arXiv:2208.01862v1 [cs.RO] 3 Aug 2022

MixNet: Structured Deep Neural Motion Prediction for Autonomous Racing

Phillip Karle[1]*, **Ferenc Török**[1], **Maximilian Geisslinger**[1], **Markus Lienkamp**
Institute of Automotive Technology (FTM)
Technical University of Munich
Garching, Germany

Abstract

Reliably predicting the motion of contestant vehicles surrounding an autonomous racecar is crucial for effective and performant planning. Although highly expressive, deep neural networks are black-box models, making their usage challenging in safety-critical applications, such as autonomous driving. In this paper, we introduce a structured way of forecasting the movement of opposing racecars with deep neural networks. The resulting set of possible output trajectories is constrained. Hence quality guarantees about the prediction can be given. We report the performance of the model by evaluating it together with an LSTM-based encoder-decoder architecture on data acquired from high-fidelity Hardware-in-the-Loop simulations. The proposed approach outperforms the baseline regarding the prediction accuracy but still fulfills the quality guarantees. Thus, a robust real-world application of the model is proven. The presented model was deployed on the racecar of the Technical University of Munich for the Indy Autonomous Challenge 2021. The code used in this research is available as open-source software at `www.github.com/TUMFTM/MixNet`.

1 Introduction

Developing reliable autonomous vehicles has various purposes, including safer, more relaxing, and more efficient traveling. To fuel innovation in the field, autonomous vehicle competitions such as the DARPA Grand Challenge (Buehler et al., 2007), Formula Student[2] and Roborace[3] have taken place. There is, however, an aspect of autonomous racing that has not been covered before: full-scale multi-vehicle racing against other competitors. The Indy Autonomous Challenge (IAC) and its successor, the Autonomous Challenge at the CES 2022[4] (AC@CES) were meant to take this enormous next step. In the competition, the teams were provided with the same hardware and developed their own autonomous software stacks. Wheel-to-wheel racing poses serious challenges, which are also present in everyday road traffic. To operate safely and efficiently in a dynamic environment, the vehicle has to predict the future motion of other dynamic objects. Usually, the methods tackling this problem take the motion histories of the objects together with some environmental information and make predictions conditioned on these. Several solutions use the current estimated state of the vehicle and extrapolate it using a physics-based model (Ammoun and Nashashibi, 2009; Xie et al., 2017; Deo et al., 2018; Schubert et al., 2008). Although these methods are appealing due to their

*corresponding author
[1]phillip.karle@tum.de, ferenc.toeroek@tum.de, maximilian.geisslinger@tum.de
[2]https://www.imeche.org/events/formula-student/team-information/fs-ai
[3]https://roborace.com/
[4]https://www.indyautonomouschallenge.com/

transparency and small computational demands, the predictions become less reliable in the long term. Other approaches, which we will cover in detail, use machine learning techniques to match the motions of the objects to motion or behavior patterns and forecast accordingly. These methods often provide long-term predictions up to 8 s and are more expressive. On the other hand, the approximators used are often black-box models and hence performance or quality guarantees are hard to give.

The prediction algorithm developed for the race has to fulfill some prerequisites: 1) it should output a single trajectory for each object; 2) the prediction horizon should be 5 s long; 3) the available time window for calculation is 20 ms with a single CPU core of computing capacity available; 4) it should be robust against noise in the inputs; 5) The output should always be smooth and should lie inside the racetrack. The available input information consists of the motion histories of the objects and the map of the racetrack. Our approach to fulfill these requirements is shown in Figure 1. We propose a structured deep neural motion forecasting model. The approach, although deep learning-based, creates trajectory prediction in a structured way. The object's past movement is encoded by Recurrent Neural Networks (RNNs) and transferred into a latent space. In contrast to common approaches, which determine the future trajectory in the decoder step entirely by deep neural network, MixNet calculates weighting parameters for superposition of dynamically feasible base curves derived from the track. By this, the MixNet is capable of providing strong quality guarantees for its predictions, which enables real-world application in a broad range of scenarios. An illustrative example is given in Figure 2. MixNet outputs a prediction based on the encoded scenario understanding with high accuracy together with a smooth trajectory shape. In comparison, the benchmark model, which has the same encoder architecture, i.e., incorporates the same scenario understanding, but determines the prediction by means of a LSTM-based decoder cannot guarantee a stable output, so especially on a long prediction horizon the predicted trajectory gets noisy. Besides that, our approach comprises the feature of fusing external velocity information and overriding the calculated velocity profile. By this, the model can be further constrained, which are essential feature for a safe real-world application and useful to apply the model to a new Operational Design Domain (ODD). Besides that, an implemented fuzzy logic that models overtaking maneuvers improves the interaction-awareness to resolve non-feasible colliding trajectory predictions.

The structure of the paper is as follows. In Section 2 we summarize the main research directions in the field of motion prediction for autonomous driving and discuss the applicability for our purpose regarding the aforementioned prerequisites. In Section 3 we introduce our approach of the MixNet and additional features that cover the velocity fusion, safety checks and interaction-awareness. Besides that, we present our recorded dataset and the training procedure. Then in Section 4 we evaluate its performance compared to the baseline model and analyze its robustness against noisy input and the superpositioning weights. In Section 5 we discuss future research directions based on the shown results. Finally, in Section 6, we conclude our work and outline the scope of the paper.

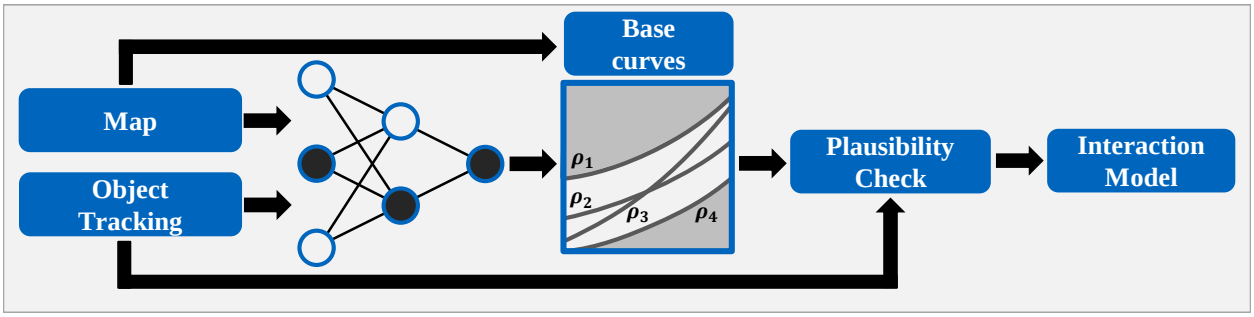


Figure 1: Overview of the MixNet prediction module. It combines a comprehensive, learned scenario understanding by means of an RNN-encoder and semantic knowledge to constrain the output by base curves extracted from the track map.

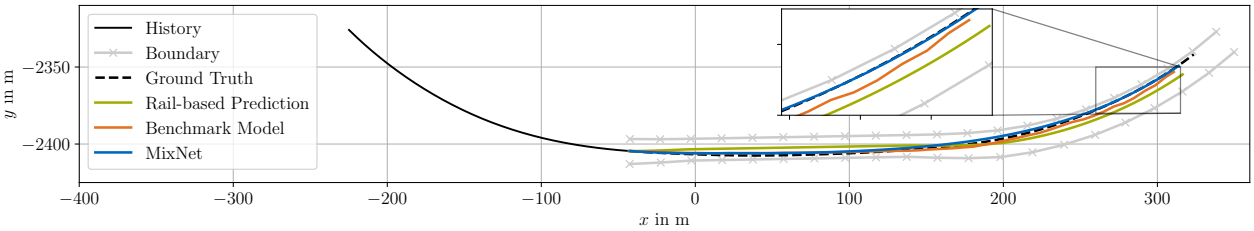


Figure 2: Exemplary data sample. The inputs to the MixNet, which are object history and sampled track boundaries, and the different prediction modes of a rail-based approach, the benchmark model and the MixNet are shown. MixNet combines the best of two worlds: the superposition of continuous base curves and the comprehensive, learned scenario understanding.

