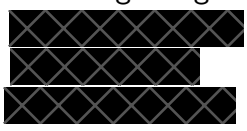
Grid-Based Stochastic Model Predictive Control with Occupancy Grids

Scientific thesis for the procurance of the degree B.Sc.
from the Department of Electrical and Computer Engineering at the Technical University of Munich.

Supervised by Univ.-Prof. Dr.-Ing./Univ. Tokio habil. Martin Buss
M.Sc. Tim Brüdigam
Chair of Automatic Control Engineering

Submitted by cand. ing. Diego Martí Monsó


Submitted on Munich, 13.10.2021

TECHNISCHE UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR STEUERUNGS- UND REGELUNGSTECHNIK
UNIV.-PROF. DR.-ING./UNIV. TOKIO MARTIN BUSS
DR.-ING. MARION LEIBOLD

May 25, 2021

B A C H E L O R T H E S I S
for
Diego Martí Monsó
Student ID 03716066, Degree EI

Grid-Based Stochastic Model Predictive Control with Occupancy Grids

Problem description:

For autonomous vehicles it is essential to model and estimate the environment. Occupancy grids are a popular concept for a generic environment representation with a grid cell discretization and occupancy probabilities of the individual cells. Recent approaches extend this strategy toward dynamic environments, e.g., [4], which can form the basis for a robust grid-based multi-sensor object tracking [3]. Such a dynamic occupancy grid representation can also be used directly for trajectory planning by adding a grid prediction, similar to [2]. A grid-based stochastic model predictive control (SMPC) method has been proposed recently in [1], where environment uncertainty is handled by SMPC chance constraints. However, this proposed grid-based SMPC method only assumes and simulates simple occupancy grid data and does not yet fully exploit the information provided by occupancy grids.
The aim of this work is to improve the grid-based SMPC approach such that the predictions exploit occupancy grid information. The results will be evaluated in the SUMO traffic simulator.

Tasks:

- Literature research on SMPC and occupancy grids
- Exploitation of occupancy grid data to improve grid-based SMPC
- Evaluation of simulation results with the SUMO simulator

Bibliography:

[1] T. Brüdigam, F. di Luzio, L. Pallottino, D. Wollherr, and M. Leibold. Grid-based stochastic model predictive control for trajectory planning in uncertain environments. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.
[2] M. Schreiber, S. Hoermann, and K. Dietmayer. Long-term occupancy grid prediction using recurrent neural networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9299–9305, 2019.
[3] S. Steyer, C. Lenk, D. Kellner, G. Tanzmeister, and D. Wollherr. Grid-based object tracking with nonlinear dynamic state and shape estimation. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–20, 2019.
[4] S. Steyer, G. Tanzmeister, and D. Wollherr. Grid-Based Environment Estimation Using Evidential Mapping and Particle Tracking. *IEEE Trans. Intell. Veh.*, 3(3):384–396, September 2018.

Supervisor:	M. Sc. Tim Brüdigam
Start:	XX.XX.2021
Intermediate Report:	XX.XX.2021
Delivery:	XX.XX.2021

(M. Leibold)
Akad. Rat

Abstract

Stochastic Model Predictive Control is a trajectory planning strategy for autonomous vehicles that is able to handle probabilistic uncertainties with chance constraints. The planning algorithm optimizes the path of an autonomous vehicle based on forecasts about the future development of the surrounding environment. Nevertheless, the optimal control problem has a high computational cost, which can be reduced through the introduction of grid-based environment representations that build on Occupancy Grids. For the predictions, assumptions about other traffic participants are founded on hand-engineered heuristics and probabilistic models. However, due to the multimodal nature of driving scenarios and the probabilistic dependencies of traffic events, conventional prediction models are generally not able to scale and generalize to complex and diverse scenarios. Conversely, Recurrent Neural Networks have the capability to learn to model probability estimation based on training data, such as sequences of Occupancy Grids. Thus, we develop two different deep learning long-term prediction models and present a procedure to integrate the data-driven forecasts into the current planning method. Lastly, the approach is demonstrated in a simulation framework in an autonomous driving scenario.

Contents

Chapter 1

Introduction

In recent years, research on autonomous driving has intensified, with increasing support from both academia and industry. Some of the most important tasks of an autonomous agent encompass environment estimation, as well as long-term and efficient trajectory planning in uncertain traffic scenarios. Popular approaches to address these challenges include the Occupancy Grid (OG), Model Predictive Control (MPC), and diverse Machine Learning techniques, such as deep Recurrent Neural Networks (RNNs).

The OG [STW17], together with other grid-map environment representations that build on top of it, enables a robust and consistent modeling of the surroundings of an intelligent agent. An OG maps the territory around the controlled entity into a grid where each cell is assigned an occupancy probability that estimates the likelihood of the presence of an obstacle. Improved versions of the OG, like the Dynamic Occupancy Grid (DOG), include dynamic information of all the objects within the detection range and can be used for dynamic object tracking. The use of grid environment maps is widespread in robotics and is gaining in popularity in the automotive sector.

Furthermore, MPC is a control method that has also seen a gradual, hardware-enabled rise in adoption. MPC repeatedly solves an open-loop optimal control problem at every sampling instant [GP17, Mes16]. With the help of a dynamic system model, a cost function with state and input constraints is minimized over a sequence of control inputs that extends from the current time step to the control horizon. Even though only the first control action of the sequence is applied, making predictions about future time steps allows to anticipate events and avoid overshoot. As the control process advances, updated measurements – acting as a feedback to the MPC – are considered and the horizon recedes. To better deal with probabilistic uncertainties, inherent to real-world scenarios, Stochastic Model Predictive Control (SMPC) introduces chance constraints that allow to account for uncertainties in the control design. The SMPC method is also an effective approach for trajectory planning in automated driving and its efficiency can be further improved with grid-based SMPC implementations with OGs [BDP+20].

Lastly, different Recurrent Neural Network (RNN) architectures [She20], which are founded on fully connected Long Short-Term Memories (LSTMs), have proven an outstanding performance in sequence-to-sequence mapping tasks in fields like natural language processing [Neu17] or prediction of future video frames [SHD19]. The latter example is of special interest, as OGs have a similar structure to video frames: an OG can be interpreted as a grayscale image, where the brightness intensity of each pixel is directly obtained from the occupancy value at the corresponding cell in the OG. On top of that, the pixel map contains shapes and patterns that can be recognized by a learning algorithm. Moreover, a series of OGs produces an animation of the driving scenario. Thus, RNN-based deep learning approaches can also be applied to forecast the future development of grid environments by inference from an OG sequence of previous sampling steps, as in [PKK+18, SHD19, MR19]. In this work, a similar procedure is integrated into the current grid-based SMPC method to analyze motion patterns of surrounding obstacles and anticipate their maneuvers for safe trajectory planning.

Problem Statement

To reduce the computational complexity derived from chance constraints in SMPC, a grid-based technique has been proposed in [BDP+20]. The same approach is refined in [Hö20] to integrate the dynamic object tracking algorithm from [STW17]. However, the complex grid environment representations that are used contain and entail valuable information that is not fully taken advantage of. Besides, predictions about future states of surrounding traffic participants are only founded on rather simple, manually-designed heuristics that are not able to generalize to diverse traffic scenarios.

Therefore, the objective of this work is to better exploit available OG and dynamic object data by adding an online long-term grid prediction to the current grid-based SMPC method. We develop the data-driven grid prediction with deep learning techniques, first as a classification LSTM network, and second as a deep regression LSTM network. Both approaches are able to forecast the future motion of various dynamic objects within the detection range for multiple steps with a long-term prediction horizon. In addition, we present a methodology to integrate the prediction models into the grid-based SMPC trajectory planning algorithm.

Finally, a framework is designed to simulate different driving scenarios and evaluate the method with the proposed modification. Unlike the simulations performed in some of the previous work, which are carried out with prerecorded data, the simulation framework is designed in such a way that the actions of the controlled vehicle affect the local environment and elicit reactions from other agents.

Chapter 2

Preliminaries

This chapter opens with the literature review in Section 2.1, with focus on related work about OGs, different SMPC approaches, RNN-based deep learning architectures, and microscopic traffic simulation frameworks. Subsequently, Sections 2.2 - 2.4 briefly present each of the methods and the corresponding notation used in this work.

2.1 Related Work

Due to its extraordinary ability to cope with probabilistic uncertainties in real-world systems, SMPC has become a subject of study for automotive applications, such as vehicle path planning. In [Mes16], multiple SMPC approaches are presented and the need for efficient uncertainty propagation algorithms is addressed. To propagate uncertainties to future predictions, some SMPC implementations like [BDP+20, Hö20] rely on the method introduced in [CGLB14]. Furthermore, a strategy to achieve a numerically tractable formulation of chance constraints is proposed in [BDP+20]. The authors take advantage of the OG to find a convex set of admissible space, which can be expressed as a standard MPC constraint, alleviating the computational complexity of the online optimal control problem. In [STW17], density-based clustering and particle filters are used to obtain object tracks of dynamic traffic participants over time based on DOGs. The object tracking approach is integrated in [Hö20] into the grid-based SMPC from [BDP+20]. Albeit the substantial leap in efficiency from grid-based SMPC in comparison to standard SMPC for automated driving, the grid-based SMPC disregards much of the available data derived during the tracking of dynamic traffic participants and employs a prediction model that is hardly scalable. Thus, the method has a great potential for improvement and the waste data can be effectively used to upgrade the prediction approach.
Data-driven predictions of the future motion of traffic participants based on grid environment representations have been studied previously. The state of the art in the literature for predicting the evolution of road environments from sequences of OGs addresses the problem with deep learning models, which show a high degree of efficiency and better results than traditional approaches. For instance, a difference

learning RNN model is presented in [MR19] that shows a high degree of accuracy. In addition, the convolutional network to forecast future OGs in urban driving scenarios propounded in [HBD17] is outperformed by the authors with an RNN network in [SHD19], which takes DOGs sequentially as inputs and produces separate static and dynamic predictions. The probabilistic RNN model to predict future vehicle positions introduced in [PKK+18] is close to our implementation of the classification prediction network. In contrast to the previously discussed works, the data used in the model of [PKK+18] is not in the format of OGs, but as state vectors that describe the motion of individual vehicles. Although the network is provided with little contextual information about other traffic participants for the inference, the model still demonstrates a high performance.
Finally, a general technique to couple the microscopic traffic simulator SUMO (Simulation of Urban MObility) [LBBW+18] with another vehicle simulation environment is described in [KDEA19]. The sub-lane model that comes natively in SUMO, which allows to simulate lateral vehicle movement precisely, already integrates the functionality from the algorithm proposed in [KDEA19] to imitate smooth lane changes. Moreover, the procedure to warn external vehicles about maneuvers in advance could result in overly favorable scene development that does not take into account real-life driving errors of other traffic participants.

2.2 Occupancy Grid

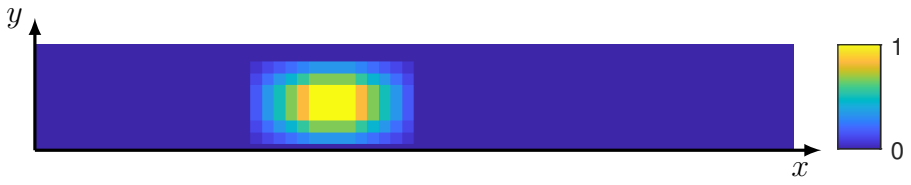


Figure 2.1: Qualitative example of an OG representing a road scenario with one detected object. The color coding indicates the estimated occupancy probability.

An OG is a two-dimensional, space-discrete representation of the local environment that provides a rectangular shaped bird's-eye view of the setting with length ι_x and width ι_y. The surrounding of the EV is divided into a grid $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{\frac{\iota_x}{a_x} \times \frac{\iota_y}{a_y}}$ of cells $c_{i,j} \in \boldsymbol{\mathcal{G}}$ of length a_x and width a_y, where the subscripts x and y denote the two spatial dimensions, and i and j are indices. Additionally, the fractions $\frac{\iota_x}{a_x}$ and $\frac{\iota_y}{a_y}$ are required to be integers. The size of the cells, which can be seen as the resolution of the grid, poses a trade-off between accuracy and computational cost. Each of the cells is assigned an occupancy value that approximates the posterior probability of the presence of an object. Different algorithmic approaches exist to estimate the occupancy values, typically as Bayesian occupancy estimations of inexact fused

sensor measurements. In addition, variants of the standard OG may include diverse channels to code supplementary information, such as dynamic estimates.

2.3 Stochastic Model Predictive Control

In contrast to standard MPC, SMPC [Mes16] accounts for a probabilistic uncertainty $\boldsymbol{w}$ in the system model. As the dynamic range of the uncertainty is typically unrestricted, fulfillment of hard state constraints cannot always be guaranteed and chance constraints are used instead, which only require a certain degree of constraint satisfaction $\beta \in [0, 1]$. Hence, SMPC allows to dynamically compensate between achieving the control objectives and satisfying probabilistic constraints. The higher the probability level β, the more conservative the resulting control behavior. Similarly to MPC, an open-loop, constrained optimal control problem is to be solved online. The cost function J is minimized over the control horizon $N \in \mathbb{N}$ with regards to the input sequence $\boldsymbol{U} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}\}$ of control actions $\boldsymbol{u}_h$ at prediction step h. The general SMPC optimal control problem is therefore given by

$$\boldsymbol{U}^* = \arg\min_{\boldsymbol{U}} J\left(\boldsymbol{\xi}_h, \boldsymbol{u}_h\right) \tag{2.1a}$$

$$\text{s.t.} \quad \boldsymbol{\xi}_0 = \hat{\boldsymbol{\xi}}_k, \tag{2.1b}$$

$$\boldsymbol{\xi}_{h+1} = f(\boldsymbol{\xi}_h, \boldsymbol{u}_h, \boldsymbol{w}_h), \quad h \in \mathbb{N}, \tag{2.1c}$$

$$\boldsymbol{u}_h \in \mathcal{U}_h, \qquad h = 0, \ldots, N-1, \tag{2.1d}$$

$$\Pr(\boldsymbol{\xi}_h \in \Xi_h) \geq \beta, \qquad h = 1, \ldots, N, \tag{2.1e}$$

with the state vector $\boldsymbol{\xi}_h$ at prediction step h, the state estimation $\hat{\boldsymbol{\xi}}_k$ at measurement time stamp k, and the constraint sets $\mathcal{U}_h$ and Ξ_h. Only the first control input $\boldsymbol{u}_0^*$ is applied to the system from the optimal control sequence $\boldsymbol{U}^* = \left\{\boldsymbol{u}_0^*, \ldots, \boldsymbol{u}_{N-1}^*\right\}$, which is the result of the SMPC optimal control problem. In the following sampling instant, the state vector $\boldsymbol{\xi}_0$ is updated.
Overall, SMPC combines capability to deal with complex multiple-input, multiple-output systems of MPC with a stochastic characterization of uncertainties. However, propagating uncertainties through the prediction horizon is computationally demanding, especially given complex or nonlinear system dynamics. Besides, the chance constraint (2.1e) is generally non-convex and intractable. Thus, OGs can be integrated in a grid-based SMPC approach [BDP+20] to derive a deterministic approximation of the chance constraints, which is further explored in the second half of this work.

2.4 Long Short-Term Memory

The Recurrent Neural Network (RNN) is an artificial neural network that is able to learn to process sequential data, such as time series data [She20]. At every time

step k, the hidden state $\boldsymbol{h}_k$ of the standard RNN cell is fed back to its input in the following time step $k+1$. In this way, information is carried on sequentially through time and the output of the cell is influenced by the previous events. However, the RNN suffers from the vanishing gradient problem during training, which prevents the network from learning long-term dependencies in the input sequence.

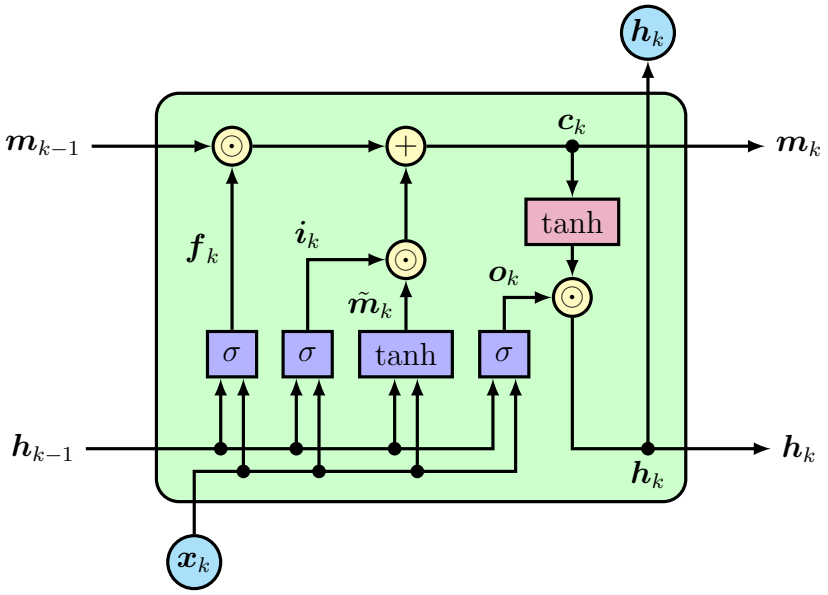


Figure 2.2: Internal structure of a standard LSTM cell. Operators are colored yellow, internal functions purple, activation functions blue, and external inputs and outputs cyan. Weight matrices and bias vectors have been left out.

