.7	1.1	2.2	3.7	6.3	8.9	12.1
interaction DEU_Merging	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.2	0.4	0.5	0.7	1.1	1.4
interaction CHN_Merging_upper	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.4	0.6	0.9
interaction CHN_Merging_mid	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.6
interaction CHN_Merging_low	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.3	0.7	1.0	1.8	2.8	4.4

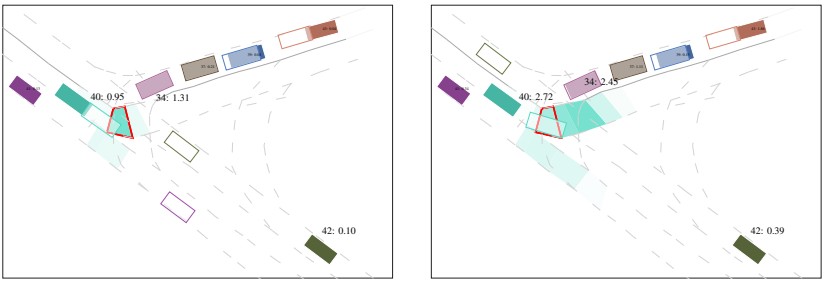
ings of assuming statistical independence of future positions are highlighted: Correspondences of maneuver options have to be guessed by the viewer, if behavior options cannot be conditioned on each other by the model itself. Also, bad predictions happen occasionally as can be seen for agent #26 who is predicted to stay within the circuit but actually took the first exit after entering. Figure 6.12 shows a different roundabout that is entered by agent #17 and successively by agent #8, one entry before agent #18 plans to enter it. The left column shows the scene at a prediction horizon of 3.6 s, the right column shows a prediction horizon of 4.5 s. Again, uncertainty stems from different maneuver options induced by position conflicts. All agents have basically two maneuver options, but correspondences have to be guessed without
All examples show the properties of the generated statistically independent predictions. First, the predictions look reasonable and correspond to the viewers expectations. Second, the mode of each maneuver option can be seen quite clearly. Third, the longitudinal uncertainty seems high compared to qualitative examples of state-of-the-art approaches in literature that have been optimized to achieve good positional results in popular benchmarks that assess predictions generated under the implicit assumption of statistical independent of future positions. Forth, correspondences of prediction modes can be guessed by the viewer, but not extracted in an automated way.

6.5.3 Quantitative Evaluation of Calibration

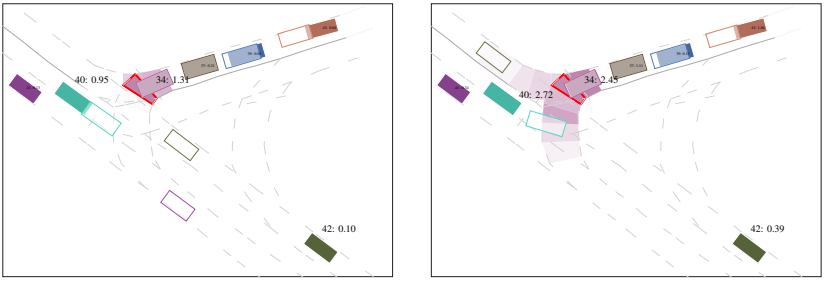
Figure 6.13 shows nine prob. mass intervals on the abscissa. For each interval, the + marks the ratio of lane tiles *with* GT being on this lane tile relative to the total number of lane tiles with prob. mass in the corresponding interval. Ideally, lane tiles with 5% to 15% prob. mass are expected to match with GT in around 5% to 15% of cases. Also, lane tiles with 85% to 95% prob. mass are expected to match with GT in around 85% to 95% of cases, and so on. The + marks should therefore be close to the symmetry axis $f(x) = x$ visualized in green.
As can be inspected, reliability holds particularly well for most scenarios. For rounD location1 and inD location1, predictions are overconfident: For lane tiles with high prob. mass, GT hits those lane tiles less frequent than expected from the assigned prob. mass. For rounD location1, this might result from the limited data that is available compared to other locations. inD location2 is the only intersection with priority to the right in the datasets. The model evaluated on inD location2 has not seen scenarios with priority to the right



(a) inD location3: The mud green car #42 on the priority road passes the intersection. It represents many of those vehicles in the datasets that simply drive straight on with constant velocity and are well captured by the model.

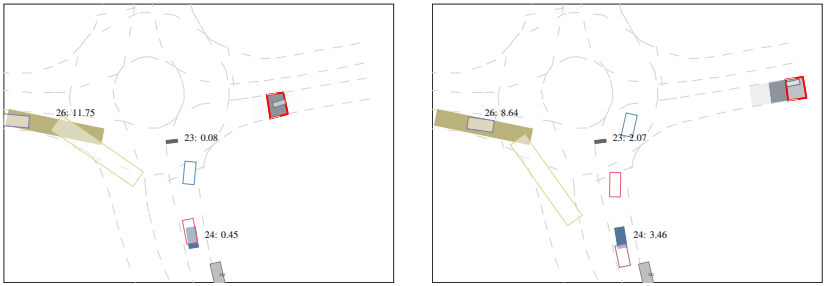


(b) inD location3: Then, the turquoise car #40 turns to the left...

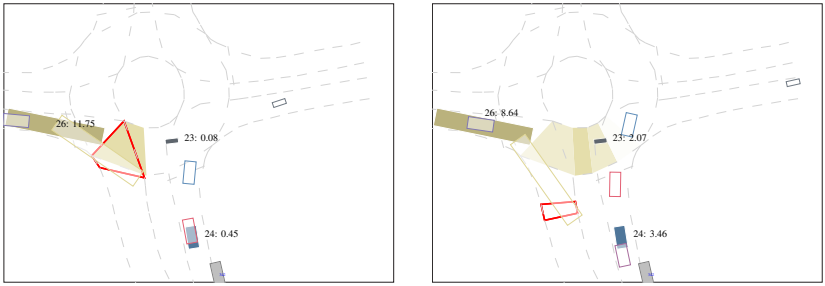


(c) inD location3: ...slightly before car #34 on the yielding branch turns onto the priority road. Still, there is a large share of prob. mass behind the stop line. A major issue is the limited visible range of the scenes. The model might have learned to always keep a bit of pro. mass behind the stop line since it can never be ruled out that a fast car is coming from invisible area of the priority road.

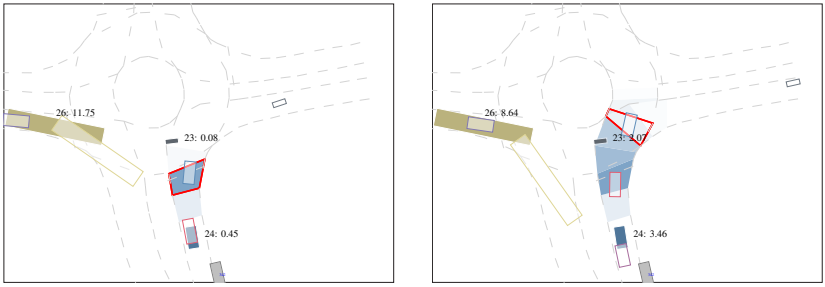
Figure 6.10: Final prediction, examples 1, inD location3. The priority main road has a branch towards the top of the picture. #42 passes. Afterwards, #40 turns left. Last, car #34 turns left. Despite not catching the time of the left turns precisely, qualitatively, the scene evolvement was correctly estimated. Left columns: 1.8 s. Right column: 3.3 s.



(a) Motorcyclist #23 is correctly predicted towards the closest exit of the roundabout. The prediction certainty might have been derived from the turning rate, that is already close to zero, since the motorcyclist's heading directly point towards the exit.

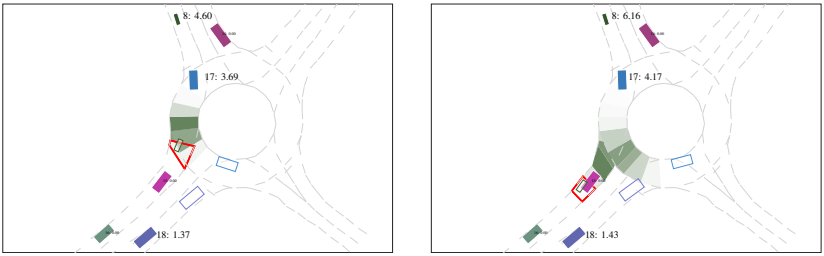


(b) Bus #26 enters the roundabout, but it is unclear whether it will take the first exit or stay within the circuit. The model incorrectly predicted the bus to stay within the roundabout with high certitude which might be a wrongful conclusion from a small turn rate, since it had to stop and yield to the motorcyclist #23 before. Is cannot be derived from the illustrated pictures that motorcyclist #23 was within the circuit before #26 could enter. This is additional information given for interpretation.

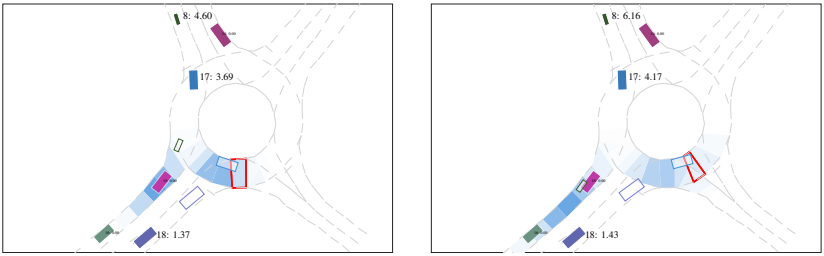


(c) Car #24 yields to the bus at first. Later, it is predicted to enter the roundabout with 30% to 50% and to yield with the 50% to 70%. The entering mode might correspond to passing the roundabout before bus #26 passed #24's entry. Otherwise, it is difficult to interpret why the bus was predicted to drive within the circuit with nearly 100% and #24 still enters.

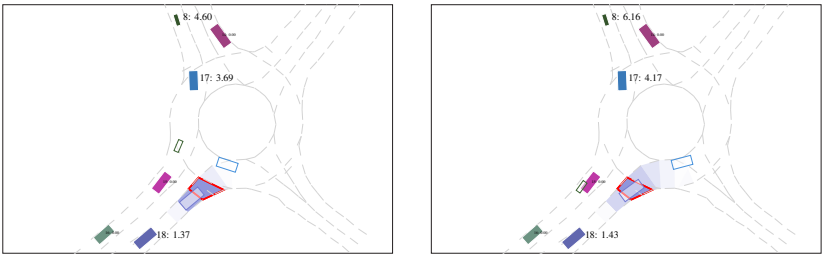
Figure 6.11: Final prediction, examples 2, rounD location1. Left columns: 2.4 s. Right column: 3.9 s.



(a) Motorcyclist #8 is predicted to directly enter the roundabout. Later, both circuiting option and exiting option are predicted with about 50% probability. Its velocity was slightly underestimated, since the prediction mode is about 3 m behind it. Thus, the motorcyclist is not predicted to have passed the entry branch of #18, yet, if #8 stayed within the circuit.



(b) Car #17 is predicted with around 50% each to leave or stay within the roundabout. Its velocity was also slightly underestimated, since the mode of the circuiting option is about 2 m behind the actual position of the car. Also, the longitudinal uncertainty is quite high for both options.



(c) Car #18 is predicted to stop before entering the roundabout, and to slowly enter the roundabout after #17 has passed. There is still more than 50% of prob. mass behind the stop line, since motorcyclist #8 was not predicted to have passed yet, if it had stayed within the circuit.

Figure 6.12: Final prediction, examples 3, rounD location2. The left column shows 3.6 s, the right shows 4.5 s. The example shows how important a conditional prediction is, even if there are only few agents in the scene with only two reasonable intention options each.

during training. Therefore, it is surprising that there is no stronger decrease in reliability, be it overconfidence or underconfidence.
In interaction CHN_Merging_lower, the network shows to be slightly underconfident in the middle intervals. This most likely also originates from it being the only scenario with an acceleration lane leading to asymmetric merging.
In total, the result shows that our model generalized properly w.r.t. to reliability.

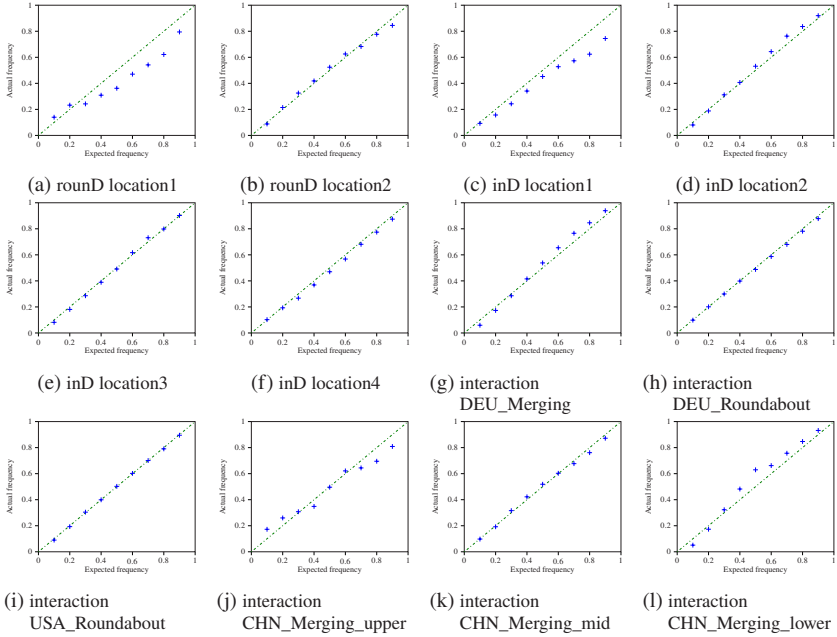


Figure 6.13: Evaluation of statistical reliability. Each + is a point $(\frac{i}{10}, n^i_{\text{GT}}/(n^i_{\text{GT}} + n^i_{\text{not GT}}))$ with its x value being the expected share of GT lane tiles to all lane tiles in the respective probability interval and its y value being the actual share.

6.5.4 Comparison with Baseline Methods

For comparison, several baseline models are created and tested, so the numbers from the quantitative evaluation can be put into perspective more easily. All baseline models are simplified versions of the proposed approach to illustrate the benefits of each design feature.