2 Related Work

To give an overview of the state of the art in the field of motion prediction, we structure this section as proposed in (Karle et al., 2022). Accordingly, the methods are categorized based on the related investigation methodology, how to describe the object's motion, into physics-based, pattern-based, and planning-based approaches. After introducing the methods for motion prediction in general, we discuss the applicability for autonomous racing in the last subsection. For further details about the overall state of the art of software for autonomous vehicle racing the reader is referred to (Betz et al., 2022b).

2.1 Physics-Based Approaches

In non-interactive scenarios with independent vehicle behavior and for short-term prediction up to 2 s, the application of kinematics- or dynamics-based vehicle models is a suitable choice (Thrun et al., 2006). Most commonly, deterministic or probabilistic kinematic models are propagated forward using a constant input assumption. One can choose various longitudinal and turning-related signals to be the input. Hence models with constant velocity, acceleration, turning radius and steering angle or the combination of these can be obtained. Schubert et al. (Schubert et al., 2008) provide a review of these models and conclude that Constant Turning Radius and Acceleration (CTRA) models provide the best compromise between prediction accuracy and computational demands. To incorporate uncertainty information, Bayesian Filters, especially Kálmán Filters (Kalman, 1960) are commonly used. If the linearity assumption of the Kálmán Filter does not hold, the Extended (EKF) or Unscented (UKF) versions are applied. By means of an Interaction Multiple Model (IMM) various kinematic models can be combined based on heuristics to improve the expressiveness in heterogeneous scenarios (Barrios and Motai, 2011).
Another physics-based approach are reachable set predictions (Koschi and Althoff, 2017), which utilize the set of physically possible behaviors. Thus all possible trajectories within the dynamic limits are determined, which are covered by a convex hull. Considering only the trajectories allowed by the traffic rules can limit the solution space accordingly and allows the use in a real vehicle for online verification (Pek et al., 2020). For the use in autonomous racing, these types of predictions are too conservative for the long-term prediction due to the wide range of driving dynamics of the racing vehicle and the lack of explicit rules as in road traffic. The application of the online verification concept in a supervisor module with the use case of autonomous racing is shown in (Stahl et al., 2020).
In general, physics-based methods are computationally cheap and their operation is transparent and well studied, which makes them appealing in safety-critical application domains such as autonomous driving. The main limitation arises from the simplified input assumption. This results in fairly accurate predictions on the short horizon. However, the prediction gets quickly outdated in the long term (>2 s) as the assumption of constant movement does not hold anymore. Hence, these methods are often combined with other approaches which tend to produce more reliable long-term predictions (Xie et al., 2017; Deo et al., 2018).

2.2 Pattern-Based Approaches

Pattern-based approaches build on the idea of taking observations of an object, matching it to a pattern, and carrying out the prediction based on it. The pattern assigned to a vehicle can be handcrafted or learned. Furthermore, patterns can be learned in physical by clustering data points or in an abstract space such as the hidden representations of Encoder-decoder models. Most of these methods are data-based approaches, hence the need for sufficiently rich datasets is inherent.

When using handcrafted patterns, the motion of an object is assigned to one of the predefined maneuver classes. Then, the output prediction is constructed considering the assigned maneuver type and usually involves the usage of prototypes. The classification can be carried out based on heuristics (Houenou et al., 2013) or ML models, such as Support Vector Machines (SVMs) (Mandalia and Salvucci, 2005), Hidden Markov Models (HMMs) (Yuan et al., 2018; Li et al., 2019b; Bicego et al., 2004) or RNNs (Khosroshahi et al., 2016). HMMs and RNNs are commonly used due to their inherent capability of interpreting the temporal evolution of a motion history. Instead of using handcrafted patterns, it is also possible to learn clusters from data. Afterwards, a single prototype trajectory (Morris and Trivedi, 2011) or a probabilistic representation (Krüger et al., 2019) is obtained for each cluster. During inference time, the output is constructed by classifying the given scenario into a specific cluster and applying the respective representant.

Encoder-decoder neural network architectures have recently shown huge potential for motion prediction tasks by dominating the leaderboards of various prediction challenges on public real-world datasets (Chang et al., 2019; Caesar et al., 2020). For the use case of motion prediction, the encoder creates a latent summary about the motion history of an object and the environment information, which serves as an abstract pattern of the scenario. Conditioned on this abstract representation, the decoder determines the future movement of the object. The encoder and the decoder models can both be based on Convolutional Neural Networks (Harmening et al., 2020; Ridel et al., 2020), RNNs (Sadeghian et al., 2018; Li et al., 2019a) or even Convolutional-RNNs (Xingjian et al., 2015; Ridel et al., 2020).

Attention mechanisms are also commonly used, to allow the network to focus on the relevant parts of the input (Sadeghian et al., 2018; Li et al., 2019a) or to take interactions between vehicles into account (Mercat et al., 2020). Alternatively, Graph Neural Networks (GNNs) can be used to model interactions in a non-euclidean space. A learned graph representation of the scene is applied to model individual interaction between the single agents. The application on public real-world datasets shows a significant improvement in prediction performance (Salzmann et al., 2020; Zhao et al., 2020). Another advantage of graph-based prediction models is the more efficient representation in contrast to grid-based approaches, which leads to a reduced calculation time (Gao et al., 2020). The bottleneck to provide a sufficiently rich dataset to train a performant algorithm is mitigated in terms of Encoder-decoder architectures due to the fact that no labeled data is needed to create a dataset. The reason for this is that the observation of an object's movement serves as ground truth for output of the prediction algorithm. Thus, the trajectories can simply be split into history and ground truth prediction parts and used in the self-supervised learning setup (Geisslinger et al., 2021).

2.3 Planning-Based Approaches

Planning-based approaches consider objects to be rational agents acting in an environment according to their hidden policies in order to reach their goals. The basis of planning-based approaches is the Markov Decision Process (MDP). The approaches usually differ from each other in approximating different parts of the MDP. One approach is to derive handcrafted cost functions to model the agent's behavior (Rudenko et al., 2018). However, the creation of a comprehensive cost function and rule set requires dedicated knowledge and complex traffic scenarios are challenging to represent. Alternatively, the driving behavior can be learned from data, which refers to the field of learning from demonstration. On the one hand, it is possible to apply Inverse Reinforcement Learning, which aims to derive a cost function that fits the observed expert behavior (Sun et al., 2018; Deo and Trivedi, 2020; Fernando et al., 2020). On the other hand, in the case of Imitation

Learning the policy is directly learned from the observed data, i.e., the observation is directly related to a specific behavior (Mayank Bansal et al., 2019). To enhance the robustness of the learned policy generative methods are used, one common approach is Generative Adversarial Imitational Learning (Kuefler et al., 2017).

