THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Flash: Fast and Light Motion Prediction for Autonomous Driving with Bayesian Inverse Planning and Learned Motion Profiles

Citation for published version:
Antonello, M, Dobre, M, Albrecht, SV, Redford, J & Ramamoorthy, S 2022, Flash: Fast and Light Motion Prediction for Autonomous Driving with Bayesian Inverse Planning and Learned Motion Profiles. in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2022.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Institute of Electrical and Electronics Engineers (IEEE), pp. 9829-9836, The 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyoto, Japan, 23/10/22. https://doi.org/10.1109/IROS47612.2022.9981347

Digital Object Identifier (DOI):
10.1109/IROS47612.2022.9981347

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2022

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 23. Apr. 2024

arXiv:2203.08251v2 [cs.RO] 13 Apr 2022

Flash: Fast and Light Motion Prediction for Autonomous Driving with Bayesian Inverse Planning and Learned Motion Profiles

Morris Antonello[1*], Mihai Dobre[1*], Stefano V. Albrecht[1,2], John Redford[1] and Subramanian Ramamoorthy[1,2]

Abstract— **Motion prediction of road users in traffic scenes is critical for autonomous driving systems that must take safe and robust decisions in complex dynamic environments. We present a novel motion prediction system for autonomous driving. Our system is based on the Bayesian inverse planning framework, which efficiently orchestrates map-based goal extraction, a classical control-based trajectory generator and an ensemble of light-weight neural networks specialised in motion profile prediction. In contrast to many alternative methods, this modularity helps isolate performance factors and better interpret results, without compromising performance. This system addresses multiple aspects of interest, namely multi-modality, motion profile uncertainty and trajectory physical feasibility. We report on several experiments with the popular highway dataset NGSIM, demonstrating state-of-the-art performance in terms of trajectory error. We also perform a detailed analysis of our system's components, along with experiments that stratify the data based on behaviours, such as change lane versus follow lane, to provide insights into the challenges in this domain. Finally, we present a qualitative analysis to show other benefits of our approach, such as the ability to interpret the outputs.**

I. INTRODUCTION

Motion prediction is receiving significant attention because of the need for reactive decision making in numerous robotics applications. In autonomous driving, it is the problem of inferring the future states of road users in the traffic scene. Navigating complex dynamic environments in a safe and efficient manner requires to observe, reason about and respond to relevant objects or hazards along the way. Complex reasoning [1], [2], [3], [4], such as when to perform maneuvers or low-level path planning optimizations, is not possible without making predictions about various properties of interest, such as agent future goals and trajectories.

Motion prediction is challenging because of the multitude of factors that determine future behaviour, not all of which may be explicitly modelled. Algorithms are expected to generalise to different environmental situations, such as a wide range of different layouts including merges, junctions, roundabouts etc. Combined with varying intentions and behaviours of different agents, this can result in many possible future outcomes that need to be captured. Furthermore, low-level aspects that affect the possible future trajectories, such as vehicle characteristics (e.g. dimensions, turning circle etc), type of vehicle (e.g. emergency services, transport, motorbikes etc) and driving styles (e.g. aggressive versus conservative), further increase the resulting uncertainty. Besides multi-modality and spatial uncertainty, using motion prediction within a safety critical and real-time motion planning system extends the list of requirements. System maintainability and physical realism are additional qualities of interest which are less frequently addressed [5], e.g. many state-of-the-art methods, such as [6], [7], [8], [9], are end-to-end regression-based systems which might not produce physically-feasible trajectories. As described in [10], machine learning systems can incur massive maintenance costs because of system-level anti-patterns, such as entanglement which prevents isolated improvements.

In contrast, hybrid systems, such as our proposed motion prediction system, allow to include light-weight machine learning models, isolate performance factors, replace components and interpret results while demonstrating state-of-the-art performance. Our proposed prediction system is general and can address the majority of the previously described challenges. In this paper, we evaluate our method in the highways setting, which defines a microcosm of the complex dynamics that one expects to find in everyday driving [11]. Highways include multiple lanes; space-sharing conflicts are common between the on-ramp vehicles and vehicles on the outermost lane and, similarly, during lane change and overtaking behaviours when vehicles need finding suitable gaps while maintaining safety distances.

Our contributions are two-fold:

- we present Flash, a novel hybrid motion prediction system that is based on the Bayesian inverse planning framework. It efficiently orchestrates map-based goal extraction, a classical control-based trajectory generator and an ensemble of light-weight neural networks specialised in motion profile prediction. This system models properties of interest such as multi-modality and motion profile uncertainty while providing strong guarantees that kinematic and motion profile constraints are respected;
- we evaluate the system thoroughly on the popular highway dataset - NGSIM [12], comparing with alternative multimodal methods. Flash improves the state-of-the-art trajectory error reported in [7] by 8.79 % for a 5 s prediction horizon, it guarantees kinematic and motion profile constraints by construction, and its training is 96.92 % faster than in [9]. We analyse single components such as motion profile prediction and Bayesian inference, showing that the modularity helps isolate performance factors and interpret predictions.