As first baseline BL1, a naive version of the Map-based Predictor is used with minimal inputs. For the agent graph, only the velocity of each agent and the distances between them are given. For the map graph, only lane tile centerline length and the transformation information for the connecting edges are given. The naive model consists of single layer linear mappings only. Since the focus shall be on actual prediction and not lane matching, the Lane Tile Matcher is used with the complete input and the two hidden layers with non-linear activation functions. Also, only a single hidden layer is used. The two multi-layer perceptrons (MLPs) of the original Initial State Estimator and the Transition Motion Estimator are exchanged with a single linear mapping that generates states and motion transition probabilities. The same holds for the long short-term memory (LSTM) that implements the Recurrent State Estimator and MLP implementing the Transition Motion Estimator. Again, only a single layer linear mapping is used. In conclusion, the recurrent part generating most of the predictions can be seen as a discrete filter optimized for the present problem of prediction.
The second baseline BL2 has the same structure as BL1 but uses all available input features for both the agent graph and the map graph.
The third baseline BL3 is the Map-Based Predictor of the original model that differs from the final prediction only by not explicitly allowing interaction.
All baselines are trained the same way as the proposed model, particularly including the adaptations of the loss function.

In Fig. 6.14, the results of the baselines and the proposed approach on rounD location1 are shown. Noticeable improvements stem from the following design changes. The naive linear model with minimal input features and without interaction (BL1) clearly performs poorly over the whole prediction (a). A huge improvement stems from adding all available input features (BL2), even though the model is still linear (b). BL2 already has the chance to implicitly learn interaction from the data by learning to stop at stop lines or yield signs, even though it cannot perceive whether other agents are about to cross on the priority lane or the circuit of a roundabout.
Surprisingly, the map-based prediction (c) is nearly as good as the final prediction (d). For GT_0, differences are quite difficult to notice. Differences in the median, the confidence interval or the antennas often seem smaller than 0.1 For GT_1 and GT_2, the visual appearances of the plots diverge. This clearly indicates an advantage of the model.

However, the question rises why the map-based prediction seems nearly as good as the final prediction, even though agents are not aware of each other. Several explanations come to mind.
First, the map-based prediction was claimed to represent trajectories that would be driven if no interaction is necessary. This suggests that the prediction has to be adopted to the route topology and requirements of comfort only. This is clearly not the case, since both the map-based prediction and the final prediction are optimized to the same data that stems from real world scenarios *with* interaction. So it is rather the best possible prediction without letting agents know about each other's existence in the scene. Just as BL2, the map-based prediction can benefit from all map features suggesting where best to stop proactively.
Second, real world TPs adopt their driving behavior proactively if curves or yield situations occur. This way, their comfort requirements regarding longitudinal and lateral dynamics can be satisfied better and energy consumption as well as material wear decreases. So there may already be hints in the velocity and acceleration agent input on how the real world TP drove proactively in order to avoid a harsh braking maneuver.
Third, and related to the aforementioned reason, conflicts occur comparably rarely, so a large share of driving consists of agents driving on the same lane they were driving on beforehand. Even though particularly difficult scenarios have been selected from the available datasets, a significant share of driving occurs on intersection branches, that do not require many speed adaptations.
Last, multi-modality of individual predictions cannot be rewarded mutually. As mentioned many times, predicting agents individually is assessed quantitatively by comparison with GT. In this case, a bimodal prediction for two agents without joint assessment might numerically lead to the same result as a bimodal prediction of a single agent that does not know whether there is another agent in the scene it has to interact with, or not. The map-based model could learn to always produce bimodal prediction at stop lines or yield signs, so the final prediction with interaction only has a numerical benefit if there is no other vehicle coming and the prediction is uni-modal.
Changing the optimization strategy leads to the result in Fig. 6.15. In (f), the loss of the final prediction used for optimization leading to results of the map-based prediction in (e). As expected, the prediction quality of the map-based prediction clearly declines compared to (c) in Fig. 6.14. The final prediction (f), however, improves a bit compared to optimizing both map-based and final loss (g). The conditional prediction (h) given the GT of the corresponding second

agent outperforms all prediction training strategies for statistically independent positions which is clearly visible for GT_1 and GT_2.

6.6 Conditional Prediction

For quantitative evaluation of sharpness and reliability of the conditional prediction, the same metrics as introduced in Section 6.5.1 and Section 6.5.3 are used. Quantitatively, only the predictions $\mathcal{P}(\Omega_{t,l,a} \mid \Omega_{t,k,b})$ conditioned on events $\Omega_{t,k,b}$ that actually happened during recording can be evaluated. Therefore, all results for the conditional predictions are evaluated twice for each agent pair conditioned on the GT of the respective other agent, if GT exists for both agents at the same prediction horizon. Again, all results are created by removing the evaluation map from the training set, so the model has not seen the corresponding map before.
All data points from the respective map are used, in which a GT lane tile exists for both the predicted and the conditioned agent.

6.6.1 Quantitative Evaluation of Sharpness

In Fig. 6.16 and Fig. 6.17, sharpness is evaluated with the displacement measure GT_i that was defined in Section 6.5.1 for four representative locations.
If the boxplots of the predictions with implicit assumption of statistical independence regarding future positions from Fig. 6.7 and Fig. 6.9 are compared with the boxplots of conditional predictions of the same locations, as expected, an increase of sharpness can be observed for the conditional predictions. Since correspondences of pairwise behavior is evaluated, a decrease of sharpness had been counter-intuitive. If the sharpness had been quantitatively the same, the reader could have interpreted it as a validation for the assumption of statistical independence. The increase of sharpness is expected to be even more intense, if only agent pairs that require interaction would have been regarded. Instead, many agent pairs do not interfere with each other[9], e.g. in sparse or slowly evolving scenes. Fore those scenes, statistical independence of future positions can be a valid assumption and barely any numerical advantage of the conditional prediction can be observed. The clear advantage of the conditional

[9] Here, both causal interaction – like priority and yielding behavior – and numerical correlation that can also stem from indirect interaction that involves more than two agents is referred.

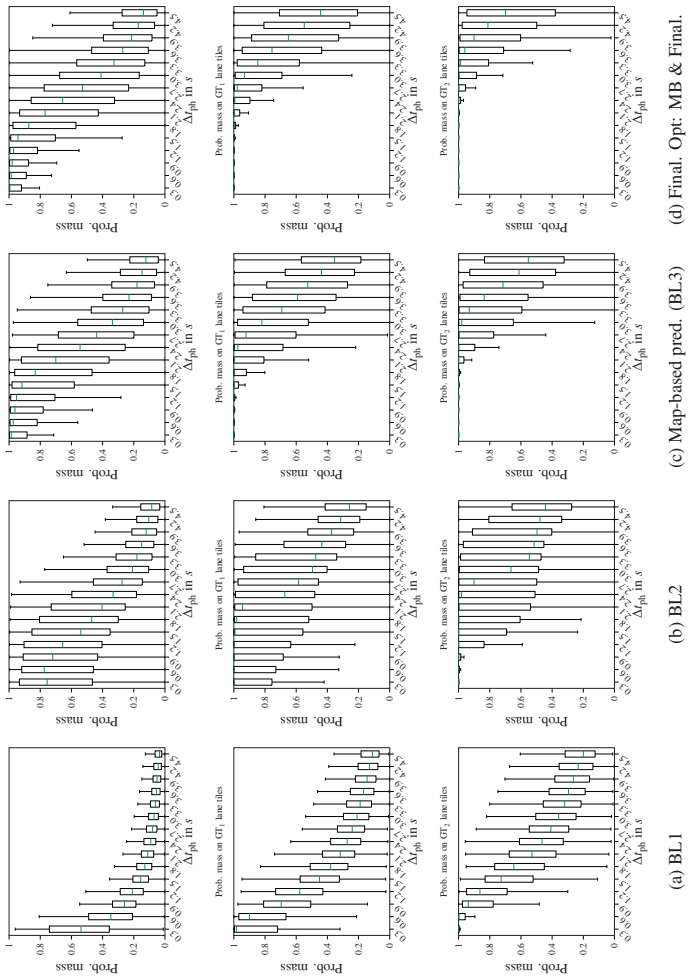


Figure 6.14: Results of different design stages on rounD location1. (a) Naive linear baseline (BL1) with minimal inputs. (b) Naive linear baseline (BL2) with full inputs. (c) Map-based (MB) prediction (if MB and final prediction are both optimized). (d) Final prediction (Final) (both predictions optimized). For all models, the same sophisticated Lane Tile Matcher was used, so the differences only stem from the prediction models. The sharpness of the map-based prediction and the final prediction are surprisingly close to each other.

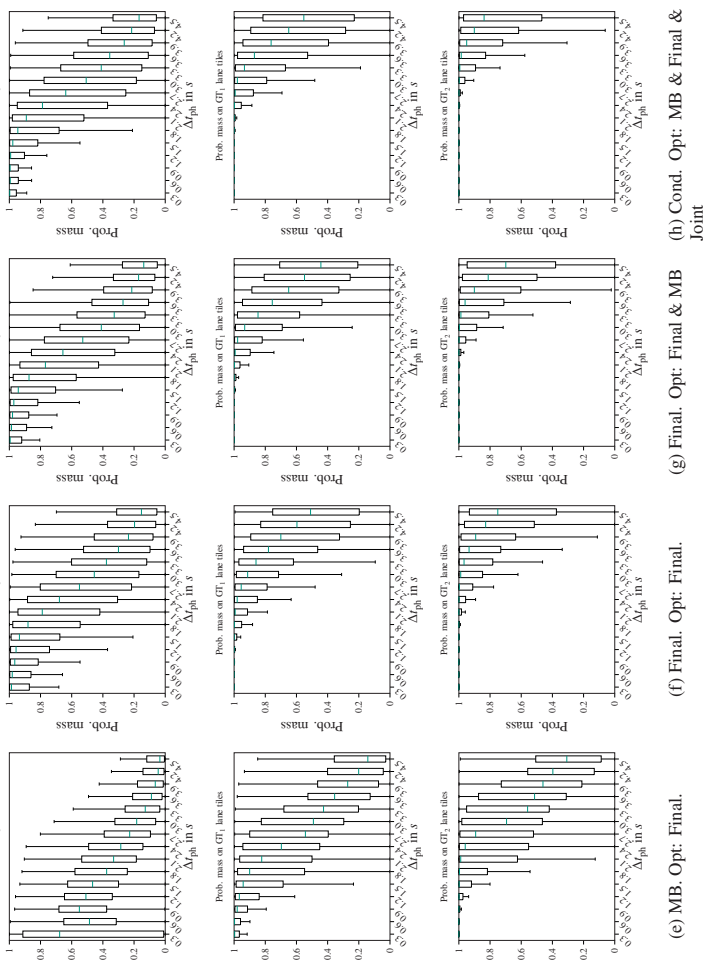
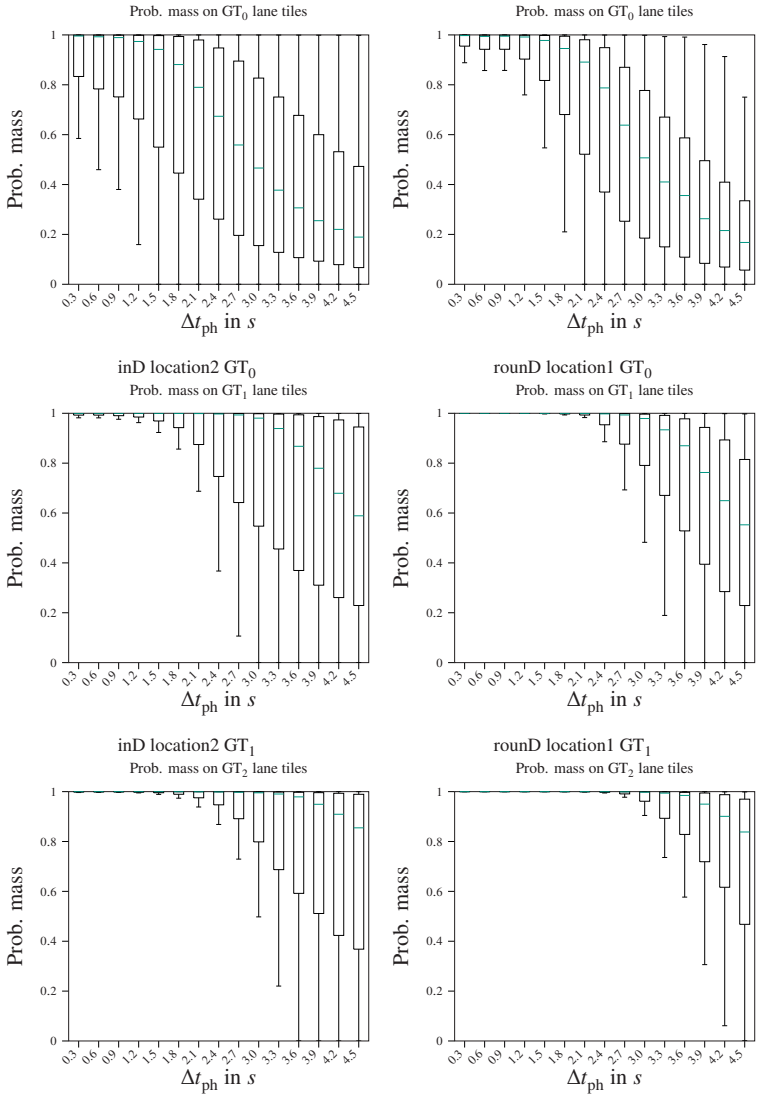


Figure 6.15: Results of different training strategies on roundD location1. (e) Map-based prediction when only final prediction is optimized. (f) Final prediction when only final prediction is optimized. (g) Final prediction when map-based and final prediction both are optimized. For comparison, so (d) of Fig. 6.14 equals (g) here. (h) Conditional prediction (Cond.) evaluated twice for each agent pair given the GT of the respective other. The conditional prediction gives the most accurate results, followed by the final prediction when optimized alone.