Both pattern-based and planning-based approaches are usually capable of producing more reliable long-term predictions compared to physics-based methods because more comprehensive features can be inputted to the model to consider interaction between different road users and map constraints. Their shortcomings come from the fact that these methods are primarily data-based and rely on learned policies or patterns. Hence, their performance highly depends on the underlying dataset, which has to represent the ODD sufficiently. Besides the amount of data, the balance of scenario types influences the prediction performance in edge cases such as safety-critical situations. So, the under-representation of these scenarios directly impacts the applicability. The application in combination with safeguarding methods is also challenging as data-based methods lack of explainability. Hence their behavior can not be supervised properly.

2.4 Applicability for Autonomous Racing

In the following the state of the art is evaluated regarding the applicability for autonomous racing. In our case we focus on the Indy Autonomous Challenge (IAC), one of the most advanced autonomous racing series, which is the target for the presented approach. In the inaugural edition the IAC was held on the Indianapolis Motor Speedway (IMS) and the Las Vegas Motor Speedway (LVMS), which are both oval circuits with additional simulation races in advance. The circuits do not offer any lanes to define lane-keeping or lane-changing maneuvers. Also, overtaking can have many different forms because the lane layout does not constrain it as in normal road traffic and it is assumed to be non-cooperative. Besides these constraints resulting from the race format, it also has to be considered that no public race dataset is available and the execution time on fixed computational resources has to fulfill real-time requirements. Due to the overall software concept (Betz et al., 2022a), it is required to provide a prediction horizon of 5 s, which is another important constraint. In consideration of missing traffic lanes, the prediction length does not allow the usage of purely physics-based methods as constant movement assumptions do not hold and the set of dynamically reachable states gets too large. The factors of missing lane information and data discourage the application of classification or clustering-based approaches. The application of planning-based algorithms is questionable because these algorithms require even more comprehensive data to derive the expert behavior properly and, especially in terms of IL, the robustness towards outliers is not given.

The design factors that motivate our approach, the MixNet, is to combine data-based encoding with dynamically feasible superposition of base curves are the real-time capability and the need for robustness and performance guarantees. By means of the Encoder network we can extract patterns from observation. These patterns cover comprehensive object behavior, so complex scenarios can be modeled. The superposition of base curves in the decoder constrains the possible prediction to lie inside the race track and it guarantees robustness in case of outliers in the input. Thus, the applicability can be enhanced significantly. However, the superposition of base curves is still flexible enough to output a high prediction accuracy as the evaluation (section 4) shows. To model interactions and to output collision-free predictions a rule-based fuzzy logic is implemented which can be applied to the MixNet's output.

3 Method

In this section, we introduce MixNet, our encoder-decoder neural network architecture for motion prediction. Besides the network architecture, we additionally describe how to incorporate external velocity information, trigger safety interventions and model interactions between objects. Finally, we describe the data mining and training procedure.

3.1 Network Architecture

The proposed network architecture of the MixNet is shown in Figure 3. The LSTM layers in the encoder create a latent summary through encoding the object motion histories. The inputs to the network are the $H = 30$ historic 2D-positions up to 3 s in the past, sampled with $f = 10$ Hz. Besides that, the relevant map information, which are the left and right track boundaries starting from the current object's position, equidistantly sampled in vector representation, is inputted to the network. Considering the expected racing speed and prediction horizon, we sample the boundaries up to a horizon of 400 m with 20 m. During inference N objects are fed into the network batch wise. Figure 2 shows an exemplary input consisting of history and track boundaries. The hidden states of the encoding LSTMs are then concatenated and passed to a linear layer, which outputs the latent representation of the scenario. The decoder, the generative part of the model, creates a prediction conditioned on the latent summary. In contrast to other works, the future states are not the direct output of an LSTM network, but the forecast trajectory is generated systematically from known schemes. First of all, the trajectory is obtained by predicting a path and applying a velocity profile to it. Both of these components are generated in a constrained way. The path is created through superpositioning various predefined base curves according to weights predicted by the network. The velocity profile is piece-wise linear and is determined by predicting an initial velocity and five constant acceleration values: one for each second of the prediction horizon. The final output trajectory is obtained by resampling the path according to the velocity profile. In this way, the set of possible output trajectories is constrained by construction.

We use four base curves for superpositioning: the two track boundaries, a pre-computed minimal curvature raceline (Heilmeier et al., 2019) and the centerline of the track. The curves are represented by discrete 2D-points of equal number. The points of the boundaries and the raceline are defined by their distances from the centerline points. Hence, the points of each curve at any index i correspond to the same cross section on the track. Due to this fact, the superpositioned curve can be obtained as follows:

$$\rho_{sup}(s) = \sum_{c \in \mathcal{C}} \lambda_c \rho_c(s) \text{ with} \tag{1}$$

$$\lambda_c \in [0, 1] \;\; \forall c \in \mathcal{C} \text{ and } \sum_{c \in \mathcal{C}} \lambda_c = 1 \tag{2}$$

where $\rho_{sup}(s)$ is the superposed curve along the arc length s of the track with base curves $\rho_c(s)$ and their corresponding weights λ_c in the set of base curves $\mathcal{C}$. Due to the fact that all base curves are sampled along the same cross sections, s is equal for all base curves. Equation 2 provides the constraints for the superpositioning weights, which result in curves that lie between the left and right boundaries. These constraints can easily be enforced by applying the *softmax* activation to the output of the linear layer.

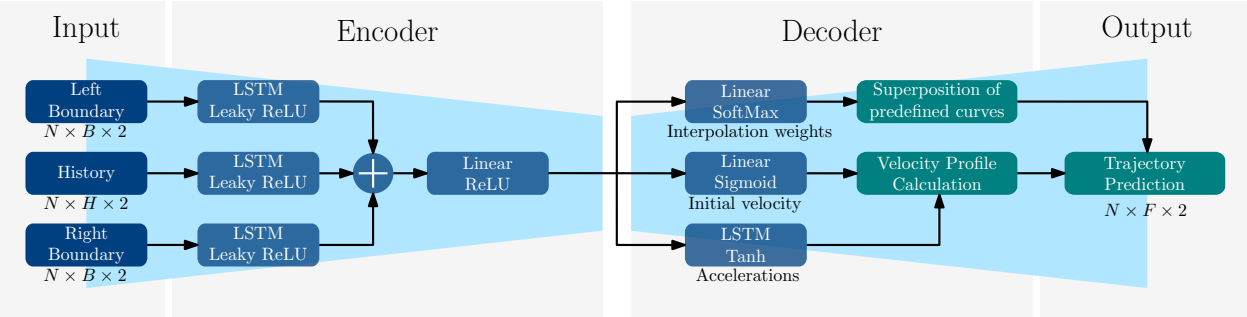


Figure 3: The architecture of MixNet. The prediction is composed of a path prediction by superposition of base curves and the prediction of an acceleration profile to apply a piece-wise linear velocity profile. Inputs to the network are $H = 30$ past 2D-positions of N objects and the related left and right track boundaries in driving direction in vector representation. Output is the trajectory prediction of N objects in 2D with a horizon of $F = 50$ steps.

We emphasize that this method does not guarantee that a predicted trajectory starts exactly from the actual position of a vehicle, but a lateral offset can occur. This is due to the fact that the superposition weights by Equation 1 are constant along the path and the model is trained to output a prediction with the smallest overall error along the prediction horizon. Hence, a lateral offset at the beginning of the predicted path is possible. To remedy this issue, during the first second of the motion, we apply a plausibility check to the output of the network by means of a comparison between the current object position and the predicted path. Above a specified threshold, a correcting shift is triggered to ensure a consistent transition between the current position and the predicted path. Especially in transient scenarios, this characteristic is beneficial: If the current motion of the vehicle is strongly transversal, the network learns to predict a path that fits the later parts of the ground truth better by sacrificing accuracy at the beginning of the horizon. With the application of the correction shift, the consistency of the predicted motion is secured, so an additional quality guarantee can be given. During inference, if an adjustment of the the first part of the trajectory is triggered to connect the actual position to the prediction path, trajectories similar to lane-change maneuvers can be obtained.