To solve this issue, the Long Short-Term Memory (LSTM) complements the standard RNN with an additional cell memory $\boldsymbol{m}_k$ and a gating mechanism that controls the information flow within the cell according to the following recursive equations:

$$\boldsymbol{f}_k = \sigma\left(\boldsymbol{W}_{fh}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{fx}\boldsymbol{x}_k + \boldsymbol{b}_f\right), \tag{2.2a}$$
$$\boldsymbol{i}_k = \sigma\left(\boldsymbol{W}_{ih}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{ix}\boldsymbol{x}_k + \boldsymbol{b}_i\right), \tag{2.2b}$$
$$\boldsymbol{o}_k = \sigma\left(\boldsymbol{W}_{oh}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{ox}\boldsymbol{x}_k + \boldsymbol{b}_o\right), \tag{2.2c}$$
$$\tilde{\boldsymbol{m}}_k = \tanh\left(\boldsymbol{W}_{mh}\boldsymbol{h}_{k-1} + \boldsymbol{W}_{mx}\boldsymbol{x}_k + \boldsymbol{b}_c\right), \tag{2.2d}$$
$$\boldsymbol{m}_k = \boldsymbol{f}_k \odot \boldsymbol{m}_{k-1} + \boldsymbol{i}_k \odot \tilde{\boldsymbol{m}}_k, \tag{2.2e}$$
$$\boldsymbol{h}_k = \boldsymbol{o}_k \odot \tanh\left(\boldsymbol{m}_k\right), \tag{2.2f}$$

where the operator $\odot$ denotes the Hadamard product (i.e., the element-wise multiplication of vectors), the hyperbolic tangent $\tanh : \mathbb{R}^\eta \to (-1,1)^\eta$ is applied element-wise, and the element-wise sigmoid logistic function $\sigma_{\text{sigmoid}} : \mathbb{R}^\eta \to (0,1)^\eta$ is used as an activation function:

$$\sigma_{\text{sigmoid}}(\boldsymbol{v})_j = \frac{1}{1+e^{-v_j}}, \tag{2.3}$$

for $j = 1, \ldots, \eta$ and $\boldsymbol{v} = [v_1, \ldots, v_\eta]^\top \in \mathbb{R}^\eta$. Furthermore, the input $\boldsymbol{x}_k \in \mathbb{R}^\delta$ at time step k is a vector of δ input features, the bias vectors $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{b}_c \in \mathbb{R}^\eta$ have length η, and $\boldsymbol{W}_{fh}, \boldsymbol{W}_{ih}, \boldsymbol{W}_{oh}, \boldsymbol{W}_{mh} \in \mathbb{R}^{\eta \times \eta}$ and $\boldsymbol{W}_{fx}, \boldsymbol{W}_{ix}, \boldsymbol{W}_{ox}, \boldsymbol{W}_{mx} \in \mathbb{R}^{\eta \times \delta}$ are weight matrices. The cell memory $\boldsymbol{m}_k \in \mathbb{R}^\eta$ at time step k is updated in (2.2e) by applying the forget gate $\boldsymbol{f}_k \in (0,1)^\eta$ from (2.2a) to the previous cell memory $\boldsymbol{m}_{k-1}$, and the input gate $\boldsymbol{i}_k \in (0,1)^\eta$ from (2.2b) to the new cell memory candidate $\tilde{\boldsymbol{m}}_k \in (-1,1)^\eta$ from (2.2d). The forget gate decides about the information that will no longer be tracked in the cell memory, while the input gate chooses and scales the elements of the new cell memory candidate that will be taken over. Next, the hidden state $\boldsymbol{h}_k \in (-1,1)^\eta$ at time step k is obtained in (2.2f) by employing the output gate $\boldsymbol{o}_k \in (0,1)^\eta$ from (2.2c) on the updated cell memory, with its range limited to $(-1,1)^\eta$ by the element-wise hyperbolic tangent. The flow of information within the LSTM cell is represented in Figure 2.2.

Overall, the LSTM cell has an increased memory compared to the standard RNN cell. In consequence, LSTMs are often used as the core block in long-term deep learning prediction models, which operate in a sequence-to-sequence fashion.

Chapter 3

Multi-Step Grid Predictor

The ability to foresee the future development of the local road environment is essential in autonomous driving, not only because anticipating events helps to avoid accidents, but also to be able to achieve more comfortable and efficient driving. Thus, an autonomous vehicle has to understand the scene context and analyze behavioral and motion patterns to draw hypotheses about the future. However, traffic scenarios typically involve a high degree of uncertainty, due to the deep probabilistic dependencies of traffic events, an almost unlimited amount of possible outcomes, and the irrationality introduced by humans. In the past, handcrafted probabilistic and heuristic models have been developed to forecast the future motion of traffic participants, but such models have been outperformed by novel deep learning approaches, as in [HBD17, PKK+18, SHD19, MR19]. In this work, we consider an Ego Vehicle (EV), i.e., the controlled agent, that interacts with external traffic participants, which are the Target Vehicles (TVs).
In this chapter, we explore two different RNN-based models for multi-step and long-term prediction of dynamic objects in driving scenarios and propose a methodology to derive probabilistic information that can later be useful in the grid-based SMPC scheme. First, the general and shared predictor structure between the two data-driven prediction models is introduced in Section 3.1. Next, some underlying concerns about the training data are debated in Section 3.2. Lastly, the two deep neural network based grid prediction models are presented as a multi-class classification in Section 3.3 and a multi-output regression in Section 3.4.

3.1 General Multi-Step Prediction Model Structure

This section outlines the shared general structure that is followed in the classification as well as in the regression model to obtain multiple steps of prediction outputs.
The multi-step prediction of multivariate time series data with a long-term prediction horizon $N \in \mathbb{N}$ is a sequence-to-sequence forecasting task that can generally be

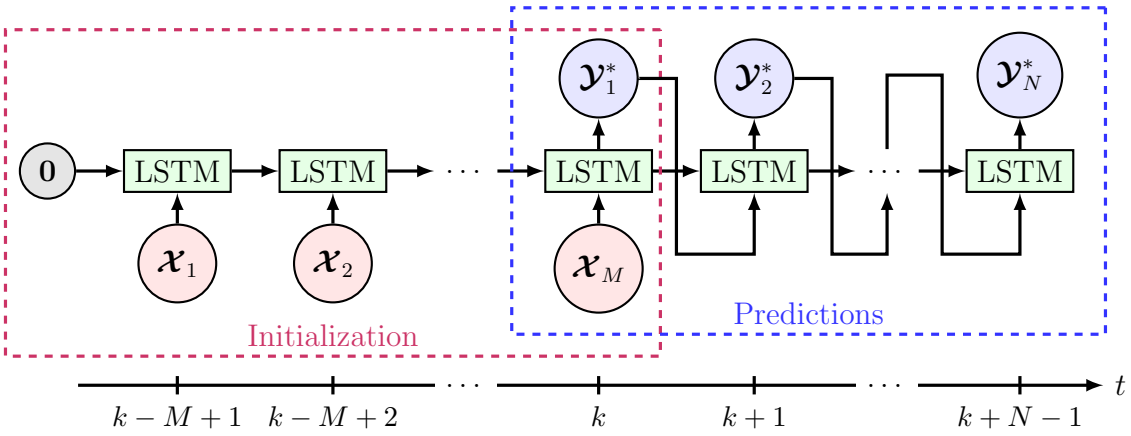


Figure 3.1: Unrolled structure of a general LSTM network used for multi-step prediction. First, the evidence sequence $\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\}$ of length M is input sequentially to the previously unitialized network, as shown by the initial zero-vector. In the next phase, N predictions $\{\boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_N^*\}$ are generated in a recursive way. The LSTM internal states $\boldsymbol{m}_k$ and $\boldsymbol{h}_k$ are propagated and updated through time, indicated by the horizontal arrows. Below, the axis of the time t with the current sampling step k can be found.

regarded as an optimization process: given a sequence of $M \in \mathbb{N}$ previous observations at time step k, the objective is to minimize the prediction error of the forecasts for prediction steps $h = 1, \ldots, N$, where every prediction instant h corresponds to the time stamp $k + h$.

In this work, we treat the data-driven forecast of the future evolution of dynamic road scenarios first as a classification problem, which estimates probabilistic features about the future TV motion, and later as a regression task, where the model output is a deterministic prediction of the road environment. For both cases, the multi-step prediction framework is constructed and trained following the same principle to produce single-step forecasts recursively as a multi-step process, depicted in Figure 3.1. The scheme is divided into two parts: an initialization phase, where the observations are input to the network, and a multi-step prediction stage, which repeats the inference step considering past model outputs as new evidence.

In the initialization phase or LSTM encoding process, a series of M observations $\boldsymbol{\mathcal{X}}_m$, with $m = 1, \ldots, M$, is given to the prediction model sequentially. The observation sequence ranges from sampling step $k - M + 1$ to the current sampling stamp k. In the two prediction approaches examined in this work, the deep neural networks have an LSTM block at the core, which is previously uninitialized and performs the fundamental function of encoding the evidence sequence in the internal state vectors $\boldsymbol{m}_k$ and $\boldsymbol{h}_k$, which are described in (2.2e) and (2.2f) respectively, to summarize the observation sequence in a low dimensional latent space representation. For that purpose, the LSTM units need to find spatio-temporal correlations in the input

sequence. In addition, further layers are found before the recurrent cells to support the LSTM in capturing spatial patterns and abstract dependencies in the observations and encode them into feature vectors. In the meantime, the produced system outputs during the initialization phase until time instance k are discarded, as predictions about past time steps are of little interest.
After the encoding phase, in the inference stage, the LSTM memory vectors $\boldsymbol{m}_k$ and $\boldsymbol{h}_k$, together with the last input of the observation sequence, are used for a first single-step forecast. The single-step inference is then repeated recursively until the prediction horizon N to generate the output sequence $\{\boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_N^*\}$ of predictions $\boldsymbol{\mathcal{Y}}_h^*$. Therefore, the output $\boldsymbol{\mathcal{Y}}_{h-1}^*$ at prediction step $h-1$ is fed back to the network input at step h and treated as a new observation $\boldsymbol{\mathcal{X}}_h$:

$$\boldsymbol{\mathcal{X}}_h = \boldsymbol{\mathcal{Y}}_{h-1}^*. \tag{3.1}$$

Besides, the cell states $\boldsymbol{m}_{h-1}$ and $\boldsymbol{h}_{h-1}$ are further propagated and updated through time. The described methodology is a common alternative to the LSTM Encoder-Decoder structure used in [PKK+18, SHD19] and differs mostly in the decoding stage after the LSTM cell output.

3.2 Training Data

Even though RNNs and its variants, like LSTM networks, exhibit a high degree of flexibility as they support multi-variate data and are able to handle sequences of variable length (both for the input and for the output), the specific network setup and the training strategy need to be designed around the training data. Nonetheless, we also have to consider the context and later application of the predictions, namely the grid-based SMPC method for trajectory planning in autonomous driving, to shape the format of the data. Different environment representation approaches are available, even within OGs, to model traffic scenarios. The selected representation algorithm is first presented and discussed in Section 3.2.1 and the collected data that is used for training and evaluation is examined thereafter in Section 3.2.2.

3.2.1 Dynamic Object Tracks

There are various possibilities for environment representations that can be chosen to train the prediction models. An essential feature of these representations is the level of abstraction of the presented data, depending on the closeness to the sensory information. The input data to the model can range from raw measurements to thoroughly processed OGs. For instance, evidential OGs additionally apply the Dempster-Shafer theory [Sha76] to fused sensor measurements to deliver probabilistic information about free, statically, and dynamically occupied space. Moreover, a method is described in [STW17] to derive object tracks from evidential OGs, by first transforming them into evidential DOGs. These object tracks allow a concise

description of dynamic agents on a road and are the environment representation of choice. The selection is reasoned after a previous introduction of the object tracking approach from [STW17].
The dynamic tracking algorithm operates on evidential OGs, to which a grid-based particle filter is applied to extract estimates of the velocities of dynamic objects, which are the TVs, to construct the evidential DOG. Thereby, the DOG represents the dynamic environment as a grid with six channels, where each grid cell $c \in \boldsymbol{\mathcal{G}}_k$ at time stamp k contains a measurement

$$\boldsymbol{\zeta}_{c,k} = \left[m(\boldsymbol{S}_{c,k}), m(\boldsymbol{D}_{c,k}), m\Big(\{\boldsymbol{S}, \boldsymbol{D}\}_{c,k}\Big), m(\boldsymbol{F}_{c,k}), \boldsymbol{v}_{c,k} \right]^\top \in \mathbb{R}^6, \tag{3.2}$$

with the static occupancy belief mass $m(\boldsymbol{S}_{c,k})$, the dynamic occupancy belief mass $m(\boldsymbol{D}_{c,k})$, the unclassified occupancy belief mass $m\Big(\{\boldsymbol{S}, \boldsymbol{D}\}_{c,k}\Big)$, the free space belief mass $m(\boldsymbol{F}_{c,k})$, and the two dimensional Cartesian velocity distribution $\boldsymbol{v}_{c,k} \in \mathbb{R}^2$ of the cell $c \in \boldsymbol{\mathcal{G}}_k$ at time step k. Next, dynamic cells are assigned individually to an object track, which we designate with the index τ for a total of $\mathcal{T}$ objects $\tau = 1, \ldots, \mathcal{T}$. New object tracks are founded on density-based dynamic cell clustering and are updated over time using a non-linear constant turn rate and acceleration motion model to describe clothoid paths, where the object geometry is approximated by an oriented minimum bounding box. After using an unscented Kalman filter on the object tracks, the set of cell measurements $\mathcal{Z}_{\tau,k} = \{\boldsymbol{z}_{c,k} \mid c \in \mathcal{C}_{\tau,k}\}$ associated to a track τ at step k, where $\mathcal{C}_{\tau,k} \subseteq \boldsymbol{\mathcal{G}}_k$ defines the set of cell indices belonging to the track τ at time stamp k, is mapped to a measurement vector

$$\boldsymbol{z}_{\tau,k} = [x_{\tau,k}, y_{\tau,k}, v_{\tau,k}, \phi_{\tau,k}, l_{\tau,k}, w_{\tau,k}]^\top \in \mathbb{R}^6, \tag{3.3}$$

with the the track τ and time k specific elements $[x_{\tau,k}, y_{\tau,k}]^\top$ as the two dimensional reference position coordinates on the grid, the scalar velocity $v_{\tau,k}$ and angle $\phi_{\tau,k}$, and the length $l_{\tau,k}$ and width $w_{\tau,k}$ according to the bounding box model of the object described by track τ.
We additionally define the time series $\mathcal{S}_\tau$ of measurement vectors $\boldsymbol{z}_{\tau,k}$ as in (3.3), which describes the object track of an object τ over all the sampling steps k:

$$\mathcal{S}_\tau = \{\boldsymbol{z}_{\tau,k} \mid, \ \forall k\}\,. \tag{3.4}$$

Then, we can construct the set $\mathcal{T}_{\text{env}}$ of time series $\mathcal{S}_\tau$, which describes all object tracks of the dynamic elements in a driving environment for an arbitrary period of time:

$$\mathcal{T}_{\text{env}} = \{\mathcal{S}_\tau \mid \tau = 1, \ldots, \mathcal{T}\}\,. \tag{3.5}$$

Overall, the set $\mathcal{T}_{\text{env}}$ is a robust representation of the dynamic part of driving scenarios and is the selected data format to develop and train the prediction models. Although environment representations that are closer to the sensor output would reduce the risk

of errors in the early stages of the environment estimation pipeline propagating to the next stages, thus leading to an overall deterioration of performance [MR19], dynamic object tracks are an excellent alternative for vehicle trajectory prediction with deep learning. First, there are two major disadvantages of raw sensor data compared to DOGs and derivatives: not only do the occupancy and velocity distributions on the DOG rely on sophisticated sensor fusion techniques that are difficult to replace with learning-based approaches, but DOGs also have a format that is independent of the sensor configuration [HBD17], which broadens the scope of uses of the grid prediction model. Furthermore, the object tracks from [STW17] additionally separate dynamic from static objects, which enables to train the learning algorithms to concentrate on dynamic objects. Otherwise, the prediction models could be biased towards static forecasts, since driving environments are typically composed mostly of static objects [MR19].