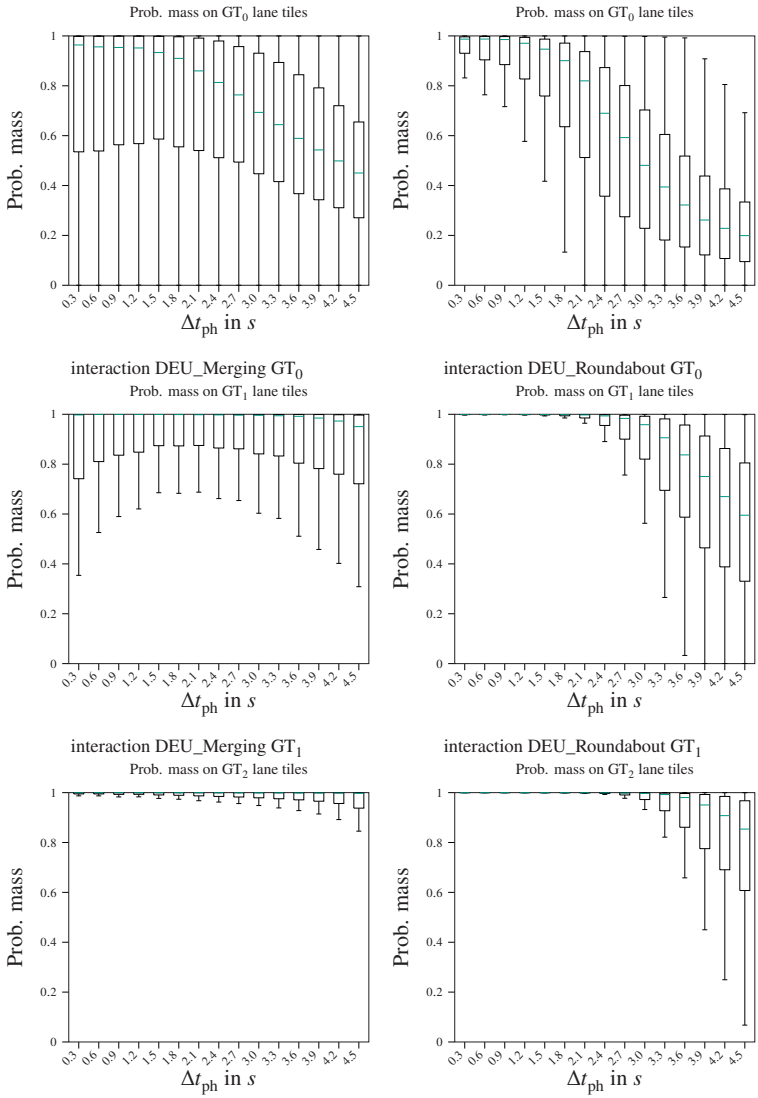
prediction even though most of the agent pairs do not have causal nor numerical interaction is therefore seen as a validation for the superiority of conditional predictions in general, and pairwise bivariate predictions specifically.
A sharpness comparison between the conditional prediction and the prediction under assumption of statistical independence for various training strategies can be found in Fig. 6.15 for rounD location1.
Exemplary results for the conditional prediction are visualized and described in Fig. 6.18, Fig. 6.19 and Fig. 6.20. A detailed description of all qualitative examples is directly in the figures' captions. This way, the reader can compare his or her own interpretation of the illustrated scenes with the given interpretation without jumping between pages.
The first two image pairs (a) and (b) in Figure 6.18 show a roundabout into which two vehicles enter at two consecutive entries. Agent #12 stays within the circuit, so agent #3 has to yield. While the prediction under assumption of statistical independence diverges for agent #12 between circuit and leaving at first exit, the conditional prediction given #12 stays within the roundabout clearly point towards the stop line and indicates that #3 will not enter the roundabout. Shortly before in (c), a third agent #7 leaves the roundabout at the exit of the former of both branches, so agent #12 can directly enter. This also was correctly predicted with the conditional prediction.
The first image pair (a) of Figure 6.19 shows a yield scenario, where an agent #46 turns to the left and has to yield to an oncoming vehicle #49. The intention prediction of the yielding vehicle #46 is ambiguous at first. A second later shown in image pairs (b) and (c), the prediction for the same yielding vehicle #46 has improved significantly, and the maneuver ambiguity has gone. This example suggests preferring conservative behavior options such as waiting in case of ambiguities.
Fig. 6.20 shows two dense merging scenarios. In the first shown in image pairs (a) and (b), vehicles #15, #16 and #17 enter the scene slightly shifted making a merging prediction particularly easy. In the second scenario shown in image pairs (c), (d) and (e), two vehicles #21 and #22 enter the scene right next to each other leading to a scene that is perceived as uncomfortable und difficult by many human drivers. The prediction matches the real evolution of the scene most likely due to different initial velocities of the two agents: The faster vehicle #22 is predicted to merge in front of the slower vehicle #21.



(a) inD location2 GT$_2$

(b) round location1 GT$_2$

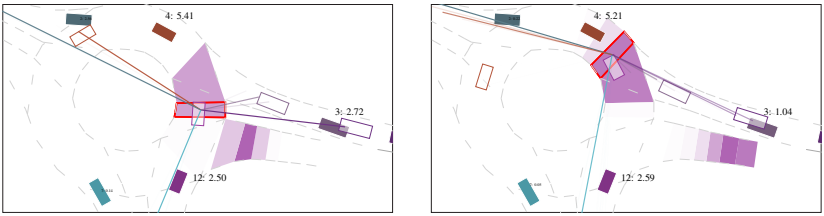
Figure 6.16: Accuracy results for conditional prediction given GT of the other agent, respectively. Left: inD location2. Right: rounD location1



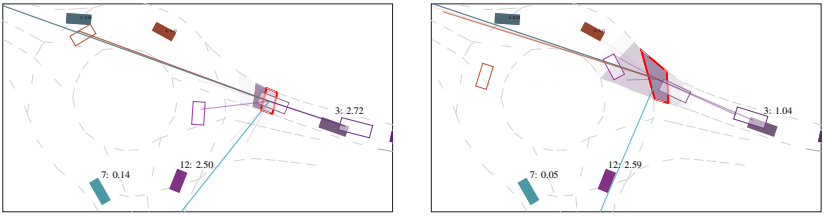
(a) interaction DEU_Merging GT_2

(b) interaction DEU_Roundabout GT_2

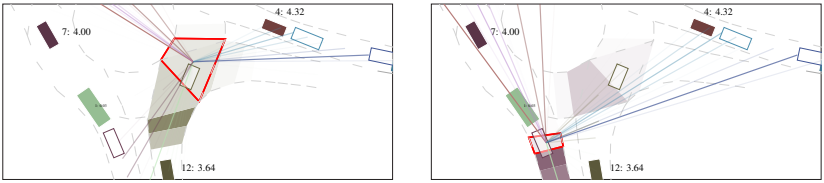
Figure 6.17: Accuracy results for conditional prediction given GT of the other agent, respectively. Left: interaction DEU_Merging. Right: interaction DEU_Roundabout.



(a) The purple car #12 is entering the roundabout and is predicted with around 50% each to stay within the circuit or leave at the first exit. The most likely position prediction for the gray car #3 conditioned on car #12 staying within the circuit is located at the entry's stop line: It is indicated by the gray line starting from the red GT lane tile of #12 pointing towards the roundabout entry.

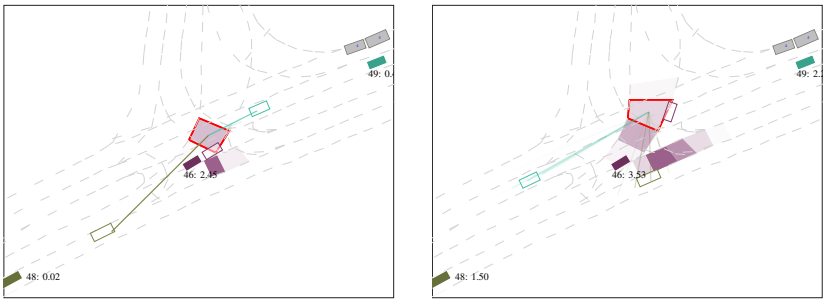


(b) Vice versa, the conditional prediction for the purple car #12 given that gray car #3 slows down at the stop line. Left, 2.1 s: At first, the conditional prob. mass indicated by the purple lines is more or less equally distributed between the two options for #12, namely leaving at the first exit or staying within the circuit. Right, 3.3 s: Since #3 had to brake in order to stay behind the stop line and let #12 pass, the purple lines now only point to the circuit of the roundabout, connecting the two corresponding behavior modes of #3 and #12. Note that the conditional prediction is normalized over all lane tiles in the scene, so the purple lines cannot be explained by leaking prob. mass of #12 at the scene's exit.

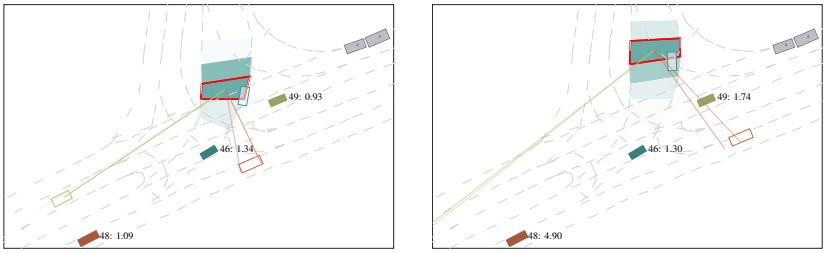


(c) In contrast, the dark brown agent #7 leaves the roundabout and the green agent #12 can directly enter. While the conditional prediction for #7 given the entering of #12 fits well (left, 3.9 s), the conditional prediction for #12 given #7 leaves the roundabout seems undecided (right, 3.9 s). This could result from the asymmetric dependency: #12 depends on car #7, since it would have to yield, but also from any car driving behind car #7. Car #7 can move independently of #12, since it has priority, and independently of a potentially following car. Since conflicts are created and solved pairwise, and conditional prediction is also produced pairwise, the connection to a potential third car might be missing for #12 given only #7 that can move independently.

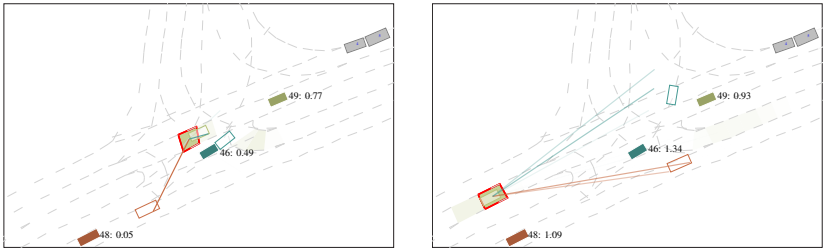
Figure 6.18: Conditional prediction, representative examples 1: interaction DEU_Roundabout. A crowded scene at a roundabout is shown for different agents of interest and different prediction horizons.



(a) Left: 1.8 s. Right: 4.2 s. For the purple car #46, both lane tile matching and position prediction for the acceleration maneuver after the blue car #49 has passed are unsatisfying.

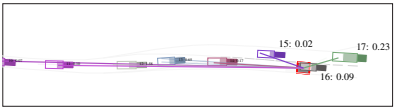


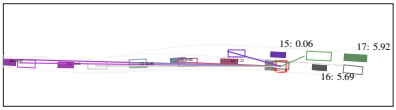
(b) Right: 3.6 s. Left: 4.5 s. A second after, lane tile matching and prediction are accurate again even though green car #49 has still not passed yet. In all four images above, however, the conditional prediction is highly accurate, both for #49 and car #48 passing below, since the lines with the corresponding colors point clearly towards to actual future positions of these agents.



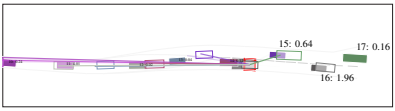
(c) Right: 1.5 s. Left: 3.6 s. Also, the conditional prediction for the blue car #46 and the orange car #48 given the green car #49's GT position fits well to the map graph. However, it may occur that the map graph does not combine the whole road surface that belongs to a certain route option. For the left turn option that #46 chooses, the road was already annotated as being extremely wide. Still, #46 chose a path that leaves the annotated route, which can lead to misleading GT lane tiles.

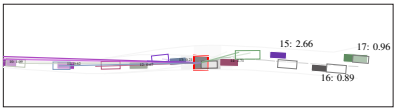
Figure 6.19: Conditional prediction, representative examples 2: inD location4. A sparse scene on a four lane road. The conditional prediction is accurate, even if the position prediction is inaccurate or misleading.



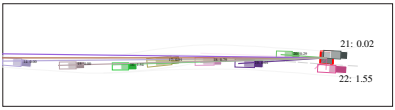


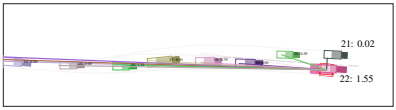
(a) Left: 0.3 s. Right: 1.2 s. A merging situation, where the purple car #15 merges before and the green car #17 merges behind the brown car #16.



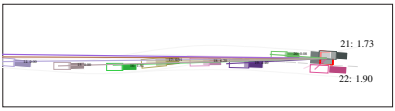


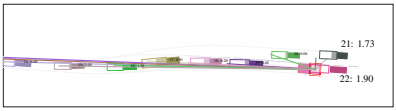
(b) Left: 2.1 s. Right: 3.3 s. The conditional predictions illustrated with purple and green lines clearly and correctly indicate the order of merging for both short und long prediction horizons



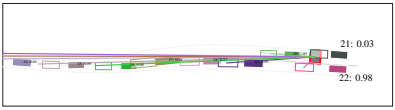


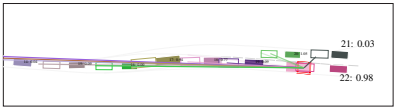
(c) 0.3 s: Some seconds later, the pink agent #22 and the dark green agent #21 enter the scene directly next to each other. Already for very short prediction horizons...





(d) 0.6 s: ...the model correctly anticipates their merging order..





(e) 0.9 s: ...which probably was derived from the speed difference between both cars: The pink agent has traveled a longer distance during the first second and therefore is predicted to merge in front of #21.