3.2 Velocity Information Fusion

A further advantage of the modular trajectory prediction is that it can also incorporate velocity information from another source to fuse it with the outputted path of the network. The proposed implementation of the MixNet offers two possibilities for incorporating external velocity information. First, one can take the complete velocity profile from an external source and use it instead of the one predicted by MixNet. The other possibility is to predict the piece-wise constant accelerations with MixNet but use the external velocity information only to determine the current velocity as initialization.

A reliable source of such information is object tracking which contains the filtered states of the surrounding vehicles. While the current velocity can be extracted from the tracked object's state independently from the ODD a complete velocity profile underlies specific assumptions. For our use case of autonomous racing on a closed track, we propose to determine a complete velocity profile along the whole prediction horizon by forward propagating the tracked object's state by means of the underlying state-space model. However, this assumption is limited to scenarios with objects at race speed. Experimental evaluation shows that the combination of the tracked initial velocity and the piece-wise constant acceleration predicted by the MixNet provides the most accurate velocity profile predictions. Hence, similar to the path prediction, the combination of an external constraint by means of the initial velocity and the scenario-aware data-based acceleration prediction performs best.

3.3 Safety Override Possibility

Predicting accurately at the beginning of the horizon is a safety-relevant issue. On the other hand, large inaccuracies towards the end of the prediction horizon lead to inefficient planning and hence bad performance during the race. Due to the constant superposition weights, this trade-off especially occurs at the turn-in point and during overtaking maneuvers. To solve this conflict, our prediction method recognizes and overrides dynamically infeasible predictions, that is, predictions which have a large lateral offset at the beginning of the horizon. Accordingly, our goal is to probabilistically identify the cases where this happens. Our measure of probability for having generated an initially highly inaccurate forecast is based on the raw path prediction output of the MixNet and the actual state of the vehicle. As stated before, without adjustment, the predicted path does not necessarily start at the actual position of the object. Thus, if the predicted path lies too far from the vehicle considering its actual position and orientation, the prediction is identified as invalid. In this case, we override it with a prediction approach, which we call *rail-based* prediction. This approach is derived from the tracked object state and offers a high robustness and guarantees kinematic consistency with the current object's state, which is of high importance as fallback option. The rail-based prediction is composed out of a separate path and velocity profile. The path is sampled starting from the current object position in parallel to the track boundaries. The velocity profile is determined by forward propagation of the state-space

model as described in 3.2. An exemplary prediction by means of the rail-based approach is shown in Figure 2. It can be seen that the rail-based prediction is consistent and accurate at the beginning of the prediction horizon, but has a high lateral error on the long-term horizon.

3.4 Interaction Modeling

As it can be seen in Figure 3 no information about surrounding race cars is fed into the neural network. Hence, there is no explicit interaction-awareness given by the model. Although this assumption holds for many race scenarios with cars following their raceline, interactive scenarios such as overtaking or blocking, which are highly interactive, require additional modeling. In the literature, these interactions are modeled using game theory (Smirnov et al., 2021) or learned by a prediction network (Deo and Trivedi, 2018). These approaches have the disadvantage that they either require a lot of computing time due to iterations or require a vast amount of relevant data with interactions.

Therefore, we propose a two-step approach for trajectory prediction on the racetrack. This first predicts each vehicle by itself and then adjusts the predicted trajectories based on interactions in a second step. In doing so, we take advantage of the set of rules in motorsports. Similar to other racing series such as Formula 1, for safety reasons the movement options of an overtaken vehicle are restricted by the rule set. In the Indy Autonomous Challenge, the vehicle in front is even forced to "hold its raceline" (Network, 2019). In other words, it may not block the overtaking vehicle or initiate other unpredictable maneuvers.

Our approach is to predict each vehicle individually, including the ego-vehicle, as a first step. Subsequently, all existing predicted trajectories of the different vehicles are examined for collisions. If no collision is detected, we assume that the influence of the interaction is minor and no adjustment of the trajectories is necessary. However, if at least one collision is detected, we do adapt the trajectories to account for interactions. For this, we examine the race order for each collision and adjust the predicted trajectory of the rear vehicle, since the front vehicle is not allowed to adjust its behavior according to the rules.

To adjust the predicted trajectory, a high-level decision is first made: the faster rear vehicle will either overtake on the left, overtake on the right or not overtake at all and stay behind the front vehicle. This high-level decision is made by a fuzzy logic that takes into account the absolute and relative positions as well as the velocities of the participants. According to the decision, the predicted trajectory is adjusted laterally (in case of an overtake) or longitudinally (in case of no overtake). The adjusted trajectories do not necessarily have to be collision-free, since new collisions with other predicted trajectories may occur. Therefore, the procedure can be repeated as often as necessary until all predictions are resolved collision-free or a termination criterion occurs. In our application, it has proven to be useful in a second iteration to ensure only that the ego vehicle is collision-free with vehicles in the rear. Otherwise it could happen that the trajectory planning tries to avoid this collision and even makes room for an overtaking vehicle, which would be contrary to winning the race.

3.5 Dataset

An essential part of successful Machine Learning applications is a rich dataset, which covers a diverse set of scenarios expected during inference. However, since this race is the first of its kind, building a dataset from previous races is not possible. Also, there is no public racing dataset available. In this respect, one of the key challenges to be solved is building a dataset for training our neural network approach.

Real-life data was not available until the last weeks before the race itself since real-world tests on the IMS track only took place then. Hence, training data had to be acquired through simulating our software pipeline (Betz et al., 2022a) against itself on a high fidelity Hardware-in-the-Loop (HIL) simulator. The HIL-simulator is able to simulate the full autonomous software stack of up to ten agents in real-time with the same interfaces and callback functions as on the real vehicle. The only constraint is a simplified perception input to reduce

the computation load, i.e., the perception pipelines are bypassed with a synthetic object list generator that is input to the tracking module. However, this generator comprises various features to imitate real perception behavior such as limited sensor range and field-of-view, addition of Gaussian and normal noise, variation of measured objects states and the simulation of false positives and false negatives. Multi-vehicle races on the HIL-simulator and data from the official simulation race of the IAC, both with highly interactive and complex scenarios, are the basis of our dataset. The recorded logs contain the output of the tracking module, which are used to recreate the trajectories of each vehicle during a race. Using the tracking module output to build our dataset has the advantage that the training input distribution of tracked objects and tracking quality will be as close as possible to the expected input distribution during the race. Besides that, the synthetic object list generator is used to vary the perception quality with the aforementioned features to augment the dataset. We added Gaussian noise during the data generation process with mean $\mu = 0$ and different variance in longitudinal and lateral directions based on the evaluated perception and tracking performance.

From these recovered trajectories and the map of the racetrack, it is possible to generate a dataset for training the model. As it was stated before, the input of our model consists of the (x, y) positions of the history trajectory and the track boundaries around and ahead of the vehicle. Before inputting these points to the model, they are transformed into a local coordinate system, which has its origin at the left boundary next to the current object position and is oriented with its x-axis tangential to the left boundary. With the process above we managed to recreate 1,569 trajectories from 226 races with in average 3.4 vehicles per race. From these trajectories, we created 358,025 input-output data pairs.