3.2.2 Data Collection

Before training the prediction models, measurement data is generated, collected, and processed in a simulation environment. In the initial stage, raw measurements are generated with the help of the microscopic traffic simulator SUMO (Simulation of Urban MObility) [LBBW+18], which models individual vehicles and their interactions at a microscopic level in a space-continuous, time-discrete setting. The simulation scenarios are highly customizable and different dynamic, behavioral, and probabilistic models are available to configure the environment and traffic.
We designed 32 independent road scenarios of a straight two-lane highway that vary in the amount, type, and density of traffic participants, as well as the environment conditions. The shared layout of the roadway consists of a 3000 m long one-way highway divided into two 3.5 m wide lanes with a maximum permissible speed of 33 m/s and where lane changes are allowed along the entire length of the route. The scenarios are intended to imitate and cover most of the situations that can be observed in real driving circumstances, such as different ranges of road congestion, partial obstruction of roadways, diverse uses of lanes, individual and group driving behavior patterns, and varying vehicle conditions.
Each driving scenario is scanned in an egocentric view, where all the entities within the local environment of the EV are sampled. The local environment is bounded by the detection range of 100 m of the EV, which results in the environment length $\iota_x = 2 \times 100\,\mathrm{m} = 200\,\mathrm{m}$. As the width of the two lanes on the highway is 3.5 m, the width of the sampled environment is $\iota_y = 2 \times 3.5\,\mathrm{m} = 7\,\mathrm{m}$. The size of the grid cells is set to $a_x = 0.5\,\mathrm{m}$ and $a_y = 0.25\,\mathrm{m}$, yielding a rectangular grid with the shape $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{400\times 28}$. During the simulation, objects on the road may enter or leave the detection zone freely. Tracking the local environment of a particular vehicle instead of simulating a stationary sensor is advantageous considering that the grid-based SMPC trajectory planning algorithm later operates in an egocentric perspective as well. The processing of the raw position, velocity, and orientation measurements

from the simulation into a set $\mathcal{T}_{\text{env}}$ is performed in MATLAB®.
We choose the sampling and prediction rate to be equal and denote the constant period duration with $T = 0.2\,\text{s}$. Hence, every scenario is recorded for 72 s at 5 Hz, which yields a total of $32 \times 72\,\text{s} = 2304\,\text{s}$ of collected data. The 32 sequences are then partitioned into 9 segments of 8 s each, where the first 4 s are used to initialize the prediction model and the remaining 4 s are employed to evaluate the forecasts. The total $32 \times 9 = 288$ sequence samples are further split randomly into a training data set, which contains 80% of the sequence samples (231 samples), a validation data set, composed by 10% of the sequence samples (29 samples), and a test data set with the remaining 10% (28 samples).

3.3 Multi-Class Classification

In the following, the classification problem is inspected in Section 3.3.1, then the beam search technique is presented in Section 3.3.2, the actual network architecture and training methodology for the classification model are introduced in Section 3.3.3, a method to derive a probabilistic representation for the grid-based SMPC method is proposed in 3.3.4 and the general approach is evaluated in 3.3.5.

3.3.1 Problem Definition

General Objective

In a multi-step classification problem, the overall target is to learn to estimate the conditional probability distribution $\Pr(\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N | \boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M)$ of an output sequence $\{\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N\}$ of length N, given the input sequence $\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\}$ of length M. Probabilistic predictions about the future can then be made by picking an output sequence $\{\boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_N^*\}$ with a high probability value. Due to the recursive principle that we follow to achieve multi-step predictions, introduced in Section 3.1, at every inference step h, the model disposes of information about the observation sequence $\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\}$ and additionally about the previous $h-1$ forecasts $\{\boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-1}^*\}$. Hence, the aim is to leverage all the information available at every prediction instant h to approximate the conditional probability $\Pr(\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N | \boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M)$ as:

$$\Pr(\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N | \boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M) \approx \prod_{h=1}^{N} \Pr(\boldsymbol{\mathcal{Y}}_h | \boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M, \boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-1}^*), \quad (3.6)$$

where the conditional probability $\Pr(\boldsymbol{\mathcal{Y}}_h | \boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M, \boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-1}^*)$ is estimated sequentially at every single-step forecast h. For the case $h = 1$, there are no previous forecasts available and the evidence of the conditional probability in (3.6) is just the the observation sequence $\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\}$. Notably, at step h, the recurrent structure of the LSTM network encodes the sequence $\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M, \boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-2}^*\}$ of past inputs until step $h-2$, regardless of whether the inputs were actual observations or previous predictions, into the internal states $\boldsymbol{m}_{h-1}$ and $\boldsymbol{h}_{h-1}$, where

the cell memory $\boldsymbol{m}_{h-1}$ acts fundamentally as a long-term memory and the hidden state $\boldsymbol{h}_{h-1}$ as a short-term memory. Consequently, the probability distribution $\Pr(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{\mathcal{X}}_1,\ldots,\boldsymbol{\mathcal{X}}_M,\boldsymbol{\mathcal{Y}}_1^*,\ldots,\boldsymbol{\mathcal{Y}}_{h-1}^*)$ at step h is successively approximated to

$$\Pr(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{\mathcal{X}}_1,\ldots,\boldsymbol{\mathcal{X}}_M,\boldsymbol{\mathcal{Y}}_1^*,\ldots,\boldsymbol{\mathcal{Y}}_{h-1}^*) \approx \Pr(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1},\boldsymbol{h}_{h-1},\boldsymbol{\mathcal{Y}}_{h-1}^*), \tag{3.7}$$

given the cell states $\boldsymbol{m}_{h-1}$ and $\boldsymbol{h}_{h-1}$ and the sample $\boldsymbol{\mathcal{Y}}_{h-1}^*$ of the output sequence at step $h-1$. Ultimately, the results from (3.6) and (3.7) lead to the following approximation of the probability $\Pr(\boldsymbol{\mathcal{Y}}_1,\ldots,\boldsymbol{\mathcal{Y}}_N|\boldsymbol{\mathcal{X}}_1,\ldots,\boldsymbol{\mathcal{X}}_M)$:

$$\Pr(\boldsymbol{\mathcal{Y}}_1,\ldots,\boldsymbol{\mathcal{Y}}_N|\boldsymbol{\mathcal{X}}_1,\ldots,\boldsymbol{\mathcal{X}}_M) \approx \prod_{h=1}^{N} \Pr(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1},\boldsymbol{h}_{h-1},\boldsymbol{\mathcal{Y}}_{h-1}^*). \tag{3.8}$$

In essence, (3.8) describes the predictive modeling task of our multi-step classification problem as a succession of single-step deductions as in (3.7), where the output $\Pr(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1},\boldsymbol{h}_{h-1},\boldsymbol{\mathcal{Y}}_{h-1}^*)$ of the LSTM-based network at every prediction step h is a multinoulli distribution that models the probability of each possible outcome, represented by a categorical label, individually as a discrete probability distribution. In a multi-class classification, the multinomial categorical distribution at the network output is obtained by applying the softmax function σ_{softmax} to the vector of activations $\boldsymbol{x} = [x_1,\ldots,x_\Lambda]^\top \in \mathbb{R}^\Lambda$ of the last network layer:

$$\sigma_{\text{softmax}} : \mathbb{R}^\Lambda \to \left\{ \sigma(\boldsymbol{x}) \in (0,1)^\Lambda \mid \sigma(\boldsymbol{x})_j \geq 0,\ \sum_{\lambda=1}^{\Lambda} \sigma(\boldsymbol{x})_j = 1 \right\}, \tag{3.9a}$$

$$\sigma_{\text{softmax}}(\boldsymbol{x})_\lambda = \frac{e^{x_\lambda}}{\sum_{i=1}^{\Lambda} e^{x_i}}, \tag{3.9b}$$

for $\lambda = 1,\ldots,\Lambda$, assuming $\Lambda \geq 2$. Thereby, the network output is a normalized vector that shares the properties of a probability distribution, as indicated in the codomain in (3.9a), and models the distribution of a categorical random variable with Λ possible categorical outcomes, where the i-th class is coded as a one-hot vector, which is a vector with a one in the i-th element and zeros elsewhere.

Input and Output

Furthermore, the selected random variable predicted at the network output has to be a discrete and dependent variable, given a group of independent variables, that can be modeled categorically. According to the discussion in Section 3.2.1, the set $\mathcal{T}_{\text{env}}$ of object tracks $\mathcal{S}_\tau$ for $\tau = 1,\ldots,\mathcal{T}$ from (3.5) is the chosen representation of the dynamic environment. Therefore, the measurement vector $\boldsymbol{z}_{\tau,k}$ of track τ at time step k from (3.3) can be put in sequentially to the network to predict the future of a single object track. Ergo, we define the observation $\boldsymbol{\mathcal{X}}_m$ at observation step $m = 1,\ldots,M$ that is used to form the evidence sequence $\{\boldsymbol{\mathcal{X}}_1,\ldots,\boldsymbol{\mathcal{X}}_M\}$ for the multi-class classification model as

$$\boldsymbol{\mathcal{X}}_m = \boldsymbol{z}_{\tau,k-M+m} \in \mathbb{R}^6, \tag{3.10}$$

where k denotes the current sampling instant and we omit the subscript τ from the observations $\boldsymbol{\mathcal{X}}_m$ and later from the targets $\boldsymbol{\mathcal{Y}}_m$ for simplicity. However, it is computationally inefficient to discretize all the variables represented by the vector elements in $\boldsymbol{z}_{\tau,k}$ to model the object track as a categorical random variable at the predictor output as well, as the number of classes increases exponentially with each additional variable. Notwithstanding, we can take advantage of the already discrete reference position $[x_{\tau,k}, y_{\tau,k}]^\top$ at the center of the front side of the box model of an object track τ, which is associated to the coordinates of a cell $c_{i,j,k} \in \boldsymbol{\mathcal{G}}_k$ on the OG $\boldsymbol{\mathcal{G}}_k$ at time stamp k, to express the predicted position of the vehicle as a categorical vector that describes $\Lambda = \frac{\iota_x}{a_x} \times \frac{\iota_y}{a_y}$ possible outcomes. In turn, the label $\boldsymbol{\mathcal{Y}}_m$ at observation step $m = 1, \ldots, M$ that is employed to form the label sequence $\{\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_M\}$ used for training is defined as a categorical vector, given by

$$\boldsymbol{\mathcal{Y}}_m = \mathrm{Proj}_1 \left([x_{\tau,k-M+m+1}, y_{\tau,k-M+m+1}]^\top \right) \in \{0,1\}^\Lambda , \tag{3.11}$$

where k is the current sampling stamp and the projection $\mathrm{Proj}_1 : \mathbb{R}^2 \to \{0,1\}^\Lambda$ maps the position $[x_{\tau,k-M+m+1}, y_{\tau,k-M+m+1}]^\top$ of the dynamic object represented by track τ one time step after the measurement vector in $\boldsymbol{\mathcal{X}}_m$ from (3.10), into a one-hot encoded categorical vector $\boldsymbol{\mathcal{Y}}_m$ of length Λ, which is the cardinality of the possible outcomes at a prediction step h. The temporal shift of one time step between the inputs $\boldsymbol{\mathcal{X}}_m$ and the labels $\boldsymbol{\mathcal{Y}}_m$ trains the prediction model to perform single-step forecasts.

In summary, the classification task of the grid prediction model is to successively estimate the categorical probability distribution $\Pr(\boldsymbol{\mathcal{Y}}_h | \boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}_{h-1})$ of the next position of an object on the OG, given the past object track as evidence. The search technique to select an outcome $\boldsymbol{\mathcal{Y}}_h^*$ from the categorical distribution is reviewed later in Section 3.3.2.

After that, the next inference step of the multi-step prediction is repeated as a single-step forecast, where the previously generated output $\boldsymbol{\mathcal{Y}}_{h-1}^*$ is fed back to the input of the predictor as new evidence, as noted in (3.1). However, the model output and input have diverse data formats and represent different information. Therefore, we need a second projection $\mathrm{Proj}_2 : \{0,1\}^\Lambda \to \mathbb{R}^6$, given by

$$\hat{\boldsymbol{z}}_{\tau,h-1} = \mathrm{Proj}_2 \left(\boldsymbol{\mathcal{Y}}_{h-1}^* \right) , \tag{3.12}$$

which maps the categorical output $\boldsymbol{\mathcal{Y}}_{h-1}^*$ of the model at prediction step $h-1$ into an approximative measurement vector

$$\hat{\boldsymbol{z}}_{\tau,h-1} = \left[\hat{x}_{\tau,h-1}, \hat{y}_{\tau,h-1}, \hat{v}_{\tau,h-1}, \hat{\phi}_{\tau,h-1}, \hat{l}_{\tau,h-1}, \hat{w}_{\tau,h-1} \right]^\top \in \mathbb{R}^6, \tag{3.13}$$

following the definition from (3.3), that can be used as an input $\boldsymbol{\mathcal{X}}_h$ to the model at step h

$$\boldsymbol{\mathcal{X}}_h \approx \hat{\boldsymbol{z}}_{\tau,h-1}. \tag{3.14}$$

In consequence, the equation (3.1) is approximated as $\boldsymbol{\mathcal{X}}_h \approx \mathrm{Proj}_2\left(\boldsymbol{\mathcal{Y}}^*_{h-1}\right)$ with the help of Proj_2. In the first place, the grid cell coordinates $[x_{\tau,h-1}, y_{\tau,h-1}]^\top$ associated to the chosen output category $\boldsymbol{\mathcal{Y}}^*_{h-1}$ at prediction step $h-1$ are directly obtained with the inverse of the first projection $\mathrm{Proj}_1^{-1} : \{0,1\}^\Lambda \to \mathbb{R}^2$ as

$$[\hat{x}_{\tau,h-1}, \hat{y}_{\tau,h-1}]^\top = \mathrm{Proj}_1^{-1}\left(\boldsymbol{\mathcal{Y}}^*_{h-1}\right). \tag{3.15}$$

Next, the scalar velocity $\hat{v}_{\tau,h}$ and orientation $\hat{\phi}_{\tau,h}$ measures are approximated as the polar representation of the estimated velocity vector $[\Delta\hat{x}_{\tau,h}, \Delta\hat{y}_{\tau,h}]^\top$, which is calculated as

$$[\Delta\hat{x}_{\tau,h-1}, \Delta\hat{y}_{\tau,h-1}]^\top = \Big([\hat{x}_{\tau,h-1}, \hat{y}_{\tau,h-1}]^\top - [\hat{x}_{\tau,h-2}, \hat{y}_{\tau,h-2}]^\top\Big)/T, \tag{3.16}$$

where $[\hat{x}_{\tau,h-1}, \hat{y}_{\tau,h-1}]^\top$ and $[\hat{x}_{\tau,h-1}, \hat{y}_{\tau,h-2}]^\top$ are the grid cell coordinates associated to the chosen output category at prediction steps $h-1$ and $h-2$ respectively and are obtained with (3.15), while T designates the sampling and inference times simultaneously. Then, the remaining elements of the approximative measurement vector $\hat{\boldsymbol{z}}_{\tau,h-1}$ of the dynamic object track τ at prediction step $h-1$ are estimated according to the following procedure, which together with (3.15) forms the projection Proj_2 from (3.12):

$$\hat{v}_{\tau,h-1} = \left\| [\Delta\hat{x}_{\tau,h-1}, \Delta\hat{y}_{\tau,h-1}]^\top \right\|_2, \tag{3.16a}$$

$$\hat{\phi}_{\tau,h-1} = \mathrm{atan2}\left(\Delta\hat{y}_{\tau,h-1}, \Delta\hat{x}_{\tau,h-1}\right), \tag{3.16b}$$

$$\left[\hat{w}_{\tau,h-1}, \hat{l}_{\tau,h-1}\right]^\top = [w_{\tau,h-2}, l_{\tau,h-2}]^\top, \tag{3.16c}$$

given the grid cell coordinates $c_{i,j,h-1}$, the estimated velocity vector $[\Delta\hat{x}_{\tau,h-1}, \Delta\hat{y}_{\tau,h-1}]^\top$ from (3.16), and the object width $w_{\tau,h-2}$ and length $l_{\tau,h-2}$ at step $h-2$. The operator $\|\boldsymbol{v}\|_2 = \sqrt{v_1^2 + \cdots + v_{n_{\boldsymbol{v}}}^2}$ denotes the Euclidean norm of an arbitrary vector $\boldsymbol{v} = [v_1, \ldots, v_{n_{\boldsymbol{v}}}]^\top \in \mathbb{R}^{n_{\boldsymbol{v}}}$ with $n_{\boldsymbol{v}} \in \mathbb{N}$ and the four-quadrant inverse tangent $\mathrm{atan2} : \mathbb{R}^2 \to (-\pi, \pi]$ is used to approximate the orientation of the object as its estimated yaw angle:

$$\mathrm{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right), & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi, & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi, & \text{if } x < 0 \text{ and } y < 0, \\ \frac{\pi}{2}, & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2}, & \text{if } x = 0 \text{ and } y < 0, \\ 0, & \text{if } x = 0 \text{ and } y = 0. \end{cases} \tag{3.17}$$