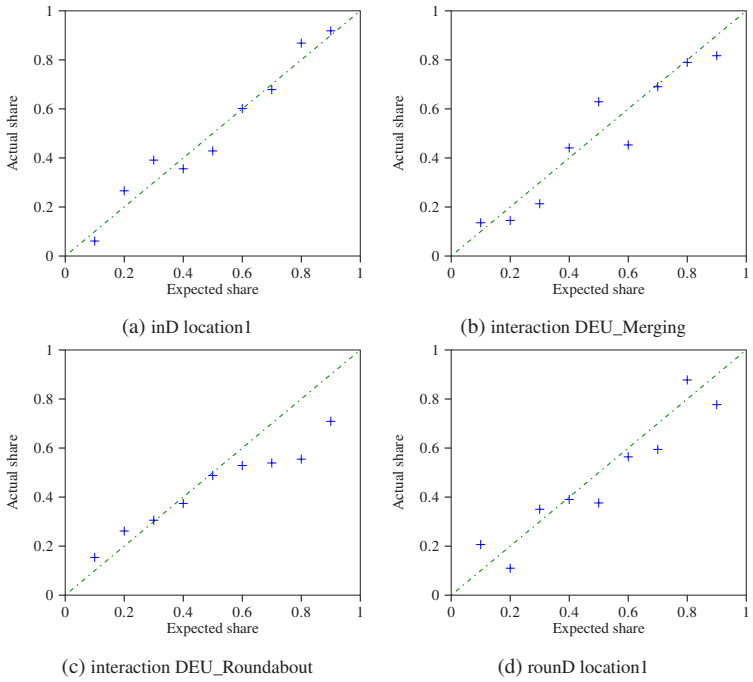
Figure 6.20: Conditional prediction, representative examples 3: interaction DEU_Merging. Two lanes merge into one shortly after an intersection with dense two lane traffic. (a) and (b) show four different prediction horizons for a first merging scenes. (c), (d) and (e) show two agents #21 and #22 at three different prediction horizons of a second merging scene. In all shown examples, the conditional prediction indicated through colored lines qualitatively matches the actually driven merging order.

6.6.2 Quantitative Evaluation of Reliability

In Fig. 6.21, the reliability plots for four representative locations are shown. Again, points + above the symmetry axis can be interpreted as underconfidence for the corresponding interval because the observed GT lane tiles occur more often than expected from the prob. mass assigned to them. Similarly, points below represent overconfidence of the model relatively to the prob. mass assigned to the GT lane tiles.
For interaction DEU_Roundabout, slight overconfidence for large occupancy probabilities can be noticed. Compared to the predictions of statistically independent positions in Fig. 6.13, the reliability generally decreases, since the mean absolute difference between the expected share and the actual share increases. Since there is no clear under or overconfidence over several interval bins, it is assumed that reliability was improved if more diverse data was available for training. For application, the increase of statistical reliability has to be considered, but since the uncertainty influences onto prediction results are manifold, the result is regarded as a proof of concept.

6.6.3 Discussion

A unique dataset was created from subsets of three publicly available drone datasets. It only contains data that suits the design requirements formulated together with the planner team at Institut für Mess- und Regelungstechnik (MRT): Since we drive in Germany only, the network should not be confused by different traffic rules of different countries. Second, the road should be tight enough to only one vehicle laterally fits onto it. Third, TPs should obey to traffic rules, so unstructured traffic shall be excluded.
This unique dataset makes it particularly difficult to compare the presented results with state-of-the-art solutions. Also, while most other researchers focus on trajectory and position prediction due to the input requirements of popular benchmarks, the proposed model outputs rather intentions than positions. An additional sampling problem would need to be solved in order to participate in benchmarks and make the proposed approach comparable.
Instead, popular metrics have been adopted to the discrete problem formulation. Also, different design stages and results of different training strategies have been juxtaposed in order to highlight their advantages.
However, it is recommended to focus particularly on two significant results rarely found in literature at all.



(a) inD location1

(b) interaction DEU_Merging

(c) interaction DEU_Roundabout

(d) rounD location1

Figure 6.21: Probabilistic reliability plots of conditional prediction given GT of the other agent.

First, the probabilistic reliability was verified for both the predictions under assumption of statistically independent future positions and for the conditional prediction. These results clearly indicate that it is valid to post-process the predicted intentions in a probabilistic way without fearing to systematically over- or underestimate probabilities. These results have been achieved without a consecutive statistical calibration step likely due to the following design decisions: The logarithmic score used in cross-entropy has been identified by Gneiting *et al.* [38] as a scoring rule that optimizes sharpness of distributions while preserving statistical calibration. Also, effort has been put into keeping the loss in the same scale regardless of the number of agents and the number of lane tiles in a scene. Last, propagating probability mass though the map graph instead of designing a network that can arbitrarily shift and position probability mass on the map graph resulted in smooth probabilistic transition.

Second, the many qualitative examples presented in this chapter highlight important takeaways for prediction. Assuming statistical independence of future positions of agents that require interaction in order to solve positional conflicts is highly questionable, if not clearly wrong. In many examples, it is hard to interpret whether the statistically independent prediction is reasonable considering the (many) different behavior evolutions of a scene. The shown conditional predictions clearly indicate an advantage of predicting pairwise bivariate distributions over predicting univariate discrete distributions for each agent. It is therefore recommended studying the qualitative examples thoroughly.

7 Conclusion

An artificial graph neural network (GNN) is presented which predicts a graph of interacting agents probabilistically on a map graph. All entities of both graphs can have their individual input variables. Besides the commonly used individual prediction for each traffic participant, the model predicts joint distributions for all pairs of agents. From these, conditional predictions for one agent given the hypothetical future position of another agent can be computed in order to analyze possible future interplay between two agents.
The model consists of several modules. The tasks of modules within the GNN are based on task components a human driver must also handle when predicting the behavior of other TPs. These tasks are determined by strict modeling of the information flow within the model: First, the agents' future positions are predicted exclusively based on the map. In this step, each agent can collect information about the map. From the map-based prediction, future position conflicts are identified for each pair of agents. Next, information is exchanged between agents to help resolve the positional conflicts. Finally, agents' positions are predicted individually and in pairs based on the map and the exchanged information. The results were successfully analyzed for prediction accuracy at different design stages and for statistical reliability.
Regarding the requirements listed in Section 1.2 at the beginning of this work, the following conclusions w.r.t. to our model can be drawn.

7.1 Degree of Requirement Fulfillment

Uncertainty Estimation

The proposed model probabilistically allocates agents to lane tiles and then propagates them probabilistically through the map graph (see evaluation Fig. 6.4). Probability mass leaking from the scene is separately penalized in the loss term (see Eq. (5.20)). Missing ground truth (GT) occasions – e.g. when an agent has left the scene – are compensated in order to smoothen the resulting gradients (see Eq. (5.23) and Eq. (5.27)). Unlike many com-

peting approaches that use a fixed number of parameterized distributions – e.g. a Gaussian mixture –, the proposed model outputs a discrete probability distribution over all lane tiles that was validated for reliability (see Fig. 6.13). Additionally, discrete joint distributions of pairs of agents are predicted and their derived conditional distributions are validated w.r.t. statistical reliability (see Section 6.6.2).

Multi-Modality

The proposed model allows positional prob. mass to flow along the map graph (see Section 5.2.2). There are three modeling assumptions that restrict a complete arbitrary flow of prob. mass limiting the model's ability to generate an arbitrary number of modes.
First, it is assumed that a discrete distribution over all lane tiles in the map graph can approximate the agent's position sufficiently. Sub-lane-tile accuracy can only be derived by fitting a parametric distribution e.g. in Frenet coordinates in a post-processing step.
Second, discrete model transition options between those lane tiles are introduced, restricting prob. mass from flowing in unlikely directions and from leaving the map graph elsewhere than at entry and exit lane tiles. This has advantages regarding temporal association of modes for reconstructing coherent trajectories. The disadvantage is that agents cannot be predicted outside the map graph or other than the predefined motion transition options. Overtaking on the opposite lane for example is currently not included, even though it regularly occurs on country roads.
Third, prob. mass leaked from a scene cannot return into the scene.
Other than that, the distribution can evolve freely which allows important prob. mass splits, namely at points where the road forks, and at stop lines, where prob. mass might divide into a mode corresponding to a decelerating behavior option and a mode corresponding to an accelerating behavior option. In particular, prob. mass can flow independently of the hidden states of an agent (see Eq. (5.3)).
The joint distribution is generated through agents' states on lane tiles whose evolvement is also restricted by transition options (see Section 5.2.8).

Ability to Handle Different Traffic Scenes and Generalize Accordingly

The proposed model consists of an agent graph whose nodes move along a map graph (see Chapter 4). The model can be seen as two constantly interacting GNNs, making it invariant w.r.t. the number and the order of agents and lane tiles. The model consists of several modules that strictly limit the information flow at inference, allowing it to generalize from a limited set of data – especially w.r.t. the number of maps – as was validated throughout Chapter 6 by defering particular maps during training.
Since only 12 maps have been identified as suitable and some represent unique scenarios, the limited data is regarded as the main bottleneck at this point. Also, there are challenging scenarios that are not covered by the available data. Examples are multi-lane roundabouts and a narrowing (road bottleneck).
Given those limitations, the presented results especially of the general statistical reliability and the accuracy of the Lane Tile Matcher (see Section 6.4) and the Cooccurrence Estimator (see Section 6.6.1) are regarded as strong evidence for the model's robustness achieved through a parameter-efficient, modularized design.

Parameter Parsimony

In order to fulfill the former requirement of generalization, the model was designed slenderly. Finally, fewer than 27,000 trainable parameters are used, limiting the number of parameters to about a third of VectorNet by Geo *et al.* [33] who already highlighted the parameter parsimony of their model.
Since an extensive hyperparameter optimization was renounced, the number of parameters is expected to be reduced further without application-relevant decrease of performance.

Computational Efficiency

The proposed method utilizes matrices that depend quadratically on the number of agents and quadratically on the number of lane tiles making its complexity with respect to number of computations and memory $\mathcal{O}(n_{\mathrm{L}}^2 \cdot n_{\mathrm{A}}^2)$. In most cases, the number of directly relevant agents and the number of directly relevant lane tiles are strongly limited – not only contextual, but also w.r.t. sensor range and occlusions (see Section 6.2).
Therefore, an issue regarding applicability is not seen here.

Complete Input Information

Environmental information is given to the proposed model through several tensors (for an overview, see Appendix A.5). Most importantly, agents have individual features, their pair-wise relationship can be represented through individual features, lane tiles have own features, and their connections as well. Traffic rules and traffic entities can therefore be given in various ways. It is possible to introduce a priority feature for an agent, e.g. for emergency vehicles. It is possible to introduce a priority feature through the (directed) edges between two agents in case a former module in the automated driving (AD) stack derived priority relations from the map. It is possible to implicitly indicate traffic rules through map entity features representing the location of a road sign.
Besides traffic rules w.r.t. vulnerable road users (VRUs) that are not regarded in this work, traffic rules that are not representable in the proposed model do not come to mind. Four way stop behavior is learnable from intersection examples with stop lines and stop signs just like priority to the right for German traffic where four way stops do not exist.
Past trajectories that many competing approaches utilize could be given as agent features, even though they were not used this way throughout this work.

Flexible Prediction Format

The output distributions are strictly map-referenced. Therefore, they can be analyzed for high-level behavior options or behavior modes. Since the output distribution was checked for reliability (see Section 6.5.3), probabilistic reliability of behavior options is ensured. For example, the prob. mass within a certain branch can be summed up in order to obtain the probability of an agent taking said branch at a certain future time step.
Also, trajectory information can be reconstructed since prob. mass is strictly limited to flow through predefined motion transition options. This way it is ensured, that modes between two consecutive future time steps can be associated. For example, a Gaussian in Frenet coordinates can be fit to each positional mode of an agent. Over the prediction horizon, modes can split. Through nearest neighbor association between two consecutive time steps, corresponding modes representing a trajectory can be extracted, and a smoothed trajectory can be optimized to those modes over time. This way, trajectories can be extracted for planning through optimization, just as many other approaches directly offer them.

Conditional Prediction

Besides predicting position distributions individually for each agent under the questionable assumption of statistical independence, joint distributions are predicted for pairs of agents (see Section 5.2.8). By deriving the conditional distribution for an agent of interest given a hypothetical position of another vehicle, e.g. the ego vehicle, this joint distribution can be analyzed for hypothetical future evolvements of the traffic scene.
While some competing approaches predict consistent modes for all agents in a traffic scene, the presented work is one among the first that specifically allows analysis for arbitrarily hypothetical behavior between agents. Also, the proposal of predicting joint distributions for pairs of agents is a simple solution that can be integrated into many other approaches.

7.1.1 Limitations

The presented approach specifically excludes unstructured road topologies and road traffic where road markings are either not given, not obeyed, or sparse enough to not give a strong path prior. East Asian urban traffic is a famous example for unstructured traffic, but also roads wide enough for more than one traffic participant (TP) in one direction, as they occur frequently in the US, are not an advantageous use case for the proposed method.
Since agents are predicted on the map graph only, the model is restricted to lane-bound vehicles, including motorcycles, cars, trucks and busses. Pedestrians and cyclists can be included by adding sidewalks to the map graph.
The goal of every TP remains unknown for prediction, whereas each TP predicts others from an ego-centric viewpoint. This means every driver knows where he or she is heading to. Not utilizing the knowledge about desired destinations introduces large uncertainty about the evolvement of a scene and separates the research field of agent prediction from the research field of multi-agent planning. The present prediction approach is designed as a prediction method for traffic scene evolvement without centering prediction around an ego vehicle. An ego trajectory is not utilized.
Within the prediction horizon Δt_{ph}, new agents might enter the scene. These are not considered in the problem formulation and lead to prediction errors especially for shortly mapped access routes of intersections, where agents directly participate in conflicts after entering the mapped scene as for example at location1 from inD.

7.2 Future Directions

As for most scientific problems, new challenges arise with the solution of another. Therefore, future directions shall be outlined as inspiration for other researchers.
One well-known problem refers to holistic prediction metrics. While agreeing with assessments from [38, 111, 119] stating that both sharpness *and* statistical reliability need to be assessed quantitatively and equally, existing prediction metrics might not be sufficient to tell apart good and bad prediction approaches. Many TPs drive proactively and can already be predicted quite well with a physical or map-based model along a lane for a couple of seconds. Therefore, a sharpness metric needs to be weighted with the intensity of an upcoming position conflict. Such a conflict identifies situations where interaction is required in order to solve the conflict. Unlike many vehicle-following scenarios, that are neither interesting nor critical for anyone, but occur most of the time in all datasets.
Another research topic is about what prediction assessment and quality is necessary for the automated vehicle (AV) to drive "well" or "safely" or "as good as a responsible human". The next recommended step if research regarding prediction was continued, is to adapt a vehicle's planning module, so it is able to make use of the output of the proposed model. Maybe the question is not how to measure and assess prediction quality and define a threshold, so an AV is expected to be able to drive with it. Maybe a prediction model is "sufficient" if an AV using the model can drive comfortably, safely and not overcautiously with the model's output.