3.6 Training

The loss function we defined for training has two terms, one for costing the fit of the interpolated curve, which relates to the error in lateral direction, and one for the velocity profile, which reflects the accuracy in the longitudinal direction. The loss L_{path} related to the lateral deviation of the interpolated curve $\hat{\boldsymbol{x}}$ from the ground truth $\boldsymbol{x}$ is costed with a Weighted Mean Square Error (WMSE) at the beginning of the curve as follows:

$$L_{\text{path}} = \frac{1}{F}\left(L_{\text{wmse}} + L_{\text{mse}}\right) \tag{3}$$

$$L_{\text{wmse}} = \sum_{i=1}^{k} (\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}})^2 \left(1 + w\left(1 - \frac{i-1}{k}\right)\right) \tag{4}$$

$$L_{\text{mse}} = \sum_{i=k+1}^{F} (\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}})^2 \tag{5}$$

$$\text{with } w = 0.5, k = 10, F = 50 \tag{6}$$

The WMSE decreases linearly from the first prediction step with an additional weight w along the weighting horizon k. The remaining prediction horizon is weighted with 1.0, which corresponds to the conventional Mean Square Error (MSE). The weighting turned out to be beneficial to limit the lateral offset at the beginning of the prediction, even if an additional correction shift is applied at inference. The loss of the velocity profile is also the MSE coming from comparing it to that of the ground truth velocity profile. The overall loss L is then obtained from the path loss L_{path} and the velocity loss L_{vel} as follows:

$$L = L_{\text{path}} + \Delta t^2 \cdot L_{\text{vel}} \tag{7}$$

Where $\Delta t = 1/f = 0.1\,\text{s}$ is the timestep size of the prediction in seconds. The multiplication of the velocity error with Δt^2 comes from the following intuition: The velocity error e_v results through integration in a

$\Delta t \cdot e_v$ displacement error and a $\Delta t^2 \cdot e_v^2$ squared displacement error, if the MSE is applied. By considering the relation $e_v^2 \sim L_{\text{vel}}$, it follows that multiplying the velocity loss term with Δt^2 is necessary to be able to add two loss terms of an identical unit, which is m^2.

The hyperparameters are optimized by Bayesian optimization (Nogueira, 2014). To train the network we use a learning rate of $5 \cdot 10^{-5}$ with a rate decay of 0.997 per epoch. The L_2 weight regularization has a strength of 10^{-7} and we use a batch size of 128. We train the model for 50 epochs and take the model with the best validation for the evaluation presented in Section 4. The final net and training parameters are published in the open source code.

In conclusion, the proposed method offers guarantees that the predicted trajectories are always smooth and lie inside the racetrack by means of the structured composition of the prediction. Besides that it is possible to fuse velocity information from other sources such as the state estimation to enhance the kinematic consistency of the predicted trajectory. Finally, it is possible to probabilistically detect predictions that are highly incorrect at the beginning of the horizon. In these cases, it is possible to override the predictions. For autonomous racing on a closed track, we propose to use a rail-based prediction, which outputs a constant velocity trajectory in parallel to the track boundaries starting from the current object's position.

4 Experiments

In this section, we describe the test procedure, which comprises details about the test dataset and present a comprehensive evaluation of MixNet's prediction performance. The conducted experiments reveal the overall prediction performance and analyzes the model's robustness. Besides that, we investigate the composition of the base curves.

4.1 Test procedure

The set of predictions that are reachable through MixNet is constrained. This is exactly what allows for the quality guarantees mentioned before. To demonstrate that, despite this boundedness, MixNet is still flexible enough to output an accurate prediction, we compare its performance to a purely LSTM-based encoder-decoder architecture, which we define as benchmark model. The benchmark model's encoder architecture is identical to MixNet's. The difference lies in the decoder architecture, which is in case of the benchmark model also constructed with LSTM-layers. Thus, the model directly iteratively outputs a 2D trajectory prediction with a shape of $F \,\text{x}\, 2$ for N objects. MixNet and the benchmark model have $198,214$ and $193,797$ trainable parameters respectively. We train both models on the same dataset which is described in subsection 3.5. For MixNet, we incorporate initial velocity information from object tracking into the velocity profile, but use the piece-wise constant acceleration outputted by the MixNet.

For reproducibility, we have recorded 10 scenarios on our HIL simulator. These include interactive race scenarios with different numbers of vehicles with various speed limits. The recordings can be replayed identically to the pipelines using the two prediction models. For the recording of the interactive scenarios, we used MixNet as the prediction model to run the full software stack. We would like to emphasize that this does not induce any bias in the evaluation process as the respective planning behavior of each objects differs from the MixNet model behavior. We report the performances of the models by analyzing the absolute error distributions in the lateral and longitudinal directions with respect to the horizon length using the Mean Average Error (MAE), which is defined as follows:

$$\text{MAE} = \frac{1}{F} \sum_{i=1}^{F} |\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}}|_2 \tag{8}$$

4.2 Overall Prediction Performance

The overall MAE on the recorded interactive scenarios of MixNet and the benchmark model are 4.91 m and 5.36 m respectively. The reason why MixNet, although being constrained, outperforms the benchmark model becomes apparent when we look at the lateral and longitudinal error distributions on the prediction horizon illustrated in Figure 4. As it can be seen, the magnitude of errors in the lateral direction is very similar in the two cases. This result justifies the hypothesis that obtaining the prediction path by superpositioning our chosen base curves covers the set of possible trajectories properly. The superiority of MixNet in the overall error comes from the fact that it produces smaller errors in the longitudinal direction. Thus, it can be concluded that the combination of the initial velocity from the tracking module and the assumption of piece-wise constant acceleration models the longitudinal movement of the objects more accurate than the iteratively determined output of the LSTM-decoder.

The error distributions with respect to the average velocity of the history are shown in Figure 5a. The predictions are separated into the three bins $v < 30\,\mathrm{m\,s^{-1}}$, $30\,\mathrm{m\,s^{-1}} < v < 60\,\mathrm{m\,s^{-1}}$ and $60\,\mathrm{m\,s^{-1}} < v$ based on the average velocity of their histories, which are associated with a slow-speed and start scenario range, a mid-speed range and a top-speed range. As Figure 5a shows, the models have similar characteristics regarding their velocity dependent accuracy. The mean error is the highest at low speeds and it gradually decreases as the velocity grows. There are two main reasons for this. Firstly, most of the transient scenarios like start scenarios, which are challenging to predict, happen at lower speeds. Once the cars have reached their normal racing speed, the scenarios tend to be more steady in the longitudinal direction. Secondly, since most of the racing happens at high speeds, the majority of the training data could be acquired in this velocity range. The number of vehicles does not have a large effect on the accuracy of the predictions (Figure 5b). This is due to the fact that the fuzzy logic can resolve prediction conflicts accurately by deciding between right and left overtake. Moreover, maneuvers with more than two vehicles racing wheel-to-wheel at the same time, which would result in strong lateral interaction, are rare. Instead, scenarios with more than two vehicles mainly result in sequential overtaking maneuvers between two vehicles respectively.