3.3.2 Beam Search Algorithm

Multiple search approaches are introduced in [Neu17] to select an output $\boldsymbol{\mathcal{Y}}^*_h$ from the categorical distribution $\Pr\left(\boldsymbol{\mathcal{Y}}_h | \boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}^*_{h-1}\right)$ at the output of the network

at prediction step h as the next single-step prediction. The greedy best search algorithm that simply picks the candidate that maximizes the probability distribution $\Pr\left(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}^*_{h-1}\right)$ at every inference step h as

$$\boldsymbol{\mathcal{Y}}^*_h = \arg\max_{\boldsymbol{\mathcal{Y}}_h} \Pr\left(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}^*_{h-1}\right) \tag{3.18}$$

is typically the choice for single-step classification tasks. However, we perform the single classification step successively and thus aim to maximize the conditional probability $\Pr\left(\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N|\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\right)$ of the total output sequence $\{\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N\}$, which may not always be the result of optimizing the single-step probability distribution at all prediction steps h as in (3.18). For instance, it could be the case that a false initial prediction leads to less likely or uncertain events that would otherwise not have been considered with a different initial forecast. In this scenario, the error from the early prediction is propagated to the following steps. Alternatively, the categorical distribution $\Pr\left(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}^*_{h-1}\right)$ could be a multimodal distribution, which is often the case in nature and real-life situations like driving environments, where different outcomes $\boldsymbol{\mathcal{Y}}_h$ form equally good hypotheses.
To reduce the impact of the potential production of false or uncertain predictions due to the maximization of the one-step probability distribution, the beam search algorithm has become popular in the fields of neural machine translation and text generation [Neu17]. Instead of selecting a single hypothesis $\boldsymbol{\mathcal{Y}}^*_h$ at each inference step h, the beam search algorithm chooses and keeps the $W \in \mathbb{N}$ best candidates $\boldsymbol{\mathcal{Y}}^*_{h,\omega}$, where W denotes the beam width and $\omega = 1, \ldots, W$. For this purpose, we first compute the W output probability distributions $\Pr\left(\boldsymbol{\mathcal{Y}}_{h,\omega}|\boldsymbol{m}_{h-1,\omega}, \boldsymbol{h}_{h-1,\omega}, \boldsymbol{\mathcal{Y}}^*_{h-1,\omega}\right)$ at prediction step h, given the W LSTM states $\boldsymbol{m}_{h-1,\omega}$ and $\boldsymbol{h}_{h-1,\omega}$, and the W previous prediction hypotheses $\boldsymbol{\mathcal{Y}}^*_{h-1,\omega}$. Then, a score vector $\boldsymbol{s}_{h,\omega} \in \mathbb{R}^{\Lambda}_{<0}$ is assigned to each of these W vector probability distributions by computing the element-wise natural logarithm $\log : \mathbb{R}^{\Lambda}_{>0} \to \mathbb{R}^{\Lambda}$ of the probability distribution:

$$\boldsymbol{s}_{h,\omega} = \log\left(\Pr\left(\boldsymbol{\mathcal{Y}}_{h,\omega}|\boldsymbol{m}_{h-1,\omega}, \boldsymbol{h}_{h-1,\omega}, \boldsymbol{\mathcal{Y}}^*_{h-1,\omega}\right)\right). \tag{3.19}$$

In this way, the score of an arbitrary sequence of outputs can be obtained as the sum of the respective individual scores. For instance, the probabilities of the best W previous output sequence hypotheses $\left\{\boldsymbol{\mathcal{Y}}^*_{1,\omega}, \ldots, \boldsymbol{\mathcal{Y}}^*_{h-1,\omega}\right\}$ at prediction step $h-1$, which are given by

$$\Pr\left(\boldsymbol{\mathcal{Y}}^*_{1,\omega}, \ldots, \boldsymbol{\mathcal{Y}}^*_{h-1,\omega}|\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\right) \approx \prod_{i=1}^{h-1} \Pr\left(\boldsymbol{\mathcal{Y}}^*_{i,\omega}|\boldsymbol{m}_{i-1,\omega}, \boldsymbol{h}_{i-1,\omega}, \boldsymbol{\mathcal{Y}}^*_{i-1,\omega}\right), \tag{3.20}$$

are each assigned a score $s^{\text{seq}}_{h-1,\omega} \in \mathbb{R}^-$ by summing up all the scores $\boldsymbol{s}_{i,\omega,\boldsymbol{\mathcal{Y}}^*_{i,\omega}}$ of the previously chosen sequence samples $\boldsymbol{\mathcal{Y}}^*_{i,\omega}$ for $i = 1, \ldots, h-1$ as

$$s^{\text{seq}}_{h-1,\omega} = \sum_{i=1}^{h-1} \boldsymbol{s}_{i,\omega,\boldsymbol{\mathcal{Y}}^*_{i,\omega}}. \tag{3.21}$$

Finally, the overall scores of all $W \times \Lambda$ sequence candidates at step h are obtained as the sum $\boldsymbol{s}_{h,\omega} + s^{\text{seq}}_{h-1,\omega}$ of the score vectors $\boldsymbol{s}_{h,\omega}$ from (3.19) of new potenital outcomes and the scalar value $s^{\text{seq}}_{h-1,\omega}$ from (3.21) that assigns a score to each of the W sequences of previous outcomes until step $h-1$. Among these, only the W candidates with the highest total scores are chosen as possible predictions $\boldsymbol{\mathcal{Y}}^*_{i,\omega}$, effectively pruning the remaining $W \times \Lambda - W$ candidate sequences from the search tree. An example of the beam search algorithm, where the greedy best search approach would fail to find the best output sequence, is shown in Figure 3.2.

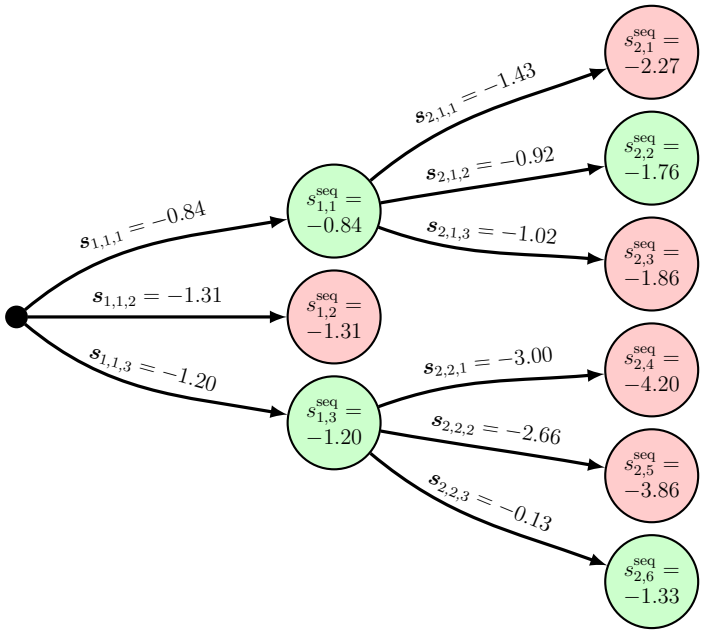


Figure 3.2: Beam search on an example search tree for $N = 2$ prediction steps, where the number of possible output categories Λ is $\Lambda = 3$ and the beam width W is chosen to $W = 2$. Green nodes are selected by the beam search, red nodes are pruned. The symbol $\boldsymbol{s}_{h,\omega,j}$ represents the j-th element of the w-th score vector for prediction step h and $s^{\text{seq}}_{h,i}$ is the score of the i-th sequence candidate at step h, where $h = 1, \ldots, N$, $\omega = 1, \ldots, W$, $\lambda = 1, \ldots, \Lambda$, and $i = 1, \ldots, W \times \Lambda$. Figure inspired by [Neu17].

3.3.3 Network Architecture and Training Procedure

Network Architecture

The neural network used for classification has a rather shallow architecture, with one LSTM layer followed by two stacked Fully Connected (FC) feedforward layers. The observed data is put sequentially into an input layer of size 6, which is the length of the measurement vectors from (3.3). Then, the core LSTM processes the multivariate time-series data into the latent space vectors $\boldsymbol{m}_h \in \mathbb{R}^{512}$ and $\boldsymbol{h}_h \in \mathbb{R}^{512}$ at a step h. Lastly, the two FC layers, one of size 512 and with the hyperbolic tangent tanh

activation function, and the other of size 11200 and with the final softmax activation function, help to capture spatial dependencies and construct the output categorical distribution. The output length of 11200 corresponds to the number of grid cells in the grid $\boldsymbol{\mathcal{G}}$ for the later evaluation setup and the number of classes is therefore also given as $\Lambda = 11200$. The network architecture is illustrated in Figure 3.3.

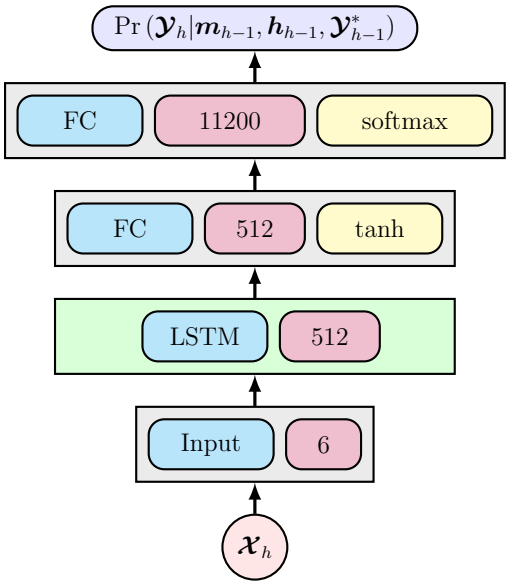


Figure 3.3: Architecture of the LSTM network used to predict the future of dynamic traffic scenarios as a classification task. The input $\boldsymbol{\mathcal{X}}_h$ is a vector with 6 elements and the output $\Pr(\boldsymbol{\mathcal{Y}}_h|\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}^*_{h-1})$ is a vector probability distribution of length 11200. For the evidence of the conditional probability at the network output, we assume that (3.1) is given. Every layer is represented as a container with at least two elements: the layer type in cyan, the output size in purple, and optionally the activation function in yellow. The width of the container varies depending on the number of units on the layer and the activations are propagated between the layers, as depicted by the vertical arrows.

Training Procedure

During the initialization and prediction stages, the network produces a total of $M + N - 1$ single-step predictions, where the first $M - 1$ forecasts (during the initialization) are discarded, as described in Section 3.1. Thus, the network is trained to produce $M + N - 1$ single-step predictions by choosing the observation sequence length M^{train} and the prediction horizon N^{train} for training as

$$M^{\text{train}} = M + N - 1, \qquad N^{\text{train}} = 1. \tag{3.22}$$

Later, during the prediction phase, the first $N - 1$ network outputs are treated as new evidence, which justifies training the network with a longer observation length as in (3.22).

We use the Adam [KB14] solver to train the presented neural network. The training algorithm performs a stochastic optimization of the network parameters based on the error measure given by the loss function L. The training hyperparameters, as well as the previously exposed network architecture and hyperparameters, are tuned with the the Experiment Manager app of the Deep Learning Toolbox™ [TM21] in MATLAB®, based on an automated empirical optimization process that tries out multiple different combinations exhaustively. The initial learning rate is set to 0.002 and is updated every 90 epochs by multiplying with 0.2. The size of the mini-batch is chosen to 128 and the training process is terminated whenever the generalization stops to improve, which we consider when the loss on the validation data set, measured every 10 iterations, does not decrease for 5 consecutive times.
As the loss function, we make use of the cross-entropy loss for single-label classification and weighted classification tasks with mutually exclusive classes that comes with the Deep Learning Toolbox™:

$$L_{\text{class}} = -\frac{1}{M^{\text{train}}} \sum_{m=1}^{M^{\text{train}}} \sum_{\lambda=1}^{\Lambda} \gamma_\lambda t_{m,\lambda} \log \rho_{m,\lambda}, \tag{3.23}$$

with the weight γ_λ for class λ, the indicator $t_{m,\lambda}$ that the m-th observation belongs to the λ-th class, and $\rho_{m,\lambda}$ is the output for sample m for class i, which is the value from the softmax function.

3.3.4 Probability Grid Derivation

This section presents a technique to find an alternative grid representation to the OG that additionally models probabilistic uncertainties, based on the raw output of the grid prediction module. Building on top of the standard OG, a Probability Grid (PG) is computed at every prediction step h to model the local environment stochastically. The PG consists of a grid $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{\frac{\iota_x}{a_x} \times \frac{\iota_y}{a_y}}$ of occupancy values $p_{i,j} \in \boldsymbol{\mathcal{P}}$ that describe the likelihood of the cell being occupied. However, this value is the result of multiple operations on different probability density functions and hence does not correspond to the probability value stored at the analogous cell $c_{i,j}$ in the OG. In this section, we propose an approach to build a PG from the output of the classification multi-step prediction module.
The advantage of modeling the multi-step prediction task as a classification problem is that the categorical distributions $\Pr\left(\boldsymbol{\mathcal{Y}}_h | \boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}^*_{h-1}\right)$ at the output of the softmax layer at every prediction step h, i.e., before choosing an outcome $\boldsymbol{\mathcal{Y}}^*_h$, already have the format of a probability grid map unrolled as a Λ dimensional vector. However, we have $\mathcal{T} \times W$ such probability distributions with W sequence hypotheses for each of the $\mathcal{T} \in \mathbb{N}$ TVs in the scenario. Consequently, the $\mathcal{T} \times W$ categorical probability distributions at the end of the neural network prediction pipeline at every prediction stamp h, which we abbreviate with

$$\boldsymbol{\mathcal{P}}_{h,\tau,\omega} = \Pr\left(\boldsymbol{\mathcal{Y}}_{h,\tau,\omega} | \boldsymbol{m}_{h-1,\tau,\omega}, \boldsymbol{h}_{h-1,\tau,\omega}, \boldsymbol{\mathcal{Y}}^*_{h-1,\tau,\omega}\right), \tag{3.24}$$

where $\tau \in \mathcal{T}$ and $\omega = 1, \ldots, W$, have to ultimately be aggregated to form a single and valid PG $\boldsymbol{\mathcal{P}}_h$.

Union of Hypotheses

The first step is to combine the W hypotheses available at a prediction step h for a TV τ. The PGs $\boldsymbol{\mathcal{P}}_{h,\tau,\omega}$ are aggregated as a weighted sum

$$p_{i,j,h,\tau} = \sum_{w=1}^{W} \varepsilon_\omega p_{i,j,h,\tau,\omega}, \tag{3.25}$$

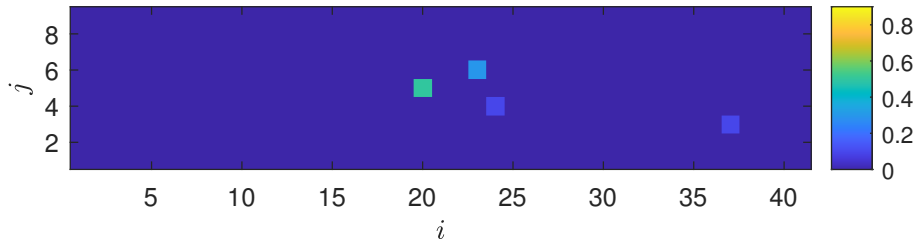
where the weights ε_ω are obtained as the probability of the w-th hypothesis sequence $\exp\left(s^{\text{seq}}_{N,\omega}\right)$, given the sequence score $s^{\text{seq}}_{N,\omega}$ from (3.21) at prediction step N, normalized as

$$\varepsilon_\omega = \frac{\exp\left(s^{\text{seq}}_{N,\omega}\right)}{\sum_{i=1}^{W} \exp\left(s^{\text{seq}}_{N,i}\right)}, \tag{3.26}$$

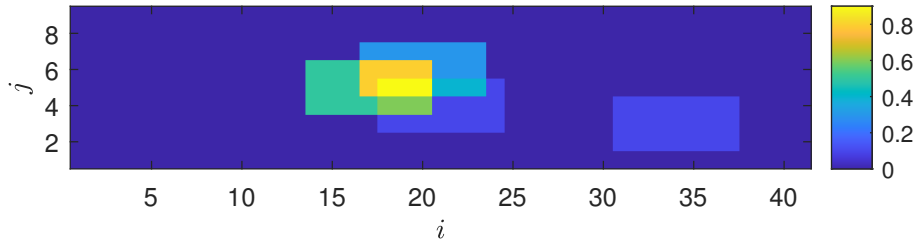
for $\omega = 1, \ldots, W$. As a result, the previous $\mathcal{T} \times W$ PGs $\boldsymbol{\mathcal{P}}_{h,\tau,\omega}$ are reduced into $\mathcal{T}$ PGs $\boldsymbol{\mathcal{P}}_{h,\tau}$ at every prediction step h.