Besides these general concerns regarding the field of prediction for AD, the following should be tackled in the future regarding the proposed model.
Cutting lanelets into smaller lane tiles and utilizing matrix operations was an important step for identifying an efficient conflict representation. This makes the output distribution flexible compared to Gaussian mixtures that are often used, and to predict joint distributions for pairs of agents. However, the representation is a compromise between rasterized input and directly using lanelets from a given map. There surely is a possibility of using raw lanelet maps, probably with lanelet-wise coordinate frames, and introducing as well as solving pair-wise conflicts similarly as proposed for a map discretized in

lane tiles. Also, joint distributions can be predicted for pairs of agents located on lanelets instead of lane tiles.
Finding a loss function that weights cost based on graph distance to GT is required, both for lane tile maps or lanelet maps.
Next, having more data was beneficial in several regards. First, more trajectories per scenario are beneficial. Second, more scenarios are required for better generalization. Last, limited sight was observed to be a frequent problem for some intersections with shortly mapped road branches, as described before. In practice, modeling occlusions properly might be the key for avoiding prediction artifacts due to limited view.
In Section 5.1, the model design is described as strictly aligned with how humans predict other TPs. One feature that is not utilized in the model is knowledge about what each agent is actually planing to do. In the presented model, agents are predicted from the viewpoint of a sovereign viewer that does not know – neither a priori, nor a posteriori – what agents plan to do. However, during driving, each TP knows quite precisely his or her destination. Such knowledge of course can only be introduced during training since it will not be available at inference. One possibility might be to only train with the losses on the lane tiles of the track each TP did actually drive. Only the behavior on the desired path helps to understand the concrete longitudinal motion of the TP, since the track was already specified by the destination of the TP and known to the TP when entering an intersection. For a multi-lane road this assumption might not hold, since lane changes are often the result of spontaneous behavior decisions. But for many other cases, the destination and the path of a TP does not change within the current scenario.

7.2.1 Towards a Holistic Prediction Framework

Generally, different classes of TPs do not necessarily need to be predicted with a prediction framework that models interaction between all agents at all times explicitly. For short prediction horizons of up to 1.5 s, predicting a physical model might be enough to directly avoid crashes in the last moment. Physical models vary according to the transportation system. While for pedestrians a 2D constant velocity Kalman filter (KF) might be useful, lane bound vehicles such as cars might be best predicted with a constant acceleration, constant turn rate model. For physically more challenging vehicle models such as bicycles, motorbikes and electric scooters, specific models for state estimation can be

used as proposed by Wirth *et al.* [112]. This two-wheeler state then allows for model-based prediction as presented by Wirth *et al.* [113].
For long-term prediction of several seconds, probabilistic models are required. In the case of pedestrian movement, single agent predictions might already be good enough as presented by Rehder *et al.* [87] and Lorenzo *et al.* [1].
Prediction of lane bound vehicles in the long-term is recommended to be done with a conditional behavior prediction approach as proposed in the presented work that explicitly incorporates interaction.

8 Acknowledgement

The author acknowledges support by the state of Baden-Württemberg through bwHPC.

A Appendix

A.1 Datasets

The presented approach works best if a couple of requirements regarding the training dataset are fulfilled. They are listed in Section 6.1.
Some intersections from INTERACTION dataset [120], rounD [55] and inD [11] fulfill those requirements. The chosen intersections are described in detail in the following. Besides knowing the map topology for visual inspection of prediction examples, it is important to know about the surrounding of the corresponding intersection. For getting an overview, aerial images of the intersections are shown in Fig. 6.1. All datasets provide the necessary static and dynamic features that directly give or at least can easily be transformed to the inputs features listed in Section 4.2 and Section 4.1.6.

A.1.1 inD

From inD, all intersections (location 1-4) are used.
Location 1 with geographic coordinates N50.7820°, E6.0712° is an urban intersection of two tertiary roads, priority road Süsterfeldstraße and Kühlwetterstraße, on which traffic has to yield. One branch of Kühlwetterstraße (south) is a dead end with barely any traffic. All branches have a speed limit of 50 km/h. Traffic is not too intense, pedestrians and cyclists rarely occur.
Location 2 with geographic coordinates N50.7684°, E6.1022° is an urban intersection of Bismarckstraße with its two branches Schlossstraße and Rehmannstraße. Priority is giving to the right. The speed limit is 30 km/h. On the eastern branch of Bismarckstraße, there is a zebra crossing which is strongly used by pedestrians. Cyclists also occur often. Visibility between the intersection's branches is strongly reduced by buildings close to the intersection and many parked cars. Due to the strong vulnerable road user (VRU) traffic that is not covered by the presented approach, results on this scenario have to be assessed with care.

Location 3 with geographic coordinates N50.7786°, E6.1654° is an urban intersection where Heckstraße joins Von-Coels-Straße, and the latter has priority. The speed limit is 50 km/h for all branches. On the south branch of Von-Coals-Straße and at Heckstraße, there are pedestrian crossing paths that are not zebra crossings, but they are rarely used.
Location 4 with geographic coordinates N50.7852°, E6.1311° is an intersection where Neuköllner Straße, that comes from an industrial zone, joins Charlottenburger Allee, which is connecting the motorway Berlinger Ring with a residential area. The speed limit is 50 km/h for both roads despite the Charlottenburger Allee is a non-urban four lane road where many vehicles exceed the speed limit. Charlottenburger Allee is the priority road. Directly at the intersection there are two bus stops that frequently interfere with the traffic flow.

A.1.2 rounD

Location 1 with geographic coordinates N50.7906°, E6.0600° is an urban roundabout with four branches: the urban roads Süsterfeldstr (south) and Kackertstraße, the rural road Süsterfeldstraße (west) and to the north it is connected to the motorway Toledoring/Pariser Ring. Priority is given as usual for roundabouts in Germany: whoever wants to enter the roundabout has to yield. Speed limit is 50 km/h to all direction. In the north branch leading to the motorway, the speed increases so traffic is accelerating and decelerating, respectively. VRU traffic is possible but negligible. Bus traffic occurs frequently.
Location 2 with geographic coordinates N50.8739°, E6.1067° is a roundabout connecting the urban main road Geilenkirchener Straße with the side roads Thiergartenstraße and Ritzerfelder Straße. On the main road, 50 km/h are allowed, on the side roads, the speed limit is 30 km/h. VRU traffic is possible but exists only occasionally. Right in front of the roundabout to the southwest, there is a bus stop that interferes with traffic flow at times.

A.1.3 INTERACTION

Unlike the other datasets, INTERACTION is anonymized in the sense that it does not contain geo-referenced coordinates. The location names of INTERACTION are cryptic but contain at least letters for the country and a semantic

description of the scene. The meaning of the remaining letters is unknown, so they are omitted.
Location DEU_Roundabout is a roundabout in an industrial district. All of its branches have a speedlimit of 50 km/h. Both VRU and bus traffic happen occasionally. Bus stops are nearby, but interferes less frequent with normal traffic compared to the other locations. The right branch leads to a motorway, the left branch leads to the town center, and the lower branch leads to an industrial area and is therefore used a bit less frequent.
Location DEU_Merging is a standalone merging scenario with a speed limit of 50 km/h. Quite often, traffic slows down after merging due to the upcoming signalized intersection. There is a separate bike lane that does not interfere with road traffic. The merging scenario is subsequent after a large urban intersection, so vehicles often take a slow turn into the merging area.
Location CHN_Merging is a highway scenario with a merging scenario at the top and the bottom. Traffic is usually dense and slow, also, lane bounds are strictly obeyed, so it is considered suitable for the present use case even though it was recorded in China. The scenario is split into three separate scenarios. The _lower merging scenario with traffic flowing to the right is separated. It contains an asymmetric merging situation from an acceleration lane onto the two lane motorway. The _upper scenario is a symmetrical merging of two lanes flowing to the left. It is spatially independent of the _middle scenario, where two lanes are going to the left.
The speed limit is given with 80 km/h.
Location USA_Roundabout is a wide roundabout where lane bounds are also usually respected. Despite being crossed by several broad crosswalks, those are barely used. Also, inner traffic has priority, so it aligns with the other roundabouts. The speed limit is given with 25 mph $\approx$ 40 km/h.

A.2 The Graph Spectral Domain

The following specification is necessary for comprehension of Section 3.1. The degree matrix $\mathbf{D} \in \mathbb{N}^{n_\mathrm{N} \times n_\mathrm{N}}$ is a diagonal matrix whose trace sums up the connection of the corresponding node i with $\mathbf{D}_{i,i} = |\mathcal{E}_i|$. For directed graphs, in- and out-degree matrices are distinguished. The Laplacian matrix $\mathbf{Q} = \mathbf{D} - \mathbf{A}$ is the difference between degree and adjacency matrix. In- and

out-degree Laplacians can be distinguished as well. For the Laplacian matrix, an eigendecomposition

$$\epsilon^m = \mathbf{v}^{m\mathrm{T}}\mathbf{Q}\mathbf{v}^m \tag{A.1}$$

can be conducted where $\mathbf{v}^m$ is an eigenvector corresponding to eigenvalue ϵ^m. The index of eigenvalues m orders the eigenvalues by absolute value, starting with the largest. With the matrix of ordered eigenvectors $\mathbf{V} = [\mathbf{v}^m] \in \mathbb{R}^{n_\mathrm{N} \times n_\mathrm{N}}$ a signal $\mathbf{x}$ can be transformed into the spectral domain with

$$\mathbf{x}_{\text{spectral}} = \mathbf{V}^\mathrm{T}\mathbf{x}. \tag{A.2}$$

The Fourier transformation of graphs is valuable for conducting the operation of discrete convolution which has shown to be useful for many deep learning applications. Convolution defined in the spatial or temporal domain is equivalent to multiplication in the respective Fourier domain.
Now, the equivalent operation shall be derived for graphs. In graphs, the node feature vector $\mathbf{n}$ is the signal and the convolution takes place with a weight vector $\mathbf{w}$ of the same size. Both node feature and weight can be transformed to the Fourier domain, multiplied with each other and transformed back into the spatial domain according to

$$\tilde{\mathbf{n}} = \mathbf{V}(\mathbf{V}^\mathrm{T}\mathbf{n} \odot \mathbf{V}^\mathrm{T}\mathbf{w}). \tag{A.3}$$

A.3 Hyperparameters

The hyperparameter used for evaluation have shown to be reasonable during the development of the model. Thus, it is expected that intense and structured hyperparameter tuning can still improve the model's performance.
In Table A.1, the hidden feature sizes are listed. The Conflict Identifier consists processes inputs $\mathbf{C} \in \mathbb{R}^{2 \times n_\mathrm{T} \times n_\mathrm{T}}$ of two convolutional layers and one fully-connected layer. The first conv. layer has 20 filter masks of size $[2 \times 6 \times 6]$ and a stride of 3, the second conv. layer has 15 filter masks of size $[2 \times 2]$. The fully-connected layer maps each 3×3-sized feature to a 1×1-vector element of size 15. In Table A.2, hyperparameters not directly related to module architectures are listed.

Table A.1: Hyperparameters of the Proposed Modules

Module	Layer type and hidden features
Lane Tile Matcher	MLP $[24, 15]$
State Estimator	MLP $[20, 20]$
Motion Transition Estimator	MLP $[24, 12]$
Conflict Identifier	CNN $[2 \times 6 \times 6, 20]$, $[2 \times 2, 15]$, $[3 \times 3, 15]$
Notification Extractor	MLP $[22, 12]$
Message Aggregator	MLP $[22, 22]$
Self-Notification Extractor	MLP $[20, 12]$
Cooccurrence Estimator	MLP $[12, 12]$
Δt_{ph}	4.5 s
n_{T}	15

Table A.2: Non-Module related Hyperparameters

Name	Value
Activation function	ELU [25]
Optimizer	Adam [53] (standard settings)
Weight for l2-regularization	10^{-7}
Training steps (each step is one scene)	$150,000$
Weight initialization	Gaussian with $\mu = 0, \sigma = 0.02$

Table A.3: Overview of all agent features and agent edge features. They all are scalars.

Feature name	Meaning
v	Speed
a	Acceleration
w	Width
l	Length
omega_dot	Yaw rate
sin_omega	Sine of agent orientation
cos_omega	Cosine of agent orientation
distance	Euclidean distance between two agents
sin_alpha	Sine of yaw angle between two agents from the view point of first agent
cos_alpha	Cosine of yaw angle between two agents from the view point of first agent
sin_beta	Sine of angle between longitudinal axes
cos_beta	Cosine of angle between longitudinal axes

A.4 Input Features

In Fig. 4.7, input features used for lane tiles and lane tile connections are listed. Experiments have been conducted with additional features that have not been used for evaluation. Those are written in blue. The same holds for input features for agents and agent edges that are listed in Fig. 4.8.

Table A.4: Overview of all lane tile features and lane tile connection features. They all are scalars. See visual explanation in Fig. 4.7.

Feature name	Meaning
l_centerline	Length of lane tile centerline
w_begin	Width of lane tile beginning
w_end	Width of lane tile end
area	Area of lane tile
sin_phi	Sine of lane tile orientation
cos_phi	Cosine of lane tile orientation
v_max	Speed limit
l_be	Distance between beginning of first and end of second lane tile
sin_omega	Sine of connection line between beginning of first and ending of second lane tile
cos_omega	Cosine of connection line between beginning of first and ending of second lane tile
angle_be	Relative angle between centerlines
T_sin	Sine component of 2D transformation matrix between beginning of first and beginning of second lane tile pose
T_cos	Cosine component of 2D transformation matrix between beginning of first and beginning of second lane tile pose
T_x	Longitudinal component of 2D transformation matrix between beginning of first and beginning of second lane tile pose
T_y	Lateral component of 2D transformation matrix between beginning of first and beginning of second lane tile pose
stop_line	1, if agents have to pass a stop line to get from one lane tile to the next, else 0
yield_sign	1, if agents have to pass a yield sign; 2, if agents have to pass a stop sign to get from one lane tile to the next

A.5 Input and Output Summary

Compact lists of variable names, their meaning and their shapes in Table A.5 for the inputs and in Table A.6 for the outputs are given below. A detailed explanation is given throughout Chapter 4.