4.3 Robustness

To investigate the robustness properties of both data-based approaches, we have replayed the test scenarios with extra Gaussian noise added. It should be noted that the original test data is already noisy, but its magnitude is much smaller. We carried out several experiments, all with zero-mean disturbances with different variances in the lateral and longitudinal directions. Table 1 reviews the MAEs of the approaches in the different test cases. As the analysis reveals, MixNet and the benchmark model are both robust against

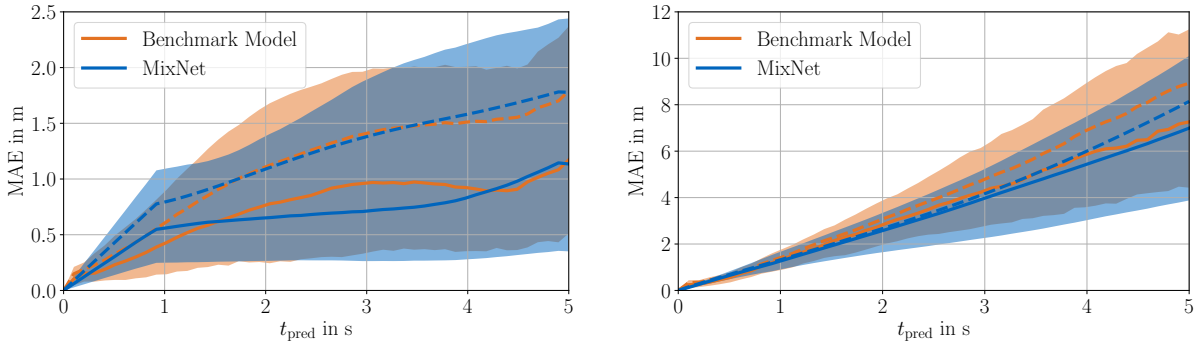
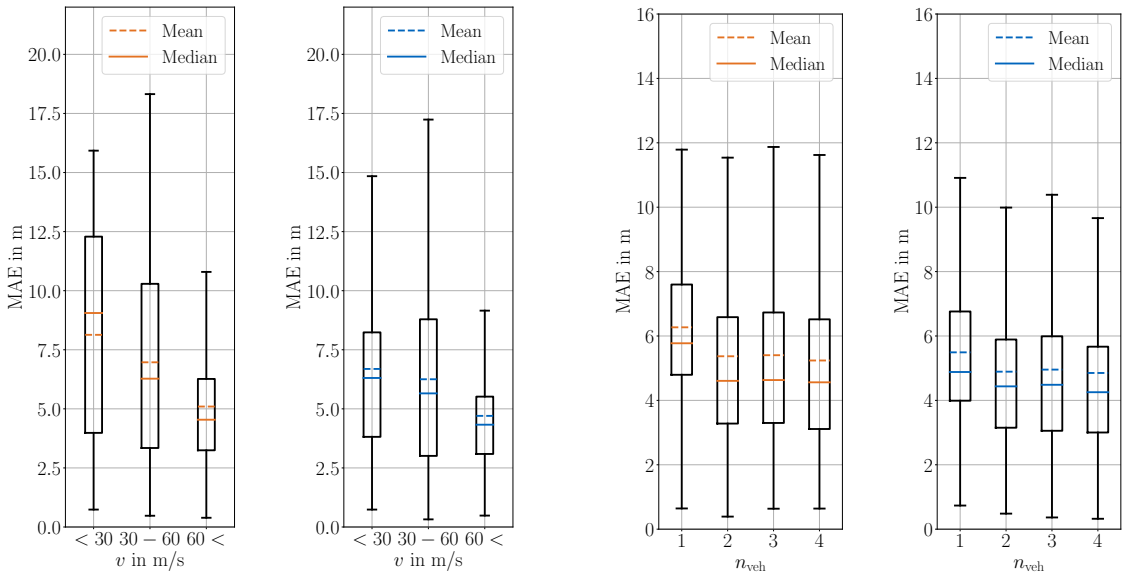


Figure 4: Lateral (left) and longitudinal (right) error distribution on the prediction horizon. The solid and dashed lines denote the median and mean errors respectively. The colored areas illustrate the range between the Q1 and Q3 quartiles.



(a) Object velocity.

(b) Number of objects.

Figure 5: The error distributions of the benchmark model (orange) and the MixNet (blue).

the added noise in the chosen range, although performance degradation occurs. However, it can be noticed that the MAE of the MixNet is lower in all cases. In the case of adding noise in lateral and longitudinal direction, the MAE of the benchmark model increases less than that of the MixNet in case of a standard deviation of 0.5 m in each direction. However, a bigger standard deviation of 1.0 m in both direction results in a significant increase in the MAE of the benchmark model by 21.2%. In contrast, the MixNet's relative increase in the MAE is only 13.4%. This observation indicates that the constraints applied to the MixNet result in the desired model behavior that the model is more robust against variations of the input. The application of Gaussian noise in only one direction reveals that the benchmark model is less sensitive in the lateral direction. However, the relative increase in the MAE for both models is similar. The application of Gaussian noise in the longitudinal direction clearly shows a big advantage of the MixNet. The incorporation of external velocity information from the tracking module results in a significantly more robust prediction accuracy as the relative increase in the MAE is only half as big as in the case of the benchmark model.

Table 1: Robustness of the the benchmark model and MixNet against zero-mean Gaussian noise.

Standard Deviation		**Benchmark Model**		**MixNet**	
σ_{lon} in m	σ_{lat} in m	abs. MAE in m	rel. MAE in %	abs. MAE in m	rel. MAE in %
0.0	0.0	5.36	-	4.91	-
0.5	0.5	5.61	+4.6	5.23	+6.5
1.0	1.0	6.50	+21.2	5.57	+13.4
0.0	1.0	5.76	+7.5	5.39	+9.7
1.0	0.0	6.09	+13.6	5.29	+7.7

4.4 Superpositioning weights

To investigate how consistently MixNet predicts the weights for curve superpositioning, we input synthetic history trajectories generated with random weights at the entrance of one of the turns on the racetrack. We then observe how well the outputs of MixNet match the inputs. Figure 6 illustrates the input and output weights of the four base curves in scatter plots.

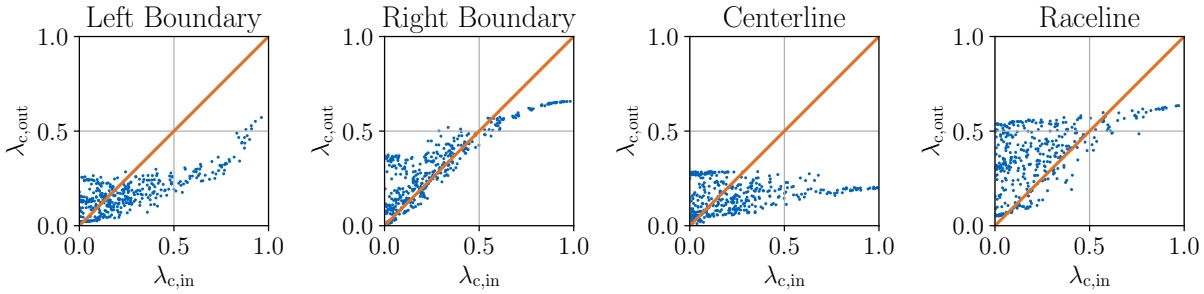


Figure 6: The relation between the inputted superpositioning weights of the synthetically generated trajectories and the outputted weights of MixNet.

Ideally, the model would output the exact same weights with which the input history is built up resulting in 45° straight lines in all of the plots. The closest to this is the figure of the raceline weights. Meanwhile, the centerline weights hardly seem to correlate with the input at all. Generally, the network seems to overuse the raceline. The reason for that is that the trajectories in the data usually have a strong raceline component. Hence, the network often assumes raceline following, although the current object might differ from the raceline. Such an example of raceline overuse can be seen in Figure 7. In the turn, the prediction fits the ground truth very well close to the inner bound. However, in the turn exit, the model assumes more optimal driving and a lateral error occurs compared to the ground truth, which follows a more narrow line. The example also reveals that even though the same lateral position at a specific point results, a different combination of superposition weights influences the overall prediction path. As it can be derived from Equation 1 and 2 any point on the racetrack can be obtained through infinitely many different weighting combinations of the 4 base curves. Thus, in the shown scenario, a higher weight of the left boundary would result in the same lateral position in the turn, but a different, in this case the correct, position at the turn exit.