Grid Filtering

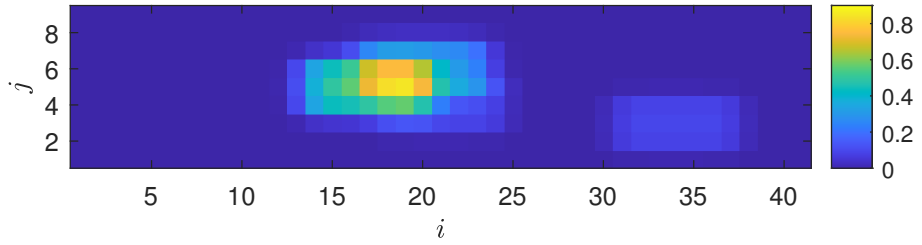
Before the $\mathcal{T}$ probability distributions $\boldsymbol{\mathcal{P}}_{h,\tau}$ are also united into a single PG $\boldsymbol{\mathcal{P}}_h$, each of the aggregated categorical distributions $\boldsymbol{\mathcal{P}}_{h,\tau}$ at every step h first needs to undergo some adaptations. There are two issues that need to be addressed to produce a valid PG that can later be used for trajectory planning. First, the probabilities at the predictor output belong to the random variable that models the presence of a point-mass object in each cell of the grid. Conversely, the PG that is used to derive the drivable space in the grid-based SMPC routine models for every grid cell the likelihood of the cell being occupied by any box object in the surroundings of the EV. Second, no prediction uncertainty is given by the deep learning model, while the grid-based SMPC scheme additionally provides a probabilistic framework to account for possible prediction errors by considering uncertainty in the forecasts.
The first concern can be approached by expanding the probability values $p_{i,j,h,\tau} \in \boldsymbol{\mathcal{P}}_{h,\tau}$ of the $\mathcal{T}$ distributions at step h additively along the box dimensions of the TV τ. Furthermore, a probabilistic uncertainty model can be derived based on a measure of model confidence during the multi-step predictive task and then applied to the PG to solve the second subject.
Contemplating a PG as a single channel pixel map, we approach the PG modification task with image processing techniques. In particular, we apply a series of low-pass blurring filters, each of which fulfills a different function. For simplicity, we omit the subscripts h and τ in the following discussion. Also, we consider the PG $\boldsymbol{\mathcal{P}}$ to be resized to the size $\frac{\iota_x}{a_x} \times \frac{\iota_y}{a_y}$ of the OG $\boldsymbol{\mathcal{G}}$, as introduced in Section 2.2. An artificial example of such a categorical probability distribution is illustrated in Figure 3.4a.



(a) Artificial categorical distribution, reshaped as a grid map. Each cell holds a value $p_{j,j,h,\tau,\omega} \in \boldsymbol{\mathcal{P}}_{h,\tau,\omega}$ that models the likelihood of the presence of a point-mass object on the grid cell $c_{i,j}$.



(b) Result after box filtering to represent the TV as a box object. The box model size of the TV is chosen exemplarily to $l = 7a_x$ and $w = 3a_y$.



(c) Gaussian filtering to model prediction uncertainty. The variance measures σ_x^2 and σ_y are set exemplarily to $\sigma_x^2 = 0.5$ and $\sigma_y^2 = 0.75$.

Figure 3.4: The qualitative effect of the box and Gaussian filtering on a categorical probability grid map.

In general, the principle of the filtering procedures used in this work is to convolute a two dimensional kernel $\boldsymbol{\mathcal{K}}$ over an image, which is in this case the PG $\boldsymbol{\mathcal{P}}$. The two dimensional discrete convolution of a PG $\boldsymbol{\mathcal{P}}$ with probability values $p_{i,j} \in \boldsymbol{\mathcal{P}}$ with an arbitrary kernel $\boldsymbol{\mathcal{K}}$ with elements $k_{i,j} \in \boldsymbol{\mathcal{K}}$, where the indices i and j indicate the two spatial dimensions, is given by

$$p_{i,j} = \sum_{q=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} p_{q,r} \kappa_{i-q+1,j-r+1}. \tag{3.27}$$

For the purpose of extending the probability values of the modeled point-mass according to the box model of the TV to cover the vehicle area, we apply a box linear filter with a two dimensional kernel $\boldsymbol{\mathcal{K}}^{\text{box}} \in \{0,1\}^{l_x^{\text{box}} \times l_y^{\text{box}}}$ to the PG $\boldsymbol{\mathcal{P}}$, where $l_x^{\text{box}} \in \mathbb{N}$ and $l_y^{\text{box}} \in \mathbb{N}$ denote the length and the width of box kernel. Every element

$\kappa_{i,j}^{\text{box}} \in \boldsymbol{\mathcal{K}}^{\text{box}}$ is assigned a value as follows:

$$\kappa_{i,j}^{\text{box}} = \begin{cases} 1, & \text{if } 1 \leq i \leq \left\lceil l_x^{\text{box}}/2 \right\rceil \text{ and } 1 \leq j \leq l_y^{\text{box}} \\ 0, & \text{otherwise,} \end{cases} \tag{3.28}$$

with he indices $i = 1, \ldots, l_x^{\text{box}}$ and $j = 1, \ldots, l_y^{\text{box}}$, the ceiling operation $\lceil \cdot \rceil$, and the kernel lengths l_x^{box} and l_y^{box} as

$$l_x^{\text{box}} = 2l/a_x - 1, \qquad l_y^{\text{box}} = w/a_y. \tag{3.29}$$

The parameters and the zero-elements in the kernel are designed such that the box linear filtering expands every probability value in $\boldsymbol{\mathcal{P}}$ along the bi-dimensional shape of the TV, considering the fact that the reference position of the TV (i.e., the position of the point of mass) is given as the grid cell coordinates associated to the center of the front side of the box model representation of the vehicle. We assume that the TV is oriented in the positive x direction, which results in an unoriented box model given by the kernel $\boldsymbol{\mathcal{K}}^{\text{box}}$. While this is sufficient to demonstrate the method in the later simulation setup, where the yaw angle ϕ of the TVs stays relatively close to zero, an oriented box kernel can also be estimated individually for each TV following the same approximative approach as in (3.16b). The effect accomplished by the presented box filtering procedure is depicted in Figure 3.4b.
Next, the prediction uncertainty must be integrated into the PG artificially, as the probabilistic values in $\boldsymbol{\mathcal{P}}$ do not contain any information about the model confidence. Thus, we can apply an additional smoothing filter, which allows to model uncertainty in the TV motion with an arbitrary probability distribution. In an exemplary way, we show the method with a Gaussian filter that uses a two dimensional Gaussian kernel $\boldsymbol{\mathcal{K}}^{\text{Gauss}} \in (0,1)^{l_x^{\text{Gauss}} \times l_y^{\text{Gauss}}}$, where $l_x^{\text{Gauss}} \in \mathbb{N}$ and $l_y^{\text{Gauss}} \in \mathbb{N}$ describe the length of the kernel in both spatial dimensions. We additionally define a help function $\text{center} : \boldsymbol{\mathcal{G}} \to \mathbb{N}^2$ that outputs the two spatial indices $[i, j]^\top$ belonging to a cell $c_{i,j}$ on the grid $\boldsymbol{\mathcal{G}}$. Then, the elements $\kappa_{i,j}^{\text{Gauss}} \in \boldsymbol{\mathcal{K}}^{\text{Gauss}}$ of the kernel are given by

$$\kappa_{i,j}^{\text{Gauss}} = \frac{\exp\left(-\frac{1}{2}\left(\text{center}\left(c_{i,j}\right) - \boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\text{center}\left(c_{i,j}\right) - \boldsymbol{\mu}\right)\right)}{\sqrt{(2\pi)^2 \det \boldsymbol{\Sigma}}}, \tag{3.30}$$

with the indices $i = 1, \ldots, l_x^{\text{Gauss}}$ and $j = 1, \ldots, l_y^{\text{Gauss}}$, the mean vector $\boldsymbol{\mu} \in \mathbb{N}^2$, the position covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times 2}$, and the precision matrix $\boldsymbol{\Sigma}^{-1} \in \mathbb{R}^{2\times 2}$. The mean $\boldsymbol{\mu}$, which is established as the center of the kernel, and the covariance $\boldsymbol{\Sigma}$ are defined as

$$\boldsymbol{\mu} = \begin{bmatrix} l_x^{\text{Gauss}}/2 \\ l_y^{\text{Gauss}}/2 \end{bmatrix}, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}. \tag{3.31}$$

The variance measures σ_x^2 and σ_y^2 are the square of the standard deviations σ_x and σ_y respectively, which can be chosen as an arbitrary prediction error value, measured

on the test data set. If chosen appropriately, the error measures can be interpreted as an expected deviation in each direction that can be used for the covariance matrix $\mathbf{\Sigma}$. Then, the size of the kernel can also be determined as

$$l_x^{\text{Gauss}} = 2\lceil 2\sigma_x \rceil + 1, \qquad l_y^{\text{Gauss}} = 2\lceil 2\sigma_y \rceil + 1. \tag{3.32}$$

The chosen size guarantees that l_x^{Gauss} and l_y^{Gauss} are odd integers and keeps the kernel dimensions manageably small, while still ensuring that most of the data in the Gaussian distribution, which is typically found within a distance of twice the standard deviation from the mean in each dimension, is considered. The qualitative effect that is accomplished with the Gaussian filter is illustrated in Figure 3.4c

After applying both filters to the PG, we normalize $\boldsymbol{\mathcal{P}}$ to share the properties of a probability distribution. Assuming that the probability values $p_{i,j}$ are all already non-negative ($p_{i,j} \geq 0$), which should be guaranteed by the previous steps, the PG is normalized by scaling every element $p_{i,j} \in \boldsymbol{\mathcal{P}}$ with the inverse of the sum of all the elements in $\boldsymbol{\mathcal{P}}$ as

$$p_{i,j} = \frac{p_{i,j}}{\sum_{p \in \boldsymbol{\mathcal{P}}} p}. \tag{3.33}$$

The described normalization step in (3.33) also guarantees that the values $p_{i,j}$ are less or equal to one ($0 \leq p_{i,j} \leq 1$), which is a requirement for the following modification to the PG.

Besides, it is worth to mention that we intentionally aggregate the W hypotheses before the filtering process to prioritize the computational efficiency and avoid W repetitions of the convolutional filtering. Otherwise, the same box filter could be applied to each of the W probability distributions $\boldsymbol{\mathcal{P}}_\omega$, followed by a Gaussian filter with a different uncertainty measure for each hypothesis.

Combination of Objects

For the rest of this section, we reintroduce the subscripts h and τ. We now consider each of the $\mathcal{T}$ distributions $\boldsymbol{\mathcal{P}}_{h,\tau}$ at every prediction step h to be valid PGs that are mature to be used in the grid-based SMPC trajectory planning method. As a final step, it remains to aggregate the $\mathcal{T}$ forecasts made for each TV into a single PG $\boldsymbol{\mathcal{P}}_h$, such that the final output of the grid prediction module is a sequence $\{\boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_N\}$ of length N of unique PGs. The $\mathcal{T}$ predictions for each TV at step h are combined as follows:

$$p_{i,j,h} = 1 - \prod_{\tau \in \mathcal{T}} \left(1 - p_{i,j,h,\tau}\right), \tag{3.34}$$

effectively forming a single PG $\boldsymbol{\mathcal{P}}_h$ for step h.

3.3.5 Evaluation and Discussion

Training Results

We implement and train the neural network presented in Section 3.3.3 with the Deep Learning Toolbox™ in MATLAB®. The length of the initialization and prediction phases are selected as $M = 20$ and $N = 20$, whereas the sampling and prediction time T is $T = 0.2\,\mathrm{s}$. In total, the network produces 20 predictions with a long-term time horizon of $N \times T = 4\,\mathrm{s}$. Thus, each sequence sample has a length of $(M+N) \times T = 8\,\mathrm{s}$, as mentioned in Section 3.2.2.

However, each scenario includes a varying amount of vehicles. Since the network is not given information about the entire driving environment, as represented by the set $\mathcal{T}_{\mathrm{env}}$, but only the measurement tracks of each individual TV, as described by the measurement sequences $\mathcal{S}_\tau$, the data samples used for training are different as the training data sets introduced in Section 3.2.2. We additionally separate each of the time series measurements $\mathcal{S}_\tau$ for every TV into a different sample and consider only the TVs that are within the detection range of the EV for the entire length of the sequence sample, we obtain a total 481 sequence samples for individual object tracks, which are split into a training data set with 385 sequence samples, and validation and test data set with 48 sequences each.

After the training process, which ended after 870 iterations, we evaluate the prediction accuracy of the neural network measuring the Top-Ω Mean Absolute Error (MAE), as suggested in [PKK+18]. In the following, only the subscript h is used for simplicity. Given Ω network output candidates, where Ω must be less or equal to the beam width W ($\Omega \leq W$), we find the output $\boldsymbol{\mathcal{Y}}_h^*$ that best approximates the corresponding target label $\boldsymbol{\mathcal{Y}}_h$. Then, the Top-Ω MAE_h^Ω at a step h, together with the Top-Ω X-MAE_h^Ω and Y-MAE_h^Ω in the Cartesian x and y directions, is calculated as

$$\mathrm{MAE}_h^\Omega = \frac{1}{\Psi} \sum_{\psi=1}^{\Psi} \left\| \begin{bmatrix} x_{\psi,h}^* \\ y_{\psi,h}^* \end{bmatrix} - \begin{bmatrix} x_{\psi,h} \\ y_{\psi,h} \end{bmatrix} \right\|, \tag{3.35a}$$

$$\text{X-MAE}_h^\Omega = \frac{1}{\Psi} \sum_{\psi=1}^{\Psi} \left| x_{\psi,h}^* - x_{\psi,h} \right|, \tag{3.35b}$$

$$\text{Y-MAE}_h^\Omega = \frac{1}{\Psi} \sum_{\psi=1}^{\Psi} \left| y_{\psi,h}^* - y_{\psi,h} \right|, \tag{3.35c}$$

where Ψ is the number of samples in the test data set, $|\cdot|$ denotes the operator that returns the absolute value, $x_{\psi,h}^*$ and $y_{\psi,h}^*$ are the grid cell coordinates of the grid cell encoded in $\boldsymbol{\mathcal{Y}}_h^*$, and $x_{\psi,h}$ and $y_{\psi,h}$ are the analogous coordinates in the ground truth $\boldsymbol{\mathcal{Y}}_h$. Similarly, we also compute the general Top-Ω standard deviation σ_h^Ω of the

predictions and the Top-Ω standard deviations $\sigma^{\Omega}_{x,h}$ and $\sigma^{\Omega}_{y,h}$ in x and y directions as

$$\sigma^{\Omega}_{h} = \sqrt{\frac{1}{\Psi-1}\sum_{\psi=1}^{\Psi}\left\|\begin{bmatrix}x^{*}_{\psi,h}\\ y^{*}_{\psi,h}\end{bmatrix} - \begin{bmatrix}x_{\psi,h}\\ y_{\psi,h}\end{bmatrix}\right\|^{2}}, \tag{3.36a}$$

$$\sigma^{\Omega}_{x,h} = \sqrt{\frac{1}{\Psi-1}\sum_{\psi=1}^{\Psi}\left|x^{*}_{\psi,h} - x_{\psi,h}\right|^{2}}, \tag{3.36b}$$

$$\sigma^{\Omega}_{y,h} = \sqrt{\frac{1}{\Psi-1}\sum_{\psi=1}^{\Psi}\left|y^{*}_{\psi,h} - y_{\psi,h}\right|^{2}}. \tag{3.36c}$$

This way, the error measures $\sigma^{\Omega}_{x,h}$ and $\sigma^{\Omega}_{y,h}$ can be interpreted as the expected deviation of the predictions from the targets and are suitable candidates to model the standard deviation of the probability density function used in Section 3.3.4 as follows:

$$\sigma_x = \sigma^{\Omega}_{x,h}/a_x, \qquad \sigma_y = \sigma^{\Omega}_{y,h}/a_y, \tag{3.37}$$

where we divide by the respective grid cell lengths a_x and a_y to return the values in the format of grid cell indices. The Top-Ω MAE performance values from (3.35) and Top-Ω standard deviations with $\Omega = 3$, measured for prediction steps $h = 1, \ldots, 20$ on all the samples in the test data set, can be found in Table 3.1.