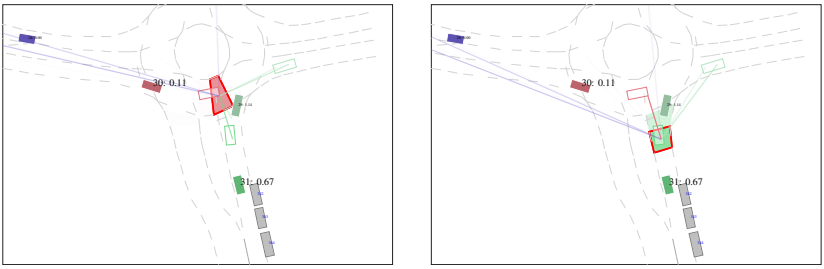
Table A.5: Summary of all input tensors used for inference.

Feature	Meaning	Shape
$\mathbf{P}$	Matrix of positions (optional, option 1)	$\{0,1\}^{n_\mathrm{L} \times n_\mathrm{A}}$
$\mathbf{P}_\mathrm{Cand}$	Matrix of position candidates (optional, option 2)	$\{0,1\}^{n_\mathrm{L} \times n_\mathrm{A}}$
$\mathbf{F}_\mathrm{A}$	Matrix of agent features	$\mathbb{R}^{n_\mathrm{A} \times n_\mathrm{F,A}}$
$\boldsymbol{F}_\mathrm{AE}$	Tensor of agent edge features	$\mathbb{R}^{n_\mathrm{A} \times n_\mathrm{A} \times n_\mathrm{F,AE}}$
$\mathbf{F}_\mathrm{L}$	Matrix of lane tile features	$\mathbb{R}^{n_\mathrm{L} \times n_\mathrm{F,L}}$
$\boldsymbol{F}_\mathrm{LE}$	Tensor of lane tile connection features	$\mathbb{R}^{n_\mathrm{L} \times n_\mathrm{L} \times n_\mathrm{F,LE}}$
$[\tilde{\mathbf{t}}^{a,l}]$	Agent-lane-tile 2D transf. info (optional, option 2)	$\mathbb{R}^{n_\mathrm{A} \times n_\mathrm{L} \times 4}$
$\mathbf{A}$	Matrix of consecutive lane tile connections	$\{0,1\}^{n_\mathrm{L} \times n_\mathrm{L}}$
$\mathbf{R}$	Matrix of right lane tile connections	$\{0,1\}^{n_\mathrm{L} \times n_\mathrm{L}}$
$\mathbf{L}$	Matrix of left lane tile connections	$\{0,1\}^{n_\mathrm{L} \times n_\mathrm{L}}$
$\mathbf{\Pi}$	Matrix of overlapping lane tiles (crossing matrix)	$\{0,1\}^{n_\mathrm{L} \times n_\mathrm{L}}$
n_A	Variable number of agents in the scene	$\mathbb{N}$
n_L	Variable number of lane tiles in the scene	$\mathbb{N}$

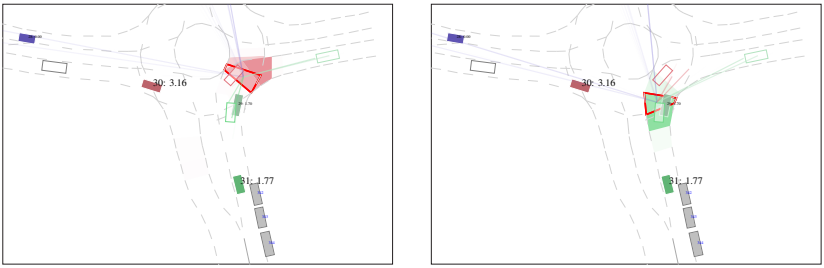
Table A.6: Summary of all output tensors received after inference.

Feature	Meaning	Shape
$\boldsymbol{P}_\mathrm{Map}$	Position prediction tensor *without* interaction	$[0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A}}$
$\boldsymbol{P}_\mathrm{Final}$	Position prediction tensor *with* interaction	$[0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A}}$
$\mathbf{\Xi}$	Joint position prediction tensor with interaction	$[0,1]^{n_\mathrm{T} \times n_\mathrm{L} \times n_\mathrm{A} \times n_\mathrm{L} \times n_\mathrm{A}}$

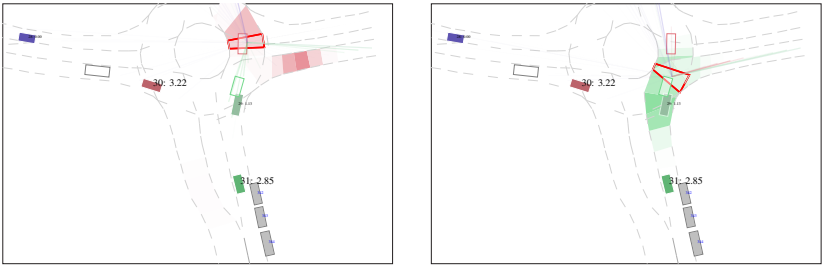
A.6 Further Examples



(a) 2.1 s: While the prediction of car #30 fits quite well...

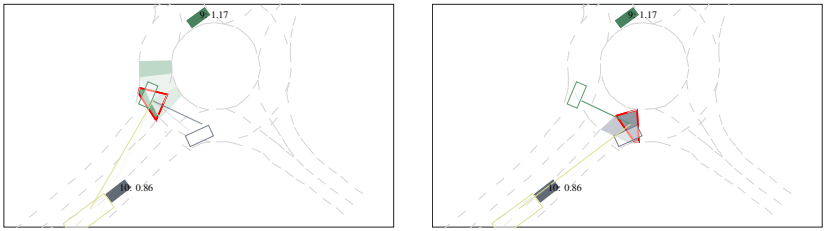


(b) 3.3 s: ...the prediction for car #31 does not fit for horizons larger than 3.3 s.

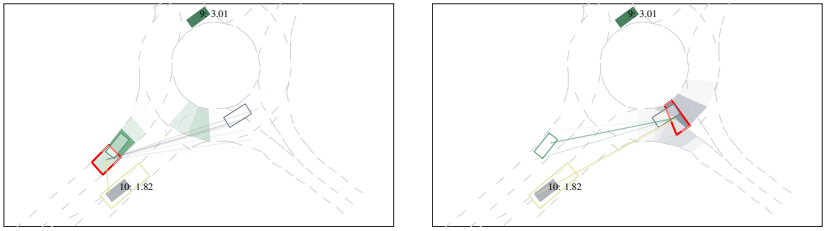


(c) 4.5 s: However, the conditional prediction fits well for both cars and all prediction horizons.

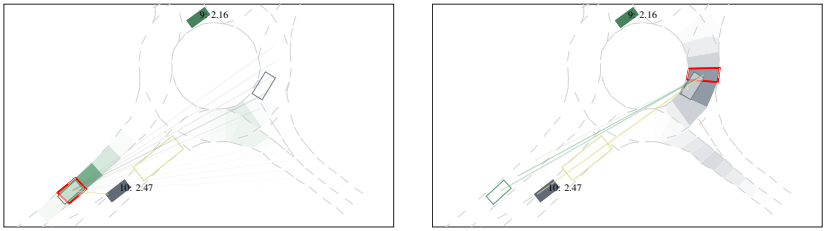
Figure A.1: Conditional prediction, representative examples 4: rounD location1.



(a) 2.1 s: Both the conditional prediction for car #10 and for car #9 fit well.

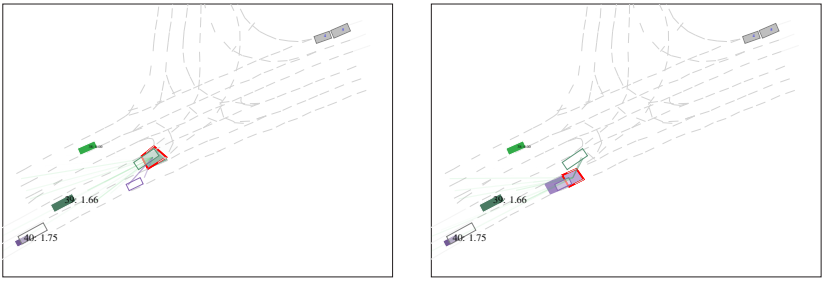


(b) 3.3 s: Also, the prediction seems reasonable...

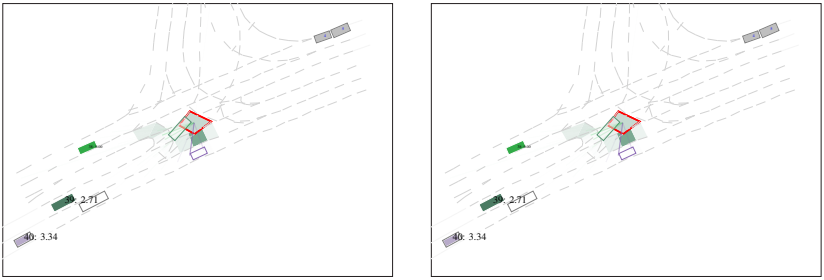


(c) 4.5 s: ...over the whole prediction horizon.

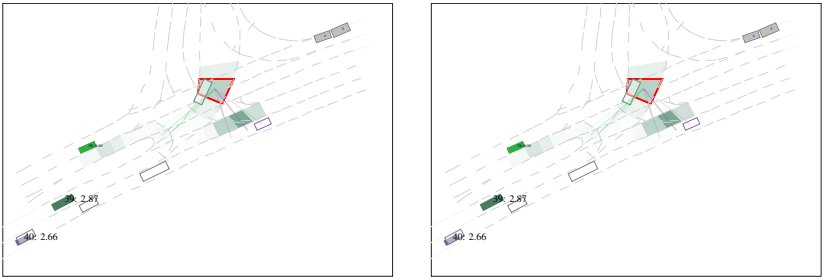
Figure A.2: Conditional prediction, representative examples 5: rounD location2.



(a) 2.1 s: Both prediction for car #40 and for car #39 seam reasonable.



(b) 3.3 s: Note that this scenario is the only one with a U-turn, so the model overestimates the probability for such a sharp curves.



(c) 4.5 s: However, one might have expected at least a bit of prob. mass directly at the stop line.

Figure A.3: Conditional prediction, representative examples 5: rounD location2. A simple example of to agents driving straight on. Since both agents drive independently, the conditional predictions are not that expressive. Also note that the bright green agent on the opposite road has already left the scene, so the conditional prediction for it was most likely normalized over very small number, leading to a bad conditional prediction for it.

References

[1] Rnn-based pedestrian crossing prediction using activity and pose-related features. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020, S. 1801–1806

[2] Alahi, Alexandre ; Goel, Kratarth ; Ramanathan, Vignesh ; Robicquet, Alexandre ; Fei-Fei, Li ; Savarese, Silvio: Social LSTM: Human trajectory prediction in crowded spaces. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, 2016, S. 961–971

[3] Audi AG: *Abmessungen Audi A1 Sportback*. https://www.audi.de/de/brand/de/neuwagen/a1/a1-sportback.html#layer=/de/brand/de/neuwagen/a1/a1-sportback/layer/layer-dimensions.html, Last accessed: 2023-02-27, 2018

[4] Audi AG: *Abmessungen Audi A6 Avant*. https://www.audi.de/de/brand/de/neuwagen/a6/a6-avant.html#layer=/de/brand/de/neuwagen/a6/a6-avant/layer/layer-dimensions.html, Last accessed: 2023-02-27, 2018

[5] Audi AG: *Abmessungen Audi A3 Sportback*. https://www.audi.de/de/brand/de/neuwagen/a3/a3-sportback.html#layer=/de/brand/de/neuwagen/a3/a3-sportback/layer/layer-dimensions.html, Last accessed: 2023-02-27, 2020

[6] Battaglia, Peter ; Pascanu, Razvan ; Lai, Matthew ; Rezende, Danilo J. u. a.: Interaction Networks for Learning about Objects, Relations and Physics. In: *Advances in Neural Information Processing Systems*, 2016, S. 4502–4510

[7] Battaglia, Peter W. ; Hamrick, Jessica B. ; Bapst, Victor ; Sanchez-Gonzalez, Alvaro ; Zambaldi, Vinicius ; Malinowski, Mateusz ; Tacchetti, Andrea ; Raposo, David ; Santoro, Adam ; Faulkner, Ryan u. a.: *Relational inductive biases, deep learning, and graph networks*. https://arxiv.org/abs/1806.01261, Last accessed: 2023-04-10, 2018

[8] Bianchini, M. ; Gori, M. ; Sarti, L. ; Scarselli, F.: Recursive processing of cyclic graphs. In: *Transactions on Neural Networks* 17 (2006), Nr. 1, S. 10–18. http://dx.doi.org/10.1109/TNN.2005.860873. – DOI 10.1109/TNN.2005.860873

[9] Biktairov, Yuriy ; Stebelev, Maxim ; Rudenko, Irina ; Shliazhko, Oleh ; Yangel, Boris: Prank: motion prediction based on ranking. In: *Advances in Neural Information Processing Systems* 33 (2020), S. 2553–2563

[10] Bishop, Christopher M. ; Nasrabadi, Nasser M.: *Pattern recognition and machine learning*. Springer, 2006

[11] Bock, Julian ; Krajewski, Robert ; Moers, Tobias ; Runde, Steffen ; Vater, Lennart ; Eckstein, Lutz: The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In: *Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020, S. 1929–1934

[12] Brandes, Ulrik: Graphentheorie. In: *Handbuch Netzwerkforschung* (2010), S. 345–353

[13] Bruna, Joan ; Zaremba, Wojciech ; Szlam, Arthur ; LeCun, Yann: Spectral networks and deep locally connected networks on graphs. In: *2nd International Conference on Learning Representations (ICLR)*. Banff, Canada, 2014