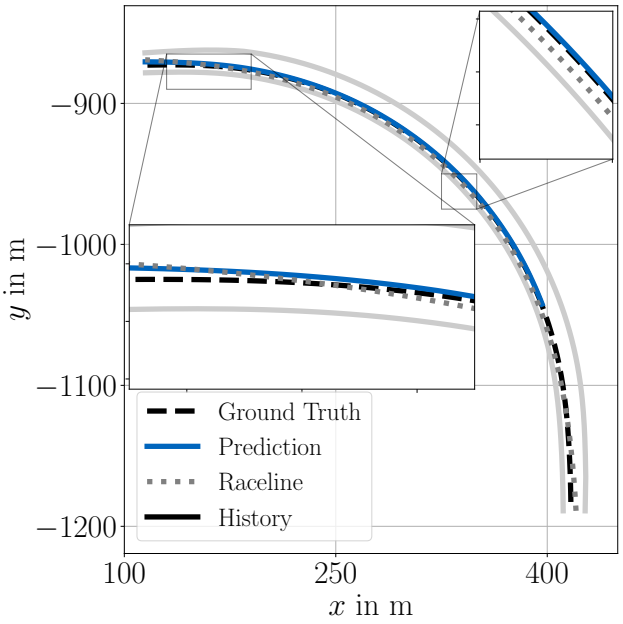


Figure 7: Exemplary scenario with overuse of the raceline in a turn.

Even though the superposition weights differ from the inputted ones, the overall errors of the prediction are very low in these synthetic cases. The fact that the centerline is underused indicates that this base curve

is indeed redundant as it lies close to the middle between left and right boundary for the major part of the track. Hence, we conduct the following analysis to prove this hypothesis and approximate the centerline as follows:

$$\rho_{center}(s) \approx 0.5 \cdot \rho_{left}(s) + 0.5 \cdot \rho_{right}(s) \tag{9}$$

In this case, the weights corresponding to the centerline can be redistributed and added to the left and right boundary weights without changing the overall superpositioned output as follows:

$$\lambda_{left} = \lambda_{left} + 0.5 \cdot \lambda_{center} \tag{10}$$

$$\lambda_{left} = \lambda_{left} + 0.5 \cdot \lambda_{center} \tag{11}$$

If we plot the left and right boundary weights again, we get the input-output weight relationships illustrated in Figure 8. Here, the weights for the right boundary already match very well the desired linear figure. The left boundary is still mostly substituted by over weighting the raceline. From this we conclude that the superpositioning weights produced by the network, although they do not match exactly the input weights, which is also not expected, due to the redundancy of the base curves, are consequent and reasonable.

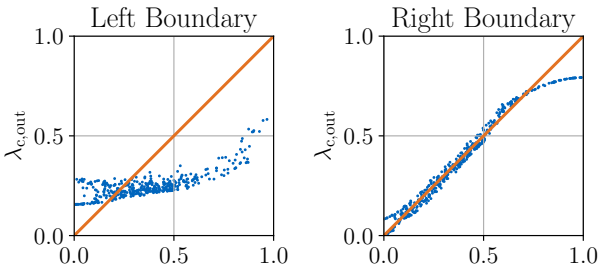


Figure 8: The relation between the input superpositioning weights of the synthetically generated trajectories and the outputted weights of MixNet if one redistributes the weights of the centerline to the right and left boundaries.

4.5 Computation time

The average computation times for predicting four vehicles on a single core of an Intel i7-4720HQ 2.6 GHz CPU for MixNet is 9 ms and 15 ms for the benchmark model. The models have encoders of identical sizes, but the LSTM-decoder of the benchmark model takes longer to execute due to iterative calculation of the vehicles' future trajectory, which is the reason for the higher calculation time. Even on a CPU-single core the computation time is low enough that the model can be applied in a full software stack without access to the GPU, which would reduce the computation time significantly.

5 Future Work

An interesting future research direction could be to determine a richer set of base curves for superpositioning. Including for example base curves which cover lateral motion on the track could result in even higher expressiveness. Additionally, the flexibility of the approach could be enhanced by outputting multiple weights along the path instead of one constant set of weight parameters. Thus, the trade off between accurate prediction towards the end of the prediction horizon and a low lateral offset between the current object position and the start of the predicted trajectory could be solved. However, both future directions require more data and add complexity to the training process. Besides that the robustness for real-world application has to be reviewed in case of the more flexible network with varying superpositioning weights.

Moreover, interaction-awareness could be improved by directly incorporating surrounding objects in the network input. The state of the art offers various methods such as grid-based methods or GNNs to model interactions. Our presented rule-base approach is robust and explainable and performances well with a low number of cars (< 4). However, the scalability of the approach is limited because the sequential application of the rule-base overtaking decision results in an increasing number of prediction collisions, when aplied to a higher number of vehicles.

Another idea that would be worth investigating is how the knowledge of MixNet could be generalized to previously unseen tracks. Although the predicted path can be constrained by choosing the respective base lines to lie inside them, the learned scenario understanding depends on the applied base curves. This can be explained by the fact that the network itself does not know the base curves, hence the forecasts on another racetrack, which inherently comes with different base curves, would not result in meaningful forecasts. This shortcoming, however, could be remedied if the base curves would also be input to the model. Another option enhance the generalization capabilities is to build a modular network with a a backbone network with a general motion encoding, which is trained on a large scale dataset and a track specific head.

Lastly, we would like to draw the attention to an additional feature of MixNet that could be further explored. Namely, that the path predicted does not only exists in the *future*. Superpositioning the base curves according to the predicted weights results in a path that covers a complete lap on the track. Hence, it also exists behind the current position of the vehicle. This means, that the predicted path can directly be compared to the history trajectory of an object. This can be useful for example in obtaining a confidence value for each forecast. It is somewhat similar to what we have exploited when we filtered and overrode forecasts which were identified as potentially incorrect at the beginning of the horizon. However, instead of using only the most resent position of the vehicle, one could use all the historical states and hence get a picture of how well the predicted path fits the observed history of the car.

6 Conclusion

Autonomous driving has been an emerging research field in the past few decades. To fuel innovation, autonomous car races have been and are being organized. The first high-speed wheel-to-wheel competition was the Indy Autonomous Challenge. This paper described the motion prediction algorithm of the team TUM Autonomous Motorsport developed for this race.

Our solution is an encoder-decoder neural network architecture, called MixNet, which carries out the motion prediction task in a structured manner. The encoder part of the network creates a latent summary from the history trajectories and some relevant map information. The decoder then creates a path by superpositioning predefined base curves with weights predicted by the network. It also generates a piece-wise linear velocity profile, according to which the path is resampled to reach the output trajectory. By this, the approach is capable to combine a comprehensive scenario understanding of the RNN-encoder network and constrained, thus robust, prediction output superposed by kinematically feasible base curves.

The main contribution of this work is the development of a structured deep neural motion prediction algorithm that allows giving some quality guarantees about its output. The predicted trajectories are guaranteed to be smooth and lie inside the racetrack. Meanwhile these guarantees can be given, thanks to the highly restricted prediction set and the method is still flexible enough to produce accurate predictions in the majority of the cases. Also, since the velocity profile is predicted separately, it is possible to incorporate velocity information from other sources, such as the object tracking module's state estimator filters. Furthermore, it is possible to detect predictions that are probably inaccurate at the beginning of the horizon. These predictions could easily lead to dangerous behavior, hence filtering and overriding them are crucial in a safety-critical application like autonomous racing. Finally, a rule-based overtaking logic allows to resolve collisions between predicted trajectories. The algorithm is real-time capable on a single CPU core and its applicability in the overall software stack of the team is proven. The whole software module is available open source including the training and test data, a ROS2-launch configuration and a build file to create a Docker image to run the module containerized.