Discussion

In Section 3.3, we have studied a multi-class classification LSTM network to produce multiple steps of forecasts about the future motion of a dynamic object in a traffic environment with a long-term prediction horizon. Subsequently, we proposed an approach to derive a probabilistic map of the environment that can be utilized in the grid-based SMPC method at a later stage.
After evaluating the results on a test data set obtained from different simulations, we found that the model is able to accurately forecast the future position of a TV, but the prediction accuracy rapidly decays over time, especially between prediction steps $h = 1$ and $h = 2$. While a performance degradation over the time is expected and common in similar forecasting tasks, there are a number of factors that contribute to this phenomenon. In the first place, the approximative mapping Proj_2, which estimates the missing variables needed for the network input given the lower dimensional prediction output, introduces new uncertainty to the model artificially. Besides, the data samples on which the model operates include measurements that are relative to the EV, whose motion is unknown to the classification predictor. Moreover, no contextual information about the surrounding environment is taken into account in the evidence used for inference, so that the model cannot possibly be situationally aware and the interactions between the different agents on the road. Other aspects

Table 3.1: Top-Ω MAE_h^Ω, X-MAE_h^Ω, and Y-MAE_h^Ω, together with the standard deviations σ_h^Ω, $\sigma_{x,h}^\Omega$, and $\sigma_{y,h}^\Omega$ of the predictions for prediction steps $h = 1, \ldots, N$ with $N = 20$. Measured on all samples in the test data set.

Averaged MAE																				
h	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
MAE_h^Ω	1.17	8.07	10.07	11.65	12.78	13.42	14.28	14.86	16.09	17.15	17.97	18.86	19.09	21.21	23.01	24.28	25.57	27.08	27.57	28.18
σ_h^Ω	2.27	10.34	14.55	15.28	16.95	17.63	18.40	18.86	19.63	20.54	21.12	21.80	22.00	24.13	26.31	27.74	29.14	31.07	31.60	32.51
X-MAE_h^Ω	0.45	6.87	8.64	10.45	11.63	12.47	13.43	14.02	15.51	16.60	17.46	18.44	18.67	20.85	22.68	23.99	25.31	26.85	27.36	27.98
$\sigma_{x,h}^\Omega$	0.69	9.40	13.93	14.66	16.37	17.11	17.88	18.33	19.12	20.04	20.65	21.37	21.58	23.75	25.96	27.42	28.85	30.81	31.36	32.30
Y-MAE_h^Ω	0.79	2.91	2.64	2.84	2.97	2.85	2.91	3.06	3.06	3.09	3.05	3.01	2.97	2.96	2.87	2.84	2.78	2.78	2.64	2.49
$\sigma_{y,h}^\Omega$	2.16	4.42	4.19	4.32	4.38	4.26	4.32	4.44	4.45	4.49	4.42	4.34	4.32	4.28	4.21	4.16	4.05	4.03	3.88	3.75

that may negatively affect the profitability of the developed model are the inability to predict TVs entering or leaving the detection range and an eventually high computational cost when the beam width W is set to high values.
Nevertheless, considering the prediction of the TV motion as a classification task still offers some advantages that make the presented approach attractive, especially for the later application in the grid-based SMPC trajectory planning scheme. For instance, the probability distribution at the network output allows to account for the probabilistic nature of driving environments and can directly be used to extract a PG afterwards. In addition, the predictor is able to produce multimodal forecasts thanks to the beam search algorithm, where the beam width W can be adjusted depending on the available computational resources. Also, the error measure used to model the uncertainty in the TV motion is a good estimate of the deviation from the target values. Lastly, the fact that the network outputs are individualized for each TV gives a high degree of control over the PG, as different probabilistic models or variance measures could be employed.

3.4 Multi-Output Regression

Contrary to the classification model from Section 3.3, which produces a series of probabilistic outputs at every time step h that need to be aggregated thereafter, in this section, we develop a simpler regression grid predictor that only requires one forward pass through the network at every prediction step h and thus less intermediate steps afterwards. On the one hand, the regression system aims to reduce the computational cost and provide a faster prediction framework, but on the other hand, it comes at the expense of rejecting the probabilistic and multimodal nature of traffic environments and assuming a deterministic universe with a unique outcome for every situation. First, the regression problem addressed with the prediction model is introduced in Section 3.4.1. Next, we present the network architecture and the training strategy in Section 3.4.2, followed by the approach to derive a probabilistic map from the prediction output in Section 3.4.3. The chapter closes in Section 3.4.4 with the evaluation of the model and a discussion of the measured results.

3.4.1 Problem Definition

General Objective

In a neural regression problem, the objective is to learn the relationships between the input and output variables to approximate the underlying mapping that connects the inputs to the outputs. In general terms, we want to estimate an unknown function f that produces the next prediction $\boldsymbol{\mathcal{Y}}_h^*$, given the observation sequence $\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M\}$ of length M and the previous single-step forecasts $\left\{\boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-1}^*\right\}$ until step $h-1$ with $h = 1, \ldots, N$ as

$$\boldsymbol{\mathcal{Y}}_h^* = f\left(\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M, \boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-1}^*\right), \tag{3.38}$$

where the same function f is used for all prediction steps h and the inputs and outputs live in the same domain. Similarly to the discussion in Section 3.3.1, the sequence $\left\{\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M, \boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-2}^*\right\}$ of events that have been given to the network as past evidence until prediction stamp $h-2$ is in fact encoded into the LSTM internal states $\boldsymbol{m}_{h-1}$ and $\boldsymbol{h}_{h-1}$ used in the prediction phase. Hence, the mapping f is approximated by the neural network as a function NN, given by

$$f\left(\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_M, \boldsymbol{\mathcal{Y}}_1^*, \ldots, \boldsymbol{\mathcal{Y}}_{h-1}^*\right) \approx \mathrm{NN}\left(\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}_{h-1}^*\right), \tag{3.39}$$

whereas the network structure that forms the mapping NN needs to minimize the prediction error. Therefore, the neural architecture must possess sufficient parameters and non-linearities, which are introduced by the activation functions at each layer, to model the function f adequately as in (3.39). Then, we can estimate the single-step model output $\boldsymbol{\mathcal{Y}}_h^*$ at every prediction step h in the following way:

$$\boldsymbol{\mathcal{Y}}_h^* \approx \mathrm{NN}\left(\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}_{h-1}^*\right). \tag{3.40}$$

Input and Output

Furthermore, we address the multi-step predictive task with a multivariate multi-output time series regression with the intention of yielding a deterministic estimate of the entire picture of the scenario at the network output $\boldsymbol{\mathcal{Y}}_h^*$. Thereby, all dynamic objects on the grid are modeled jointly in a shared data structure, which is employed both for the inputs $\boldsymbol{\mathcal{X}}_k$ and the labels $\boldsymbol{\mathcal{Y}}_k$ used for supervised training. The object tracks kept in the set $\mathcal{T}_{\text{env}}$ from (3.5) are mapped onto a binary OG of values in $\{0, 1\}$, where the cells that are occupied according the oriented box model of any of the TVs $\tau \in \mathcal{T}$ within the detection range are assigned a one on the grid, and the remaining cells are filled with zeros.
The observation samples $\boldsymbol{\mathcal{X}}_m$ during the initialization phase include the EV in the OG with the environment evidence. Hence, during the prediction phase, the box model of the EV is introduced to the previous prediction $\boldsymbol{\mathcal{Y}}_{h-1}^*$ before executing (3.1). For the future EV motion, we use the SMPC trajectory $\boldsymbol{\xi}_{k-1,h}^{\text{EV}}$ computed at time step $k-1$ for predictions $h = 2, \ldots, N$.
Since the data format is an OG of binary values, the network output has to be restricted to the interval $[0, 1]^\Lambda$, which can be achieved by applying a clipped Rectified Linear Unit (ReLU) $\sigma_{\text{ReLU}}(\boldsymbol{x}) : \mathbb{R}^\Lambda \to [0, 1]^\Lambda$ to the vector of activations $\boldsymbol{x} = [x_1, \ldots, x_\Lambda]^\top \in \mathbb{R}^\Lambda$ of the last layer, where we define the element-wise clipped ReLU as

$$\sigma_{\text{ReLU}}(\boldsymbol{x})_\lambda = \begin{cases} 0, & \text{if } x_\lambda < 0, \\ x_\lambda, & \text{if } 0 \leq x_\lambda < 1, \\ 1, & \text{if } x_\lambda \geq 1. \end{cases} \tag{3.41}$$

Then, the regression function NN that estimates $\boldsymbol{\mathcal{Y}}_h^*$ as in (3.40), is given by

$$\mathrm{NN}\left(\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}_{h-1}^*\right) = \sigma_{\text{ReLU}}\left(\boldsymbol{m}_{h-1}, \boldsymbol{h}_{h-1}, \boldsymbol{\mathcal{Y}}_{h-1}^*\right). \tag{3.42}$$

3.4.2 Network Architecture and Training Procedure

Network Architecture

The mapping from (3.40) that aims to model complex traffic scenarios requires to recognize deep and non-linear dependencies between the input and output variables. Therefore, the regression network benefits from a deep architecture where every additional layer introduces new non-linearities with its respective activation function.

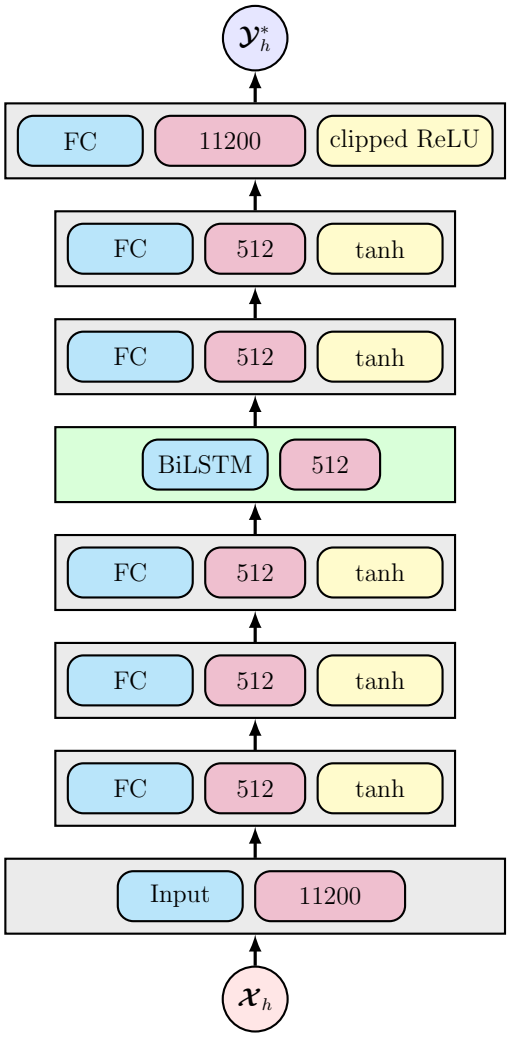


Figure 3.5: Architecture of the BiLSTM network used to predict the future of dynamic traffic scenarios as a regression task. Both the input $\boldsymbol{\mathcal{X}}_h$ and the output $\boldsymbol{\mathcal{Y}}_h^*$ at prediction step h are each a vector that contains 11200 variables. Every layer is represented as a container with at least two elements: the layer type in cyan, the output size in purple, and optionally the activation function in yellow. The width of the container varies depending on the number of units on the layer and the activations are propagated between the layers, as depicted by the vertical arrows.

The neural network consists of a series of stacked FC and LSTM layers. Data is given to the model via an input layer of size 11200, which is the amount of grid cells in the later evaluation setup. In this way, the value at every cell of the input OG is treated as an independent variable. The input layer is followed by three consecutive FC layers, each with 512 activations and the hyperbolic tangent tanh as activation function. These layers compress the input data into a low dimensional feature vector and extract patterns in the sample. Next, instead of the standard LSTM cell, we use a Bidirectional LSTM (BiLSTM) of size 512 at the network core, as suggested in [Neu17]. The BiLSTM is a common variant of the LSTM for sequential learning tasks that allows the recurrent network to learn dependencies from the entire sequence at every time step. In essence, a BiLSTM cell consists of two LSTMs of the same size, one that processes the input data in the forward direction, and another that processes the data in the backwards direction. In turn, BiLSTMs dispose of more temporal context compared to the LSTM and can improve the prediction capabilities in RNN networks. Due to the internal structure of the BiLSTM, the cell memory $\boldsymbol{m}_h \in \mathbb{R}^{1024}$ and the hidden state $\boldsymbol{h}_h \in \mathbb{R}^{1024}$ at a step h have double the size of the internal states of a standard LSTM. Hereafter, two further FC layers follow the BiLSTM, again with 512 activations each and the hyperbolic tangent tanh activation function. A final FC layer produces 11200 outputs, each corresponding to the predicted value at every cell from the output OG. Because the OGs employed during training only contain binary values, the outputs of the network are restricted to the interval $[0, 1]$ with the clipped ReLU activation function from (3.41). It is not of interest to directly classify the output values into binary numbers, as the values in the interval $(0, 1)$ can be used later to derive a probabilistic model of the future environment. The general architecture of the regression model is depicted in Figure 3.5.

Training Procedure

Just like the training procedure used for the classification network, the regression network also uses the Adam [KB14] solver. In this case, the initial learning rate is set to 0.0015 and multiplied with 0.2 every 125 epochs. The mini-batch size is 64 and the training process is concluded whenever the validation accuracy, measured every 10 iterations, stops to improve for 5 consecutive times.
As the loss function, the half-mean-squared-error loss L_{regr} for regression tasks from the Deep Learning Toolbox™ is employed, which is given by

$$L_{\text{regr}} = \frac{1}{2M^{\text{train}}} \sum_{m=1}^{M^{\text{train}}} \sum_{r=1}^{R} (\boldsymbol{y}_{m,r} - \boldsymbol{y}^*_{m,r})^2, \tag{3.43}$$

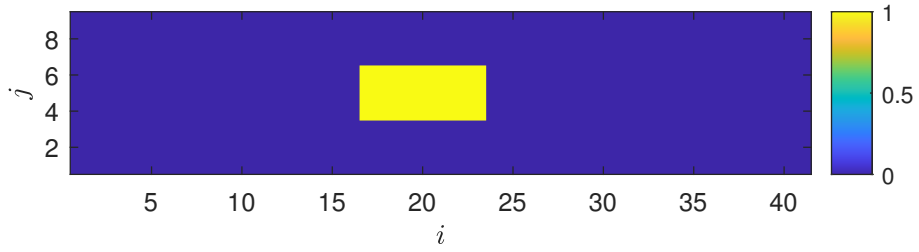
with the observation sequence length M^{train} during training, as defined in (3.22), the number of output responses R (in this case, $R = 11200$), and the r-th responses of both the ground truth label $\boldsymbol{y}_{m,r}$ and the prediction output $\boldsymbol{y}^*_{m,r}$ at observation time step m.

The exact network structure, as well as the network and training hyperparameters, have been specified with the same automated thorough sweep approach as for the classification model.

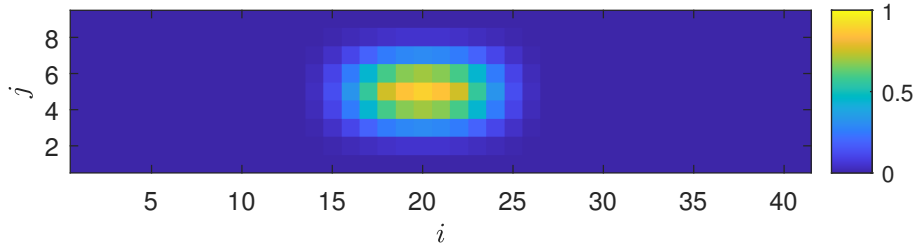
3.4.3 Probability Grid Derivation

A simplified version of the approach to derive a PG from the categorical distribution in Section 3.3.4 can be re-used to introduce a probabilistic prediction uncertainty to the deterministic outputs of the regression model. In this discussion, we consider the deterministic network output $\boldsymbol{\mathcal{Y}}_h^*$ as a PG candidate $\boldsymbol{\mathcal{P}}_h$ at step h:

$$\boldsymbol{\mathcal{P}}_h = \boldsymbol{\mathcal{Y}}_h^*. \tag{3.44}$$



(a) Artificial regression output $\boldsymbol{\mathcal{Y}}_h^*$, reshaped as a grid map. The grid cells occupied by a TV hold a value close to 1, unoccupied cells are assigned a value close to 0.