[14] Buhet, Thibault ; Wirbel, Emilie ; Bursuc, Andrei ; Perrotton, Xavier: PLOP: Probabilistic poLynomial Objects trajectory Planning for autonomous driving. (2020), S. 86–99

[15] Caesar, Holger ; Bankiti, Varun ; Lang, Alex H. ; Vora, Sourabh ; Liong, Venice E. ; Xu, Qiang ; Krishnan, Anush ; Pan, Yu ; Baldan, Giancarlo ; Beijbom, Oscar: nuscenes: A multimodal dataset for autonomous driving. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. Seattle, WA, USA, 2020, S. 11621–11631

[16] Carrasco, S. ; Llorca, D. F. ; Sotelo, M. A.: SCOUT: Socially-COnsistent and UndersTandable Graph Attention Network for Trajectory Prediction of Vehicles and VRUs. In: *IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, 2021, S. 1501–1508

[17] Casas, Sergio ; Gulino, Cole ; Liao, Renjie ; Urtasun, Raquel: Spagnn: Spatially-aware graph neural networks for relational behavior

forecasting from sensor data. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, 2020, S. 9491–9497

[18] Casas, Sergio ; Gulino, Cole ; Suo, Simon ; Luo, Katie ; Liao, Renjie ; Urtasun, Raquel: Implicit latent variable model for scene-consistent motion forecasting. In: *European Conference on Computer Vision (ECCV)*. virtual : Springer, 2020, S. 624–641

[19] Casas, Sergio ; Luo, Wenjie ; Urtasun, Raquel: Intentnet: Learning to predict intention from raw sensor data. In: *Conference on Robot Learning (CoRL)*. Zurich, Switzerland : PMLR, 2018, S. 947–956

[20] Casas, Sergio ; Sadat, Abbas ; Urtasun, Raquel: MP3: A Unified Model To Map, Perceive, Predict and Plan. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA, June 2021, S. 14403–14412

[21] Chai, Yuning ; Sapp, Benjamin ; Bansal, Mayank ; Anguelov, Dragomir: MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In: *Conference on Robot Learning (CoRL)*. Cambridge, MA, USA : PMLR, 2020, S. 86–99

[22] Chang, Ming-Fang ; Lambert, John W. ; Sangkloy, Patsorn ; Singh, Jagjeet ; Bak, Slawomir ; Hartnett, Andrew ; Wang, De ; Carr, Peter ; Lucey, Simon ; Ramanan, Deva ; Hays, James: Argoverse: 3D Tracking and Forecasting with Rich Maps. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. Long Beach, CA, USA, 2019, S. 8748–8757

[23] Chen, Xinye: Understanding Spectral Graph Neural Network. In: *arXiv preprint arXiv:2012.06660* (2020)

[24] Chung, Junyoung ; Gulcehre, Caglar ; Cho, KyungHyun ; Bengio, Yoshua: Empirical evaluation of gated recurrent neural networks on sequence modeling. (2014). `https://arxiv.org/abs/1412.3555`

[25] Clevert, Djork-Arné ; Unterthiner, Thomas ; Hochreiter, Sepp: Fast and accurate deep network learning by exponential linear units (ELUs). In: *Proc. International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico, 2016

[26] Cui, Alexander ; Casas, Sergio ; Sadat, Abbas ; Liao, Renjie ; Urtasun, Raquel: LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving. In: *Proceedings of the IEEE/CVF International Con-*

ference on Computer Vision (ICCV). Montreal, Canada, October 2021, S. 16107–16116

[27] Cui, Henggang ; Nguyen, Thi ; Chou, Fang-Chieh ; Lin, Tsung-Han ; Schneider, Jeff ; Bradley, David ; Djuric, Nemanja: Deep Kinematic Models for Kinematically Feasible Vehicle Trajectory Predictions. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, 2020, S. 10563–10569

[28] Cui, Henggang ; Radosavljevic, Vladan ; Chou, Fang-Chieh ; Lin, Tsung-Han ; Nguyen, Thi ; Huang, Tzu-Kuo ; Schneider, Jeff ; Djuric, Nemanja: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In: *International Conference on Robotics and Automation (ICRA)*. Montreal, Canada, 2019, S. 2090–2096

[29] Dawid, A P.: Present position and potential developments: Some personal views statistical theory the prequential approach. In: *Journal of the Royal Statistical Society: Series A (General)* 147 (1984), Nr. 2, S. 278–290

[30] Defferrard, Michaël ; Bresson, Xavier ; Vandergheynst, Pierre: Convolutional neural networks on graphs with fast localized spectral filtering. In: *Advances in neural information processing systems* 29 (2016), S. 3844–3852

[31] Deo, Nachiket ; Wolff, Eric ; Beijbom, Oscar: Multimodal trajectory prediction conditioned on lane-graph traversals. In: *Conference on Robot Learning (CoRL)*. Auckland, New Zealand : PMLR, 2022, S. 203–212

[32] Formelgrößen, Normenausschuß Technische Grundlagen (NATG) — E.: *DIN 1319-4. Grundlagen der Meßtechnik - Teil 4: Auswertung von Messungen; Meßunsicherheit*. "https://perinorm-s.redi-bw.de/perinorm/fulltext.ashx?fulltextid=8991742b2c7b4c40818ea5fb14e1538e&userid=1a37bae9-63b1-4cef-8afa-a5709d199868", Last accessed: "2022-09-20". https://perinorm-s.redi-bw.de/perinorm/fulltext.ashx?fulltextid=8991742b2c7b4c40818ea5fb14e1538e&userid=1a37bae9-63b1-4cef-8afa-a5709d199868. Version: 1999

[33] Gao, Jiyang ; Sun, Chen ; Zhao, Hang ; Shen, Yi ; Anguelov, Dragomir ; Li, Congcong ; Schmid, Cordelia: Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, 2020, S. 11525–11533

[34] Gilles, Thomas ; Sabatini, Stefano ; Tsishkou, Dzmitry ; Stanciulescu, Bogdan ; Moutarde, Fabien: Home: Heatmap output for future motion estimation. In: *IEEE International Intelligent Transportation Systems Conference (ITSC)*. Indianapolis, IN, USA, 2021, S. 500–507

[35] Gilles, Thomas ; Sabatini, Stefano ; Tsishkou, Dzmitry ; Stanciulescu, Bogdan ; Moutarde, Fabien: Gohome: Graph-oriented heatmap output for future motion estimation. In: *International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA, 2022, S. 9107–9114

[36] Gilles, Thomas ; Sabatini, Stefano ; Tsishkou, Dzmitry ; Stanciulescu, Bogdan ; Moutarde, Fabien: THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling. In: *International Conference on Learning Representations (ICLR)*. virtual, 2022

[37] Gilmer, Justin ; Schoenholz, Samuel S. ; Riley, Patrick F. ; Vinyals, Oriol ; Dahl, George E.: Neural message passing for quantum chemistry. In: *International Conference on Machine Learning (ICML)*. Sydney, Australia : PMLR, 2017, S. 1263–1272

[38] Gneiting, Tilmann ; Katzfuss, Matthias: Probabilistic forecasting. In: *Annual Review of Statistics and Its Application* 1 (2014), S. 125–151

[39] Goller, C. ; Kuchler, A.: Learning task-dependent distributed representations by backpropagation through structure. In: *Proceedings of International Conference on Neural Networks (ICNN)* Bd. 1. Washington, DC, USA, 1996, S. 347–352 vol.1

[40] Gori, Marco ; Maggini, Marco ; Sarti, Lorenzo: A recursive neural network model for processing directed acyclic graphs with labeled edges. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* Bd. 2. Portland, Oregon, USA, 2003, S. 1351–1355

[41] Gori, Marco ; Monfardini, Gabriele ; Scarselli, Franco: A new model for learning in graph domains. In: *IEEE International Joint*

Conference on Neural Networks (IJCNN) Bd. 2. Montreal, QC, Canada, 2005, S. 729–734

[42] Gu, Junru ; Sun, Chen ; Zhao, Hang: DenseTNT: End-to-end trajectory prediction from dense goal sets. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, Canada, 2021, S. 15303–15312

[43] Hastie, Trevor ; Tibshirani, Robert ; Friedman, Jerome H. ; Friedman, Jerome H.: *The elements of statistical learning: data mining, inference, and prediction*. Bd. 2. Springer, 2009

[44] Hochreiter, Sepp: The vanishing gradient problem during learning recurrent neural nets and problem solutions. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (1998), Nr. 02, S. 107–116

[45] Hong, Joey ; Sapp, Benjamin ; Philbin, James: Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, 2019, S. 8454–8462

[46] Hornik, Kurt ; Stinchcombe, Maxwell ; White, Halbert: Multilayer feedforward networks are universal approximators. In: *Neural Networks* 2 (1989), Nr. 5, S. 359–366

[47] Huang, Yanjun ; Du, Jiatong ; Yang, Ziru ; Zhou, Zewei ; Zhang, Lin ; Chen, Hong: A survey on trajectory-prediction methods for autonomous driving. In: *IEEE Transactions on Intelligent Vehicles (IV)* 7 (2022), Nr. 3, S. 652–674

[48] Hug, Ronny ; Becker, Stefan ; Hübner, Wolfgang ; Arens, Michael: On the reliability of LSTM-MDL models for pedestrian trajectory prediction. In: *Representations, Analysis and Recognition of Shape and Motion from Imaging Data: 7th International Workshop (RFMI)*, Springer, 2019, S. 20–34

[49] Kamijo, Ken-ichi ; Tanigawa, Tetsuji: Stock price pattern recognition-a recurrent neural network approach. In: *1990 International Joint Conference on Neural Networks (IJCNN)*. San Diego, CA, USA, 1990, S. 215–221

[50] Khandelwal, Siddhesh ; Qi, William ; Singh, Jagjeet ; Hartnett, Andrew ; Ramanan, Deva: What-If Motion Prediction for Autonomous

Driving. In: *CoRR* abs/2008.10587 (2020). https://arxiv.org/abs/2008.10587

[51] KILIÇARSLAN, Serhat ; KEMAL, ADEM ; ÇELIK, Mete: An overview of the activation functions used in deep learning algorithms. In: *Journal of New Results in Science* 10, Nr. 3, S. 75–88

[52] KIM, ByeoungDo ; PARK, Seong H. ; LEE, Seokhwan ; KHOSHIMJONOV, Elbek ; KUM, Dongsuk ; KIM, Junsoo ; KIM, Jeong S. ; CHOI, Jun W.: LaPred: Lane-Aware Prediction of Multi-Modal Future Trajectories of Dynamic Agents. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA, June 2021, S. 14636–14645

[53] KINGMA, Diederik P. ; BA, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2015)

[54] KIPF, Thomas N. ; WELLING, Max: Semi-supervised classification with graph convolutional networks. In: *5th International Conference on Learning Representations (ICLR)*. Toulon, France, 2017

[55] KRAJEWSKI, Robert ; MOERS, Tobias ; BOCK, Julian ; VATER, Lennart ; ECKSTEIN, Lutz: The round dataset: A drone dataset of road user trajectories at roundabouts in germany. In: *23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece, 2020, S. 1–6

[56] LECUN, Yann ; BENGIO, Yoshua ; HINTON, Geoffrey: Deep learning. In: *nature* 521 (2015), Nr. 7553, S. 436–444

[57] LEDERER, Johannes: *Activation functions in artificial neural networks: A systematic overview*. https://arxiv.org/abs/2101.09957, Last accessed: 2023-04-08, 2021

[58] LEE, Namhoon ; CHOI, Wongun ; VERNAZA, Paul ; CHOY, Christopher B. ; TORR, Philip H. S. ; CHANDRAKER, Manmohan: DESIRE: Distant Future Prediction in Dynamic Scenes With Interacting Agents. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, July 2017

[59] LEFÈVRE, Stéphanie ; VASQUEZ, Dizan ; LAUGIER, Christian: A survey on motion prediction and risk assessment for intelligent vehicles. In: *ROBOMECH journal* 1 (2014), Nr. 1, S. 1–14

[60] Li, Lingyun L. ; Yang, Bin ; Liang, Ming ; Zeng, Wenyuan ; Ren, Mengye ; Segal, Sean ; Urtasun, Raquel: End-to-end Contextual Perception and Prediction with Interaction Transformer. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA, 2020, S. 5784–5791

[61] Liang, Ming ; Yang, Bin ; Hu, Rui ; Chen, Yun ; Liao, Renjie ; Feng, Song ; Urtasun, Raquel: Learning lane graph representations for motion forecasting. In: *16th European Conference on Computer Vision (ECCV)*. Glasgow, United Kingdom : Springer, 2020, S. 541–556

[62] Liang, Ming ; Yang, Bin ; Hu, Rui ; Chen, Yun ; Liao, Renjie ; Feng, Song ; Urtasun, Raquel: Learning lane graph representations for motion forecasting. In: *European Conference on Computer Vision (ECCV)*. Glasgow, United Kingdom : Springer, 2020, S. 541–556

[63] Liang, Ming ; Yang, Bin ; Zeng, Wenyuan ; Chen, Yun ; Hu, Rui ; Casas, Sergio ; Urtasun, Raquel: PnPNet: End-to-End Perception and Prediction With Tracking in the Loop. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Washington D.C., USA, June 2020

[64] Maas, Andrew L. ; Hannun, Awni Y. ; Ng, Andrew Y. u. a.: Rectifier nonlinearities improve neural network acoustic models. In: *Proc. International Conference on Machine Learning (ICML)* Bd. 30, Atlanta, Georgia, USA, 2013, S. 3

[65] MAN: *MAN Lion's City 12*. `https://www.man.eu/ntg_media/media/de/content_medien/doc/bw_master/bus_1/datenblaetter/man_lions_city_12.pdf`, Last accessed: 2023-02-27, 2023

[66] Marchetti, Francesco ; Becattini, Federico ; Seidenari, Lorenzo ; Bimbo, Alberto D.: Mantra: Memory augmented networks for multiple trajectory prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Washington D.C., USA, 2020, S. 7143–7152

[67] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), S. 115–133

[68] Mercat, Jean ; Gilles, Thomas ; El Zoghby, Nicole ; Sandou, Guillaume ; Beauvois, Dominique ; Gil, Guillermo P.: Multi-Head Atten-

tion for Multi-Modal Joint Vehicle Motion Forecasting. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, 2020, S. 9638–9644