To demonstrate the performance of the method, we compared its accuracy to that of an unrestricted LSTM-based encoder-decoder architecture. The results underline that the highly restricted prediction set of MixNet does not cause large performance degradation. Contrarily, the lateral errors were almost identical in both cases and since MixNet is capable of incorporating velocity information from the tracking module, it even

outperformed the benchmark model in the overall accuracy. The model also shows great robustness to noised inputs. Finally, we investigated how consequently the network produces the superpositioning weights.

Contributions

Phillip Karle as the first author initiated the idea of this paper and contributed essentially to its conception, implementation and content. Ferenc Török developed the concept of MixNet, contributed the data generation, training and evaluation procedure and contributed to the writing of the paper. Maximilian Geisslinger contributed to the conception and implementation of this research and revision of the research article. Markus Lienkamp made an essential contribution to the conception of the research project. He revised the paper critically for important intellectual content. He gave final approval of the version to be published and agrees to all aspects of the work. As a guarantor, he accepts the responsibility for the overall integrity of the paper.

References

Ammoun, S. and Nashashibi, F. (2009). Real time trajectory prediction for collision risk estimation between vehicles. In *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, pages 417–422. IEEE.

Barrios, C. and Motai, Y. (2011). Improving estimation of vehicle's trajectory using the latest global positioning system with kalman filtering. *IEEE Transactions on Instrumentation and Measurement*, 60(12):3747–3755.

Betz, J., Betz, T., Fent, F., Geisslinger, M., Heilmeier, A., Hermansdorfer, L., Herrmann, T., Huch, S., Karle, P., Lienkamp, M., Lohmann, B., Nobis, F., Ögretmen, L., Rowold, M., Sauerbeck, F., Stahl, T., Trauth, R., Werner, F., and Wischnewski, A. (2022a). Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge.

Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., and Mangharam, R. (2022b). Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488.

Bicego, M., Murino, V., and Figueiredo, M. A. (2004). Similarity-based classification of sequences using hidden markov models. *Pattern Recognition*, 37(12):2281–2291.

Buehler, M., Iagnemma, K., and Singh, S. (2007). *The 2005 DARPA Grand Challenge: The Great Robot Race*. Springer Publishing Company, Incorporated, 1st edition.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Deo, N., Rangesh, A., and Trivedi, M. M. (2018). How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140.

Deo, N. and Trivedi, M. M. (2018). Convolutional social pooling for vehicle trajectory prediction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018-June:1549–1557.

Deo, N. and Trivedi, M. M. (2020). Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*.

Fernando, T., Denman, S., Sridharan, S., and Fookes, C. (2020). Deep inverse reinforcement learning for behavior prediction in autonomous driving: Accurate forecasts of vehicle motion. *IEEE Signal Processing Magazine*, 38(1):87–96.

Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., and Schmid, C. (2020). Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geisslinger, M., Karle, P., Betz, J., and Lienkamp, M. (2021). Watch-and-learn-net: Self-supervised online learning for probabilistic vehicle trajectory prediction. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 869–875.

Harmening, N., Biloš, M., and Günnemann, S. (2020). Deep representation learning and clustering of traffic scenarios. *arXiv preprint arXiv:2007.07740*.

Heilmeier, A., Wischnewski, A., Hermansdorfer, L., Betz, J., Lienkamp, M., and Lohmann, B. (2019). Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*.

Houenou, A., Bonnifait, P., Cherfaoui, V., and Yao, W. (2013). Vehicle trajectory prediction based on motion model and maneuver recognition. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4363–4369. IEEE.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Karle, P., Geisslinger, M., Betz, J., and Lienkamp, M. (2022). Scenario understanding and motion prediction for autonomous vehicles – review and comparison. *In press, IEEE Transactions on Intelligent Transportation Systems*.

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

Koschi, M. and Althoff, M. (2017). SPOT: A tool for set-based prediction of traffic participants. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 1686–1693.

Krüger, M., Novo, A. S., Nattermann, T., and Bertram, T. (2019). Probabilistic lane change prediction using gaussian process neural networks. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3651–3656. IEEE.

Kuefler, A., Morton, J., Wheeler, T., and Kochenderfer, M. (2017). Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE.

Li, J., Ma, H., and Tomizuka, M. (2019a). Conditional generative neural system for probabilistic trajectory prediction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6150–6156. IEEE.

Li, J., Zhan, W., Hu, Y., and Tomizuka, M. (2019b). Generic tracking and probabilistic prediction framework and its application in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3634–3649.

Mandalia, H. M. and Salvucci, M. D. D. (2005). Using support vector machines for lane-change detection. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 1965–1969. SAGE Publications Sage CA: Los Angeles, CA.

Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale (2019). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV*.

Mercat, J., Gilles, T., Zoghby, N., Sandou, G., Beauvois, D., and Gil, G. (2020). Multi-head attention for joint multi-modal vehicle motion forecasting. In *IEEE International Conference on Robotics and Automation*.

Morris, B. T. and Trivedi, M. M. (2011). Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2287–2301.

Network, E. S. (2019). Indy autonomous challenge rules v5nov2019.

Nogueira, F. (2014). Bayesian Optimization: Open source constrained global optimization tool for Python.

Pek, C., Manzinger, S., Koschi, M., and Althoff, M. (2020). Using online verification to prevent autonomous vehicles from causing accidents. *Nature Machine Intelligence*, 2(9):518–528.

Ridel, D., Deo, N., Wolf, D., and Trivedi, M. (2020). Scene compliant trajectory forecast with agent-centric spatio-temporal grids. *IEEE Robotics and Automation Letters*, 5(2):2816–2823.

Rudenko, A., Palmieri, L., and Arras, K. O. (2018). Joint long-term prediction of human motion using a planning-based social force approach. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4571–4577. IEEE.

Sadeghian, A., Legros, F., Voisin, M., Vesel, R., Alahi, A., and Savarese, S. (2018). Car-net: Clairvoyant attentive recurrent network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 151–167.

Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. (2020). Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 683–700, Cham. Springer International Publishing.

Schubert, R., Richter, E., and Wanielik, G. (2008). Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th International Conference on Information Fusion*, pages 1–6. IEEE.

Smirnov, N., Liu, Y., Validi, A., Morales-Alvarez, W., and Olaverri-Monreal, C. (2021). A game theory-based approach for modeling autonomous vehicle behavior in congested, urban lane-changing scenarios. *Sensors*, 21(4).

Stahl, T., Eicher, M., Betz, J., and Diermeyer, F. (2020). Online verification concept for autonomous vehicles – illustrative study for a trajectory planning module. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7.

Sun, L., Zhan, W., and Tomizuka, M. (2018). Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2111–2117. IEEE.

Thrun, S., Burgard, W., and Fox, D. (2006). *Probabilistic Robotics*. Cambridge University Press.

Xie, G., Gao, H., Qian, L., Huang, B., Li, K., and Wang, J. (2017). Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810.

Yuan, W., Li, Z., and Wang, C. (2018). Lane-change prediction method for adaptive cruise control system with hidden markov model. *Advances in Mechanical Engineering*, 10(9):1687814018802932.

Zhao, Z., Fang, H., Jin, Z., and Qiu, Q. (2020). Gisnet: Graph-based information sharing network for vehicle trajectory prediction. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7.