(b) Gaussian filtering to model prediction uncertainty. The variance measures $\sigma_x^2 = 1$ and $\sigma_y^2 = 1.5$ are used exemplarily.

Figure 3.6: The qualitative effect of the Gaussian filtering on a regression output.

Since $\boldsymbol{\mathcal{Y}}_h^*$ already models the shapes of the objects on the grid, there is no need to apply an additional box linear filter as in Section 3.3.4. Neither is it necessary to aggregate different sequences for different prediction hypotheses of each TV, as $\boldsymbol{\mathcal{Y}}_h^*$ already models all traffic participants (excluding the EV) jointly. It is sufficient to only apply the probabilistic convolutional filter to introduce the prediction uncertainty to the PG. Any arbitrary probability distribution can be used to model the prediction uncertainty. Again, we propose the method with a Gaussian filter defined analogously to (3.30), taking into regard (3.31) and (3.32) as well. The standard deviations σ_x and σ_y can also be chosen as an arbitrary prediction error measure, but a different one as in the classification task due to the different format of the network output.

3.4.4 Evaluation and Discussion

Training Results

We implement and train the LSTM network as introduced in Section 3.4.2 with the Deep Learning Toolbox™ in MATLAB®. The initialization and prediction lengths are again chosen as $M = 20$ and $N = 20$ with the sampling and prediction time $T = 0.2\,\mathrm{s}$. The training data is obtained as described in Section 3.2.2.
After the training process, which concluded after 80 iterations, we evaluate the prediction accuracy of the neural network measuring the Root-Mean-Squared Error (RMSE) between the outputs and the targets. In the following, we only use the subscript h for simplicity. Given a network output $\boldsymbol{\mathcal{Y}}_h^*$ and the corresponding ground truth label $\boldsymbol{\mathcal{Y}}_h$, the RMSE_h at a step h is given by

$$\mathrm{RMSE}_h = \sqrt{\frac{1}{R}\sum_{r=1}^{R}\left(y_{h,r}^* - y_{h,r}\right)^2}, \tag{3.45}$$

where $R = 11200$ is the number of output responses and cells on the grid, and $y_{h,r}^* \in \boldsymbol{\mathcal{Y}}_h^*$ and $y_{h,r} \in \boldsymbol{\mathcal{Y}}_h$ are the r-th elements of $\boldsymbol{\mathcal{Y}}_h^*$ and $\boldsymbol{\mathcal{Y}}_h$ respectively. We additionally define the X-RMSE_h and Y-RMSE_h in x and y directions, where we use the sums of the square errors along the columns and rows respectively

$$\text{X-RMSE}_h = \sqrt{\frac{1}{n_{\mathrm{row}}}\sum_{i=1}^{n_{\mathrm{row}}}\sum_{j=1}^{n_{\mathrm{col}}}\left(y_{h,i,j}^* - y_{h,i,j}\right)^2}, \tag{3.46a}$$

$$\text{Y-RMSE}_h = \sqrt{\frac{1}{n_{\mathrm{col}}}\sum_{i=1}^{n_{\mathrm{row}}}\sum_{j=1}^{n_{\mathrm{col}}}\left(y_{h,i,j}^* - y_{h,i,j}\right)^2}, \tag{3.46b}$$

where $n_{\mathrm{row}} = 400$ and $n_{\mathrm{col}} = 28$ are abbreviations for the number of rows and columns in the grid $\boldsymbol{\mathcal{G}}$, and $y_{h,i,j}^* \in \boldsymbol{\mathcal{Y}}_h^*$ and $y_{h,i,j} \in \boldsymbol{\mathcal{Y}}_h$ refer to the i-th row and j-th column in $\boldsymbol{\mathcal{Y}}_h^*$ and $\boldsymbol{\mathcal{Y}}_h$, reshaped as matrices with the shape of $\boldsymbol{\mathcal{G}}$. As the presented RMSE measures are only computed for each pair formed by an output $\boldsymbol{\mathcal{Y}}_h^*$ and the corresponding target $\boldsymbol{\mathcal{Y}}_h$ at a step h, we measure the mean and standard deviation of the different RMSE measures on all samples in the test data set for every prediction step $h = 1, \ldots, N$, which can be found in Table 3.2.
Even though they do not describe the deviation in the predicted TV motion as the error measures used for the classification model, the mean values of X-RMSE_h and Y-RMSE_h still offer a measure of the expected amount of deviation of the predicted values in the OG from the target values along each spatial dimension. Therefore, we directly use the mean values of the X-RMSE_h and the Y-RMSE_h as standard deviations σ_x and σ_y for the probability distribution used in the blurring filter that models uncertainty in the PG.

Table 3.2: Mean values of the RMSE, X-RMSE$_h$, and Y-RMSE$_h$, together with the respective standard deviations σ_h^{RMSE}, $\sigma_h^{\text{X-RMSE}}$, and $\sigma_h^{\text{Y-RMSE}}$ of the predictions for prediction steps $h = 1, \ldots, N$ with $N = 20$. Measured on all samples in the test data set.

	Averaged RMSE																			
h	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RMSE$_h$	0.09	0.11	0.11	0.12	0.13	0.13	0.13	0.14	0.14	0.15	0.15	0.15	0.15	0.16	0.16	0.16	0.16	0.16	0.16	0.17
σ_h^{RMSE}	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.09	0.09	0.09	0.09	0.09	0.09	0.09
X-RMSE$_h$	1.89	2.12	2.27	2.38	2.53	2.61	2.68	2.76	2.82	2.87	2.91	2.94	2.97	3.00	3.06	3.09	3.10	3.13	3.14	3.16
$\sigma_h^{\text{X-RMSE}}$	1.20	1.33	1.41	1.47	1.49	1.51	1.55	1.60	1.65	1.68	1.68	1.68	1.70	1.70	1.70	1.70	1.70	1.71	1.71	1.72
Y-RMSE$_h$	0.49	0.56	0.60	0.63	0.67	0.69	0.71	0.73	0.75	0.76	0.77	0.78	0.79	0.80	0.81	0.82	0.82	0.83	0.84	0.85
$\sigma_h^{\text{Y-RMSE}}$	0.32	0.35	0.37	0.39	0.39	0.40	0.41	0.42	0.44	0.44	0.45	0.45	0.45	0.45	0.45	0.45	0.46	0.46	0.46	0.46

Discussion

In Section 3.4, we have presented a multi-output regression LSTM network that generates multiple steps of forecasts in the format of an OG with a long-term prediction horizon. The approach to find a probabilistic map of the environment, previously introduced for the classification model, can also be applied to the regression output for the later utilization in the grid-based SMPC strategy for automated driving.
The main disadvantage of the prediction model is that the probabilistic nature of the road environment is not regarded and a universe with deterministic outcomes is assumed. Thus, the network is unable to produce multimodal outputs. Besides, the prediction error increases over the time, not only due to the error in the forecasted motion of the TVs, but also because the shapes of the vehicles become less exact and appear with blurred edges. Additionally, prediction uncertainty is propagated through the prediction steps, where noise in previous forecasts may lead to new objects appearing on the OG. Moreover, the error measure chosen to evaluate the deviation of the predictions in each spatial dimension does not describe accurately the uncertainty in the predicted TV motion, as it is independent of the number of vehicles in the environment. Also, we have a low degree of control during the derivation of the PG from the prediction output, since the probabilistic uncertainty can only be modeled for the entire environment, and not for each TV individually.
Despite the exposed shortcomings of the regression model, the presented predictor exhibits a number of significant strengths that can complement the classification model. First, the regression network is given information about all the dynamic objects within the detection range, which enables the model to understand vehicle interaction and group behavior patterns. Furthermore, the predictor can be trained to forecast TVs entering or leaving the space in the local environment of the EV. On another note, the prediction model only requires one forward pass through the network at every prediction step, independently of the number of objects on the grid, which makes the computational cost more consistent.

Chapter 4

Grid-Based Stochastic Model Predictive Control

In this chapter, the grid-based SMPC method for trajectory planning of autonomous vehicles is described. Section 4.1 introduces the system model and constraint sets for the EV, Section 4.2 presents the process to derive a deterministic reformulation of the chance constraint based on the PGs extracted in the previous chapter. Next, Section 4.3 we evaluate the overall method and discuss the results. To improve readability, only one of the subscripts k (referring to the sampling step) or h (designating the prediction step) is given where needed, as long as the other is not relevant to the discussion.

4.1 System Models

Modeling the EV as a point of mass at its center of gravity, a linear, discrete-time double integrator dynamic system model can be obtained:

$$\boldsymbol{\xi}_{k+1}^{\mathrm{EV}} = \boldsymbol{A}\boldsymbol{\xi}_{k}^{\mathrm{EV}} + \boldsymbol{B}\boldsymbol{u}_k, \tag{4.1}$$

where the state and control vectors $\boldsymbol{\xi}_k^{\mathrm{EV}}$ and $\boldsymbol{u}_k$ read as

$$\boldsymbol{\xi}_k^{\mathrm{EV}} = \left[x_k^{\mathrm{EV}}, v_{x,k}^{\mathrm{EV}}, y_k^{\mathrm{EV}}, v_{y,k}^{\mathrm{EV}}\right]^\top, \qquad \boldsymbol{u}_k = \left[a_{x,k}, a_{y,k}\right]^\top, \tag{4.2}$$

with x_k^{EV} and y_k^{EV} as the x and y positions at time step k, and $v_{x,k}^{\mathrm{EV}}$ and $v_{y,k}^{\mathrm{EV}}$ together with $a_{x,k}$ and $a_{y,k}$ as the velocity and acceleration in x and y direction respectively. The system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix}, \tag{4.3}$$

where T is the sampling time. In contrast to the general SMPC definition, no system disturbance is considered in the EV system model, because uncertainties occur mostly in the surrounding environment.
Both the states and the inputs need to satisfy constraints at all time steps. These constraints are

$$\boldsymbol{\xi}_k \in \left(\Xi_k^{\text{road}} \cap \Xi_k^{\text{safe}}\right), \qquad \boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max}, \tag{4.4}$$

where Ξ^{road} includes road and physical limitations, and Ξ^{safe} regards other vehicles to avoid accidents. The restrictions applied to $\boldsymbol{u}_k$ are bound to the vehicle specifications. In short, the constraints in (4.4) are encapsulated in

$$\boldsymbol{\xi}_k \in \Xi_k, \qquad \boldsymbol{u}_k \in \mathcal{U}_k. \tag{4.5}$$

4.2 Modified Stochastic Model Predictive Control

In this section, we provide an overview about the routine developed in [BDP+20] to obtain a deterministic constraint set from the chance constraints to reduce the computational complexity of the SMPC framework. Here, we assume an already given PG, as described in Sections 3.3.4 and 3.4.3.

4.2.1 Chance Constraint Reformulation

After the computation of the PG to stochastically model the territory around the EV at prediction step h, the SMPC chance constraint can be reformulated as a linear constraint. We contemplate the following chance constraint on the set of states Ξ_h^{safe}, which was introduced in (4.4) to avoid impacts with road obstacles:

$$\Pr\left(\boldsymbol{\xi}_h^{\text{EV}} \in \Xi_h^{\text{safe}}\right). \tag{4.6}$$

First, a threshold p_{th} is applied to the grid $\boldsymbol{\mathcal{P}}$ to turn it into a Binary Grid (BG), represented by $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{\frac{\iota_x}{a_x} \times \frac{\iota_y}{a_y}}$, of binary values

$$b_{i,j} = \begin{cases} 1, & \text{if } p_{i,j} \geq p_{\text{th}}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

The threshold parameter p_{th} is tunable and allows to adjust the risk aversion of the controller: similarly to the probability level β, the control behavior becomes more conservative the higher the risk factor p_{th}. In essence, the BG divides the space into two mutually exclusive and collectively exhaustive subsets of inadmissible or occupied cells, where $b_{i,j} = 1$, and admissible or free cells, which all hold the value 0. This way, the chance constraint (4.6) can be transformed into a hard state constraint

$$\boldsymbol{\xi}_h^{\text{EV}} \in \Xi_h^{\text{adm}}, \tag{4.8}$$

where Ξ_h^{adm} contains all admissible cells in $\boldsymbol{\mathcal{B}}$ and symbolizes the admissible space. Next, a linear description of (4.8) can be found to further improve the performance of the SMPC scheme. In this sense, a method that relies on Bresenham's line algorithm is proposed in [BDP+20] to find the maximal convex subset within the admissible space. Consequently, the convex space extracted from Ξ^{adm} can be expressed with a linear inequality, given by

$$\boldsymbol{A}_h^{\text{safe}} \boldsymbol{\xi}_h^{\text{EV}} \leq \boldsymbol{b}_h^{\text{safe}}, \tag{4.9}$$

with $\boldsymbol{A}_h^{\text{safe}} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{b}_h^{\text{safe}} \in \mathbb{R}^4$. The resulting constraint is tractable and reduces the computational cost of the SMPC trajectory planning method significantly, as the described procedure is repeated at every prediction step h.

4.2.2 Modified Optimal Control Problem

As an outcome of the previous steps, the SMPC optimal control problem is transformed to a standard MPC problem without chance constraints:

$$\boldsymbol{U}^* = \arg\min_{\boldsymbol{U}} \sum_{h=0}^{N-1} \left(\left\| \boldsymbol{\Delta\xi}_h^{\text{EV}} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{u}_h^{\text{EV}} \right\|_{\boldsymbol{R}}^2 \right) + \left\| \boldsymbol{\Delta\xi}_N^{\text{EV}} \right\|_{\boldsymbol{S}}^2 \tag{4.10a}$$

$$\text{s.t.} \quad \boldsymbol{\xi}_0^{\text{EV}} = \hat{\boldsymbol{\xi}}_k^{\text{EV}}, \tag{4.10b}$$

$$\boldsymbol{\xi}_{h+1}^{\text{EV}} = \boldsymbol{A}\boldsymbol{\xi}_h^{\text{EV}} + \boldsymbol{B}\boldsymbol{u}_h^{\text{EV}}, h \in \mathbb{N}, \tag{4.10c}$$

$$\boldsymbol{\xi}_0^{\text{TV}} = \hat{\boldsymbol{\xi}}_k^{\text{TV}}, \tag{4.10d}$$

$$\boldsymbol{u}_h^{\text{EV}} \in \mathcal{U}_h, \qquad h = 0, \ldots, N-1, \tag{4.10e}$$

$$\boldsymbol{\xi}_h^{\text{EV}} \in \Xi_h, \qquad h = 1, \ldots, N, \tag{4.10f}$$

$$\boldsymbol{A}_h^{\text{safe}} \boldsymbol{\xi}_h^{\text{EV}} \leq \boldsymbol{b}_h^{\text{safe}}, \qquad h = 1, \ldots, N, \tag{4.10g}$$

with the optimal control sequence $\boldsymbol{U}^* = \left[\boldsymbol{u}_0^*, \ldots, \boldsymbol{u}_{N-1}^*\right]$, which minimizes the cost function over the control input $\boldsymbol{U} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}]$, the operator $\|\boldsymbol{z}\|_{\boldsymbol{W}}^2 = \boldsymbol{z}^\top \boldsymbol{W} \boldsymbol{z}$ as the squared norm of $\boldsymbol{z}$ w.r.t. metric $\boldsymbol{W}$, the difference $\boldsymbol{\Delta\xi}_h^{\text{EV}} = \boldsymbol{\xi}_h^{\text{EV}} - \boldsymbol{\xi}_{h,\text{ref}}^{\text{EV}}$ and the EV reference $\boldsymbol{\xi}_{h,\text{ref}}^{\text{EV}}$, the weighing matrices $\boldsymbol{Q}, \boldsymbol{S} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{R} \in \mathbb{R}^{2\times2}$, the state estimations $\hat{\boldsymbol{\xi}}_k^{\text{EV}}$ and $\hat{\boldsymbol{\xi}}_k^{\text{TV}}$ at sampling step k, the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ given by (4.3), and the constraint sets $\mathcal{U}_h$ and Ξ_h according to (4.5). The TV prediction model, which can be chosen either as the classification or the regression model, is implicitly contained in constraint (4.10g), which is derived from state set Ξ_h^{safe} for collision avoidance.