[69] Mercat, Jean ; Gilles, Thomas ; El Zoghby, Nicole ; Sandou, Guillaume ; Beauvois, Dominique ; Gil, Guillermo P.: Multi-head attention for multi-modal joint vehicle motion forecasting. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, 2020, S. 9638–9644

[70] Mercedes-Benz Group AG: *smart fortwo, Abmessungen*. https://group-media.mercedes-benz.com/marsMediaSite/de/instance/picture.xhtml?oid=7542896, Last accessed: 2023-02-27, 2014

[71] Mercedes-Benz Group AG: *Die Maße der neuen Mercedes-Benz S-Klasse*. https://group-media.mercedes-benz.com/marsMediaSite/de/instance/picture/Mercedes-Benz-S-Klasse-V-223-2020.xhtml?oid=47190240, Last accessed: 2023-02-27, 2020

[72] Meyer, Annika ; Salscheider, N O. ; Orzechowski, Piotr F. ; Stiller, Christoph: Deep semantic lane segmentation for mapless driving. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, S. 869–875

[73] Meyer, Annika ; Skudlik, Philipp ; Pauls, Jan-Hendrik ; Stiller, Christoph: Yolino: Generic single shot polyline detection in real time. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, Canada, 2021, S. 2916–2925

[74] Meyer, Annika ; Walter, Jonas ; Lauer, Martin: Fast lane-level intersection estimation using markov chain monte carlo sampling and b-spline refinement. In: *Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020, S. 71–76

[75] Micheli, Alessio: Neural Network for Graphs: A Contextual Constructive Approach. In: *Transactions on Neural Networks* 20 (2009), Nr. 3, S. 498–511. http://dx.doi.org/10.1109/TNN.2008.2010350. – DOI 10.1109/TNN.2008.2010350

[76] Milakis, Dimitris ; Van Arem, Bart ; Van Wee, Bert: Policy and society related implications of automated driving: A review of literature and

directions for future research. In: *Journal of Intelligent Transportation Systems* 21 (2017), Nr. 4, S. 324–348

[77] Mozaffari, Sajjad ; Al-Jarrah, Omar Y. ; Dianati, Mehrdad ; Jennings, Paul ; Mouzakitis, Alexandros: Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. In: *IEEE Transactions on Intelligent Transportation Systems (TransITS)* (2020)

[78] Murphy, Allan H. ; Winkler, Robert L.: Reliability of subjective probability forecasts of precipitation and temperature. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 26 (1977), Nr. 1, S. 41–47

[79] Murtagh, Fionn: Multilayer perceptrons for classification and regression. In: *Neurocomputing* 2 (1991), Nr. 5-6, S. 183–197

[80] Ngiam, Jiquan ; Vasudevan, Vijay ; Caine, Benjamin ; Zhang, Zhengdong ; Chiang, Hao-Tien L. ; Ling, Jeffrey ; Roelofs, Rebecca ; Bewley, Alex ; Liu, Chenxi ; Venugopal, Ashish ; Weiss, David J. ; Sapp, Ben ; Chen, Zhifeng ; Shlens, Jonathon: Scene Transformer: A unified architecture for predicting future trajectories of multiple agents. In: *International Conference on Learning Representations (ICLR)*. virtual, 2022

[81] Nyholm, Sven ; Smids, Jilles: Automated cars meet human drivers: responsible human-robot coordination and the ethics of mixed traffic. In: *Ethics and Information Technology* 22 (2020), S. 335–344

[82] Park, Seong H. ; Lee, Gyubok ; Seo, Jimin ; Bhat, Manoj ; Kang, Minseok ; Francis, Jonathan ; Jadhav, Ashwin ; Liang, Paul P. ; Morency, Louis-Philippe: Diverse and admissible trajectory forecasting through multimodal context understanding. In: *European Conference on Computer Vision (ECCV)*. Glasgow, United Kingdom : Springer, 2020, S. 282–298

[83] Phan-Minh, Tung ; Grigore, Elena C. ; Boulton, Freddy A. ; Beijbom, Oscar ; Wolff, Eric M.: Covernet: Multimodal behavior prediction using trajectory sets. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Washington D.C., USA, 2020, S. 14074–14083

[84] Phillips, John ; Martinez, Julieta ; Barsan, Ioan A. ; Casas, Sergio ; Sadat, Abbas ; Urtasun, Raquel: Deep Multi-Task Learning for

Joint Localization, Perception, and Prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA, June 2021, S. 4679–4689

[85] Poggenhans, Fabian ; Pauls, Jan-Hendrik ; Janosovits, Johannes ; Orf, Stefan ; Naumann, Maximilian ; Kuhnt, Florian ; Mayr, Matthias: Lanelet2: A high-definition map framework for the future of automated driving. In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, 2018, S. 1672–1679

[86] Qi, Charles R. ; Yi, Li ; Su, Hao ; Guibas, Leonidas J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: *Advances in neural information processing systems* 30 (2017)

[87] Rehder, Eike ; Wirth, Florian ; Lauer, Martin ; Stiller, Christoph: Pedestrian prediction by planning using deep neural networks. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, 2018, S. 5903–5908

[88] Rhinehart, Nicholas ; Kitani, Kris M. ; Vernaza, Paul: R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Munich, Germany, 2018, S. 772–788

[89] Rhinehart, Nicholas ; McAllister, Rowan ; Kitani, Kris ; Levine, Sergey: PRECOG: PREdiction Conditioned on Goals in Visual Multi-Agent Settings. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea, October 2019

[90] Rosenblatt, Frank: The perceptron: a probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), Nr. 6, S. 386

[91] Rudenko, Andrey ; Palmieri, Luigi ; Herman, Michael ; Kitani, Kris M. ; Gavrila, Dariu M. ; Arras, Kai O.: Human motion trajectory prediction: A survey. In: *The International Journal of Robotics Research* 39 (2020), Nr. 8, S. 895–935

[92] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by back-propagating errors. In: *nature* 323 (1986), Nr. 6088, S. 533–536

[93] Sadat, Abbas ; Casas, Sergio ; Ren, Mengye ; Wu, Xinyu ; Dhawan, Pranaab ; Urtasun, Raquel: Perceive, predict, and plan: Safe motion

planning through interpretable semantic representations. In: *European Conference on Computer Vision (ECCV)*. virtual : Springer, 2020, S. 414–430

[94] Salzmann, Tim ; Ivanovic, Boris ; Chakravarty, Punarjay ; Pavone, Marco: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: *European Conference on Computer Vision (ECCV)*. Glasgow, United Kingdom : Springer, 2020, S. 683–700

[95] Samek, Wojciech ; Montavon, Grégoire ; Vedaldi, Andrea ; Hansen, Lars K. ; Müller, Klaus-Robert: *Explainable AI: interpreting, explaining and visualizing deep learning*. Bd. 11700. Springer Nature, 2019

[96] Scarselli, Franco ; Gori, Marco ; Tsoi, Ah C. ; Hagenbuchner, Markus ; Monfardini, Gabriele: The graph neural network model. In: *IEEE Transactions on Neural Networks* 20 (2008), Nr. 1, S. 61–80

[97] Sener, Ozan ; Koltun, Vladlen: Multi-task learning as multi-objective optimization. In: *Advances in neural information processing systems* 31 (2018)

[98] Servan-Schreiber, David ; Cleeremans, Axel ; McClelland, James L.: Encoding sequential structure in simple recurrent networks / Carnegie-Mellon University Pittsburgh Dept. of Psychology. Pittburgh, PA, USA, 1989. – Forschungsbericht

[99] Shalev-Shwartz, Shai ; Shammah, Shaked ; Shashua, Amnon: *On a formal model of safe and scalable self-driving cars*. `https://arxiv.org/pdf/1708.06374.pdf`, Last accessed: 2023-04-08, 2017

[100] Sivaraman, Sayanan ; Trivedi, Mohan M.: Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. In: *IEEE Transactions on Intelligent Transportation Systems (TransITS)* 14 (2013), Nr. 4, S. 1773–1795

[101] Song, Haoran ; Luan, Di ; Ding, Wenchao ; Wang, Michael Y. ; Chen, Qifeng: Learning to Predict Vehicle Trajectories with Model-based Planning. In: Faust, Aleksandra (Hrsg.) ; Hsu, David (Hrsg.) ; Neumann, Gerhard (Hrsg.): *Proceedings of the 5th Conference on Robot Learning (CoRL)* Bd. 164. virtual : PMLR, 08–11 Nov 2022 (Proceedings of Machine Learning Research), 1035–1045

[102] Sperduti, A. ; Starita, A.: Supervised neural networks for the classification of structures. In: *Transactions on Neural Networks* 8 (1997), Nr.

3, S. 714–735. http://dx.doi.org/10.1109/72.572108. – DOI 10.1109/72.572108

[103] Tang, Charlie ; Salakhutdinov, Russ R.: Multiple futures prediction. In: *Advances in Neural Information Processing Systems* 32 (2019)

[104] Tittmann, Peter: Graphentheorie. In: *Hanser Fachbuchverlag* 2 (2003)

[105] Tolstaya, Ekaterina ; Mahjourian, Reza ; Downey, Carlton ; Vadarajan, Balakrishnan ; Sapp, Benjamin ; Anguelov, Dragomir: Identifying Driver Interactions via Conditional Behavior Prediction. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China, 2021, S. 3473–3479

[106] Tutte, William T.: *Graph theory*. Bd. 21. Cambridge university press, 2001

[107] Varadarajan, Balakrishnan ; Hefny, Ahmed ; Srivastava, Avikalp ; Refaat, Khaled S. ; Nayakanti, Nigamaa ; Cornman, Andre ; Chen, Kan ; Douillard, Bertrand ; Lam, Chi P. ; Anguelov, Dragomir u. a.: Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: *International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA, 2022, S. 7814–7821

[108] VW Nutzfahrzeuge: *Der Multivan: Technische Daten*. https://www.volkswagen-nutzfahrzeuge.de/de/modelle/multivan/technische-daten.html, Last accessed: 2023-02-27, 2021

[109] Wang, Tsun-Hsuan ; Manivasagam, Sivabalan ; Liang, Ming ; Yang, Bin ; Zeng, Wenyuan ; Urtasun, Raquel: V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In: *European Conference on Computer Vision (ECCV)*. virtual : Springer, 2020, S. 605–621

[110] Werbos, Paul J.: *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. Bd. 1. John Wiley & Sons, 1994

[111] Wirth, Florian ; Krane, Stephan ; Loos, Melanie ; Rehder, Eike ; Fernandez, Carlos: What Does a Good Prediction Look Like? In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, 2019, S. 1594–1599

[112] Wirth, Florian ; Wadephul, Julian ; Scheid, Alexander ; Fernandez-Lopez, Carlos ; Stiller, Christoph: Model-based State Estimation

of Two-Wheelers. In: *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA, 2022, S. 5382–5388

[113] Wirth, Florian ; Wen, Tao ; Fernandez-Lopez, Carlos ; Stiller, Christoph: Model-based prediction of two-wheelers. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020, S. 1669–1674

[114] Ye, Maosheng ; Cao, Tongyi ; Chen, Qifeng: Tpcn: Temporal point cloud networks for motion forecasting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA, 2021, S. 11318–11327

[115] Yuan, Ye ; Weng, Xinshuo ; Ou, Yanglan ; Kitani, Kris M.: AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, Canada, October 2021, S. 9813–9823

[116] Zeng, Wenyuan ; Liang, Ming ; Liao, Renjie ; Urtasun, Raquel: LaneRCNN: Distributed Representations for Graph-Centric Motion Forecasting. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), S. 532–539. http://dx.doi.org/10.1109/IROS51168.2021.9636035. – DOI 10.1109/IROS51168.2021.9636035

[117] Zeng, Wenyuan ; Wang, Shenlong ; Liao, Renjie ; Chen, Yun ; Yang, Bin ; Urtasun, Raquel: DSDNet: Deep Structured Self-driving Network. In: *16th European Conference on Computer Vision (ECCV)*. Glasgow, United Kingdom, 2020, S. 156–172

[118] Zernetsch, Stefan ; Reichert, Hannes ; Kress, Viktor ; Doll, Konrad ; Sick, Bernhard: Trajectory forecasts with uncertainties of vulnerable road users by means of neural networks. In: *Intelligent Vehicles Symposium (IV)*. Paris, France, 2019, S. 810–815

[119] Zernetsch, Stefan ; Reichert, Hannes ; Kress, Viktor ; Doll, Konrad ; Sick, Bernhard: A Holistic View on Probabilistic Trajectory Forecasting–Case Study. Cyclist Intention Detection. In: *Intelligent Vehicles Symposium (IV)*. Aachen, Germany, 2022, S. 265–272

[120] Zhan, Wei ; Sun, Liting ; Wang, Di ; Shi, Haojie ; Clausse, Aubrey ; Naumann, Maximilian ; Kümmerle, Julius ; Königshof, Hendrik ; Stiller, Christoph ; La Fortelle, Arnaud de ; Tomizuka,

Masayoshi: INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. In: *arXiv:1910.03088 [cs, eess]* (2019). https://arxiv.org/abs/1910.03088. – Accessed on 10.03.2023

[121] Zhang, Yu ; Yang, Qiang: An overview of multi-task learning. In: *National Science Review* 5 (2018), Nr. 1, S. 30–43

[122] Zhang, Yu ; Yang, Qiang: A survey on multi-task learning. In: *Transactions on Knowledge and Data Engineering* 34 (2021), Nr. 12, S. 5586–5609

[123] Zhao, Hang ; Gao, Jiyang ; Lan, Tian ; Sun, Chen ; Sapp, Ben ; Varadarajan, Balakrishnan ; Shen, Yue ; Shen, Yi ; Chai, Yuning ; Schmid, Cordelia u. a.: Tnt: Target-driven trajectory prediction. In: *Conference on Robot Learning (CoRL)*. London, United Kingdom : PMLR, 2021, S. 895–904

[124] Zhao, Hang ; Gao, Jiyang ; Lan, Tian ; Sun, Chen ; Sapp, Ben ; Varadarajan, Balakrishnan ; Shen, Yue ; Shen, Yi ; Chai, Yuning ; Schmid, Cordelia u. a.: Tnt: Target-driven trajectory prediction. In: *Conference on Robot Learning (CoRL)*. London, United Kingdom : PMLR, 2021, S. 895–904