4.3 Evaluation

In this section, we evaluate the grid-based SMPC algorithm for trajectory planning in autonomous driving with the extended data-driven grid prediction. First, a simulation framework is presented in Section 4.3.1. Next, some qualitative simulation results are analyzed in Section 4.3.2 and the general approach is discussed in Section 4.3.3.

4.3.1 Simulation Framework

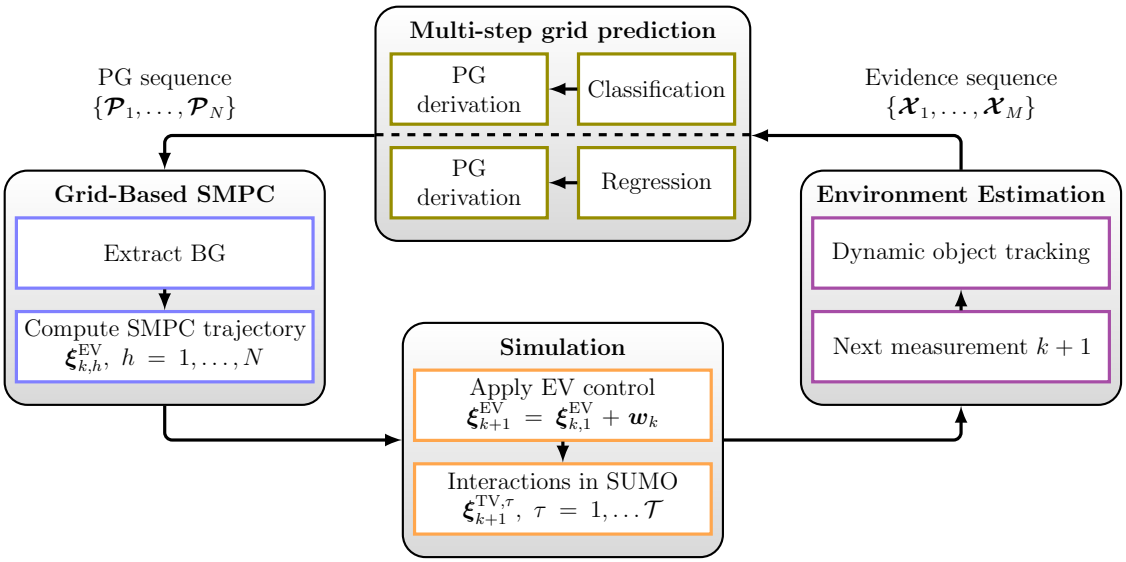


Figure 4.1: Simulation framework outline. Starting at the container at the bottom, a simulation of a traffic scenario is conducted in SUMO, where the EV is controlled externally. At each simulation step, a sequence of past observations is constructed in the environment estimation module (container to the right) from the information in the simulation. In the upper container, the multi-step grid prediction module produces a sequence of PGs either with a classification or a regression network. The PGs are then used in the grid-based SMPC (container to the left) to calculate the optimal trajectory and generate the optimal prediction sequence.

The proposed grid-based SMPC method with an integrated deep learning grid prediction module, which can either be executed as a classification or a regression network, is evaluated in a simulation framework developed in MATLAB ® and SUMO. A simulation of a road scenario is conducted in SUMO, where the interactions between the traffic participants can be evaluated step-wise at a microscopic level, meaning that the dynamics of every single vehicle are modeled individually [LBBW+18]. The EV is controlled externally from a MATLAB ® client, where SUMO is run as a server. The state $\boldsymbol{\xi}^{\mathrm{EV}}_{k+1}$ of the EV for the next simulation step $k+1$ is computed according to

$$\boldsymbol{\xi}^{\mathrm{EV}}_{k+1} = \boldsymbol{\xi}^{\mathrm{EV}}_{k,1} + \boldsymbol{w}_k, \tag{4.11}$$

where $\boldsymbol{\xi}^{\mathrm{EV}}_{k,1}$ is the first predicted state of the EV at simulation step k for prediction step $h = 1$ and $\boldsymbol{w}_k \in \mathbb{R}^4$, with $\boldsymbol{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$, is a normally distributed, zero-mean random variable with covariance matrix $\boldsymbol{\Sigma}_w$, which acts as a stochastic disturbance that introduces model uncertainty. The state $\boldsymbol{\xi}^{\mathrm{EV}}_{k,1}$ is the result of applying the first control action $\boldsymbol{u}^*_0$ from the optimal control sequence $\boldsymbol{U}^*$, as computed in (4.10a), to the EV system model from (4.1)

$$\boldsymbol{\xi}^{\mathrm{EV}}_{k,1} = \boldsymbol{A}\boldsymbol{\xi}^{\mathrm{EV}}_{k,0} + \boldsymbol{B}\boldsymbol{u}^{\mathrm{EV}}_{k}, \tag{4.12}$$

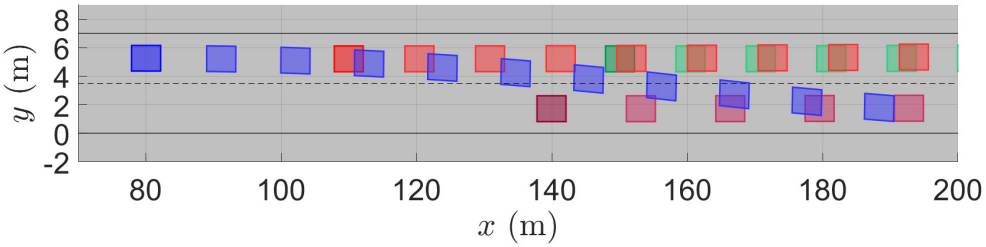
whereas $\boldsymbol{\xi}_{k,0}^{\mathrm{EV}}$ is given by (4.10b). Then, the EV state $\boldsymbol{\xi}_{k+1}^{\mathrm{EV}}$ for simulation step $k+1$ is updated in SUMO and the traffic simulation advances one step to simulation instance $k+1$. There, SUMO produces the states $\boldsymbol{\xi}_{k+1}^{\mathrm{TV},\tau}$ of all $\tau = 1, \dots \mathcal{T}$ TVs in the simulation scenario, simulating real-life driving behaviors and interactions.
Next, the measurement vectors $\boldsymbol{z}_{\tau,k}$ from (3.3) are constructed directly from the simulation data, imitating the environment estimation and dynamic object tracking approach from [STW17]. From the set $\mathcal{T}_{\mathrm{env}}$ that describes the local environment of the EV, as defined in (3.5), the observation sequence $\{\boldsymbol{\mathcal{X}}_1, \dots, \boldsymbol{\mathcal{X}}_M\}$ is constructed to be used as evidence both in the classification and in the regression prediction modules.
Subsequently, one of the two presented multi-step grid prediction methods is used to predict the future motion of the dynamic objects in the surroundings of the EV for prediction steps $h = 1, \dots, N$. The PG sequence $\{\boldsymbol{\mathcal{P}}_1, \dots, \boldsymbol{\mathcal{P}}_N\}$ is derived from the forecasts to be used in the grid-based SMPC trajectory planning scheme.
Based on the PGs $\boldsymbol{\mathcal{P}}_h$ at every step h, the chance constraints are reformulated with the help of the BG $\boldsymbol{\mathcal{B}}_h$ for efficient trajectory planning, yielding the predicted state $\boldsymbol{\xi}_{k,h}^{\mathrm{EV}}$ and the optimal control sequence $\boldsymbol{U}^*$.

4.3.2 Experiment

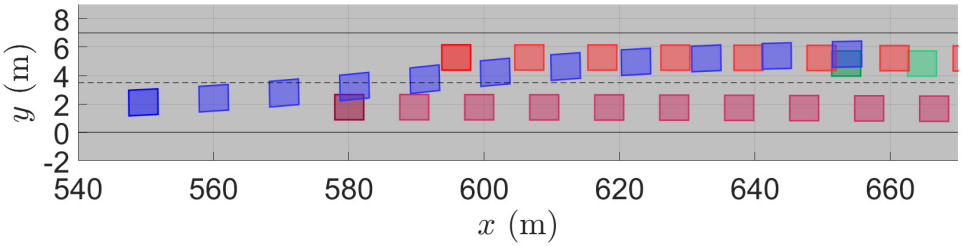
In this Section, we design a simulation of a driving scenario to qualitatively analyze the grid-based SMPC trajectory planner exemplarily with the regression prediction model from Section 3.4. As a probability threshold p_{th} to derive the BG, we use the value 0.1, which is a relatively low risk parameter, given that the values in the PG derived from the regression model are found in the interval $[0, 1]$. The simulated environment consists of the same two-lane highway as used for the data collection in Section 3.2.2.
The initial simulation setup is formed by an EV with the state $\boldsymbol{\xi}_0^{\mathrm{EV}}$ and three TVs with the states $\boldsymbol{\xi}_0^{\mathrm{TV},1}$, $\boldsymbol{\xi}_0^{\mathrm{TV},2}$, and $\boldsymbol{\xi}_0^{\mathrm{TV},3}$, which are

$$\boldsymbol{\xi}_0^{\mathrm{EV}} = [80\,\mathrm{m}, 28\,\mathrm{m/s}, 5.25\,\mathrm{m}, 0\,\mathrm{m/s}]^\top, \tag{4.13a}$$
$$\boldsymbol{\xi}_0^{\mathrm{TV},1} = [110\,\mathrm{m}, 27\,\mathrm{m/s}, 5.25\,\mathrm{m}, 0\,\mathrm{m/s}]^\top, \tag{4.13b}$$
$$\boldsymbol{\xi}_0^{\mathrm{TV},2} = [150\,\mathrm{m}, 28\,\mathrm{m/s}, 5.25\,\mathrm{m}, 0\,\mathrm{m/s}]^\top, \tag{4.13c}$$
$$\boldsymbol{\xi}_0^{\mathrm{TV},3} = [140\,\mathrm{m}, 24\,\mathrm{m/s}, 1.75\,\mathrm{m}, 0\,\mathrm{m/s}]^\top, \tag{4.13d}$$

at the initial sampling step $k = 0$. The lateral positions at 5.25 m and 1.75 m correspond to the centers of the left and right lanes respectively. The EV is given a reference velocity of 26 m/s. Here, we present the qualitative simulation results and visualize two fragments, which are depicted in Figure 4.2.
During the first simulation excerpt, which encompasses the time steps $k = 0, \dots, 20$ and is shown in Figure 4.2a, the EV performs a lane change from the left to the right lane. The EV is initially placed behind a TV (with index 1) with a gap of 30 m.



(a) Simulation fragment for time steps $k = 0, \ldots, 20$. The EV switches from the left to the right lane while the TVs stay on their initial lanes.



(b) Simulation fragment for time steps $k = 91, \ldots, 112$. The EV moves from the right to the left and starts an overtaking maneuver.

Figure 4.2: An EV (blue) interacts in a traffic scenario with three TVs (TV with index 1 is red, TV with index 2 is green, and TV with index 3 is purple). The starting positions are colored dark, and only every second simulation step is displayed in a lighter tone. Vehicles advance in positive x direction.

Therefore, the EV detects a possible collision within the prediction horizon with TV 1 and moves to the free lane, where obstacles are more distant.
In the time between the first and the second fragment shown, the TVs 1 and 2 overtake TV 3 and leave free space in the left lane. On the other side, the EV stays on the same lane behind TV 3 and the gap between the two vehicles narrows.
Then, during the second simulation section, which ranges from time steps $k = 91$ to $k = 112$ and is illustrated in Figure 4.2b, the EV is traveling at its reference velocity, which is faster than the velocity of the TV 3. Thus, the EV switches back to the left lane and starts an overtaking maneuver to pass the slower TV 3.

4.3.3 Discussion

In Chapter 4, we introduced the grid-based SMPC strategy for trajectory planning in automated driving. Then, we presented a simulation framework to test and evaluate the control achieved by the overall method using the grid prediction module proposed in Chapter 3. In a highway scenario with four total participants, we have demonstrated that the approach can work as expected with a low probability threshold p_{th} to avoid risk. The chosen threshold, which does not truly describe a real probability, causes a conservative behavior of the controller.

Chapter 5

Conclusion

In this work, we presented two data-driven grid prediction models to produce multiple steps of forecasts about the future motion of external traffic participants with a long-term horizon. Further, we proposed a general approach to process the prediction outputs into a PG that is employed in the grid-based SMPC scheme to perform efficient trajectory planning for an ego vehicle. The overall method is able to handle an increased number of dynamic objects on the road, as demonstrated in a simulation with four vehicles. Additionally, the future motion of external objects can be modeled with an arbitrary probability distribution to account for the stochastic conduct of road agents.
The first prediction model introduced in this work as a classification network produces a probability distribution over the discretized environment, describing the likelihood of the presence of a point-mass object. Furthermore, the model operates on every dynamic object individually and considers multiple possible outcomes. The second prediction model, which is a regression network, directly generates a forecast of the complete surrounding scene by considering a deterministic universe. Both of the proposed alternatives show strengths and shortcomings that can complement each other. Nevertheless, the prediction models still have potential to improve and increase the accuracy. Neural networks often benefit from deep architectures and convolutional layers to better capture spatial patterns and relashionships. Also, convolutional LSTMs improve the used FC-LSTMs in recognizing spatio-temporal dependencies. Moreover, the recurrent encoder-decoder structure has also shown an excellent performance in sequence-to-sequence tasks.
For future work, it is still of interest to evaluate the scalability of the proposed methodology in urban driving scenarios, where the interactions between the diverse agents in the environment play a more important role. In addition, it is crucial to ensure safety in automated driving, hence, measures to guarantee safety may be key in the future development of the framework. Also, the trajectory planning strategy could be tested in real-life situations and be extended to different applications of intelligent agents, such as robotics or navigation systems in varying environments, i.e., air and waterborne.

List of Figures

List of Tables

Acronyms

BG Binary Grid.
BiLSTM Bidirectional Long Short-Term Memory.

DOG Dynamic Occupancy Grid.

EV Ego Vehicle.

FC Fully Connected.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.
MPC Model Predictive Control.

OG Occupancy Grid.

PG Probabilty Grid.

ReLU Rectified Linear Unit.
RMSE Root-Mean-Squared Error.
RNN Recurrent Neural Network.

SMPC Stochastic Model Predictive Control.

TV Target Vehicle.

Bibliography

[BDP+20] Tim Brüdigam, Fulvio Di Luzio, Lucia Pallottino, Dirk Wollherr, and Marion Leibold. Grid-based stochastic model predictive control for trajectory planning in uncertain environments. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[CGLB14] Ashwin Carvalho, Yiqi Gao, Stephanie Lefevre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th International Symposium on Advanced Vehicle Control*, 2014.

[GP17] Lars Grüne and Jürgen Pannek. Nonlinear model predictive control. In *Nonlinear Model Predictive Control: Theory and Algorithms*, pages 45–69. Springer International Publishing, Cham, 2017.

[HBD17] Stefan Hörmann, Martin Bach, and Klaus Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. *Computing Research Repository (CoRR)*, 2017.

[Hö20] Florian Hölzl. Grid-based stochastic model predictive control with occupancy grids. Research Internship Report, Technical University of Munich, Germany, 2020.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[KDEA19] Moritz Klischat, Octav Dragoi, Mostafa Eissa, and Matthias Althoff. Coupling sumo with a motion planning framework for automated vehicles. In *SUMO User Conference*, EPiC Series in Computing, pages 1–9. EasyChair, 2019.

[LBBW+18] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic

traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*, pages 2575–2582. IEEE, November 2018.

[Mes16] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems*, 36(6):30–44, 2016.

[MR19] Nima Mohajerin and Mohsen Rohani. Multi-step prediction of occupancy grid maps with recurrent neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10592–10600, Los Alamitos, CA, USA, 2019. IEEE Computer Society.

[Neu17] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *Computing Research Repository (CoRR)*, 03 2017.

[PKK+18] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678, 2018.

[Sha76] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[SHD19] Marcel Schreiber, Stefan Hörmann, and Klaus Dietmayer. Long-term occupancy grid prediction using recurrent neural networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9299–9305. IEEE, 2019.

[She20] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404(8):132306, 2020.

[STW17] Sascha Steyer, Georg Tanzmeister, and Dirk Wollherr. Object tracking based on evidential dynamic occupancy grids in urban environments. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1064–1070. IEEE, 2017.

[TM21] Inc. The MathWorks. *MATLAB and Deep Learning Toolbox Release R2021b*. Natick, Massachusetts, United State, 2021.

License