[*] Equal contribution

[1]Applied Research Team, Five AI, Edinburgh, United Kingdom {morris.antonello, mihai.dobre}@five.ai

[2]School of Informatics, University of Edinburgh, Edinburgh, United Kingdom

II. Related Work

The proposed system predicts the motion of the traffic participants with a Bayesian inverse planning approach [13], [14]. Previous work [2], [15] has shown that Bayesian inference combined with defined behaviours extracted from a layout of the scene generates predictions that are explainable by means of rationality; i.e. optimality given certain metrics. Another benefit of such an architecture, where Bayesian inference is the top level component, is its efficiency as reported by [16]; we also show empirically this is true.

Planning literature has a long history in using vehicle models in combination with low-level trajectory generators, see [17] for a review of algorithms for generating paths, motion profiles and trajectories. On the other hand, such methods are rarely used in prediction. [18] have combined a data-driven method with a polynomial to offer some level of smoothness in the output. This approach doesn't offer any guarantees and the resulting trajectories can still be kinematically infeasible. To address this limitation, [5] have used a vehicle model together with a path following algorithm, in particular pure-pursuit [19]. We use a similar solution in our trajectory generation component.

There are several examples of prediction methods that focus on a simplified version of the problem: motion profile prediction, see [20] for a comparison. In this space, physics-based models, e.g. constant velocity and constant acceleration, can be accurate in the short-term, but they do not consider other traffic agents. More evolved analytical solutions, such as Intelligent Driver Model (IDM) [21] to generate a car following behaviour and MOBIL [22] for lane change behaviour, have been used successfully in planning [1] and simulation for testing [23]. Still, these are deterministic methods and limited to capture only part of the context. To address these limitations, we integrate Bayesian inverse planning with an ensemble of neural networks, in particular mixture density networks [24] modelling motion profile uncertainty [7], [25], to predict the future motion profile while considering traffic context, i.e. the motion and relative spatial configuration of neighbouring agents.

Neural network based approaches have become very popular in motion prediction. Initially, these data-driven methods have been proposed without the use of maps [26]. However, utilising a map can have countless benefits, for example anchoring to the driveable area, capturing rules of the road, reducing the hypothesis space, disambiguate intentions etc. Lately, the majority of high-performing work in structured environments relies on a map in a variety of ways: from minor cues [6], [8], [9] to a rasterized version that contains the complete information including traffic signals in some of the cases [27], [28], [29], [30]. Another categorisation criterion is that these implementations are generally end-to-end trained. Our system is also heavily reliant on a map definition. Different to previous work, our networks do not consume map information at input, but rather are trained as specialised on behaviours extracted from the layout which can be reused when that behaviour is available. Such a specialisation permits independent training, tuning and inspection.

III. Problem Definition

The objective of motion prediction is to produce possible future trajectories and estimate how likely these are given the history of observations of the observable agents. We define the history of $k+1$ observations for an agent i as a sequence of coordinates (x, y), orientations θ, and velocity vectors $\mathbf{v} \in \mathbb{R}^2$: $\mathbf{h}^i = [(x^i_t, y^i_t, \theta^i_t, \mathbf{v}^i_t)]^k_{t=0}$, preceding and containing the current timestep $t = k$. Similarly, we define the predicted $\hat{\mathbf{y}}^i = [(\hat{x}^i_t, \hat{y}^i_t, \hat{\theta}^i_t, \hat{\mathbf{v}}^i_t)]^T_{t=k+1}$ and ground truth future $\mathbf{y}^i = [(x^i_t, y^i_t, \theta^i_t, \mathbf{v}^i_t)]^T_{t=k+1}$ trajectories up to the horizon T. These can be decomposed into a path, which is a sequence of N positions $[(x^i, y^i)]^N_{n=0}$, and a motion profile, which is a sequence of speeds $[s^i_t]^T_{t=k+1}$ from which higher-order derivatives such as acceleration a^i_t and jerk j^i_t can be estimated. There are two challenges that a prediction system faces: the space of possible future trajectories is continuous and the future is uncertain. The former can be handled by predicting a spatial uncertainty attached to the discrete predicted trajectories. For example, one can model this spatial uncertainty with a multivariate Gaussian capturing the position variance at each predicted future state. The latter can be addressed by producing a multimodal output, i.e. a discrete distribution over a set of predicted trajectories $P(\hat{\mathbf{y}}^i)$.

IV. Methods

A. Overview

The proposed model performs multimodal prediction by taking a Bayesian inverse planning approach [13]; it recursively compares previously predicted trajectories with observations to estimate likelihoods and computes a joint posterior distribution over goals and trajectories using Bayesian inference. An overview of the system is shown in Figure 1. It involves four main components which we will discuss in more detail: *i.)* goal and path extraction, *ii.)* motion profile prediction, *iii.)* trajectory generation and *iv.)* Bayesian inference.

B. Goal and Path Extraction

High-definition maps are useful in various aspects of self-driving; for example a map can help disambiguate the intentions of other agents [2], [15] and aid planning in challenging traffic situations [31]. We follow the OpenDrive standard [32] and implement our own layout definition for querying the geometry of roads and lanes, their properties, e.g. lane types, and how they are interconnected with each other. Given the position and orientation of an agent i at time t, we extract its possible goals $\mathbf{g}^i_t$ by exploring all lane graph traversals up to a depth. In highway situations, the immediate goals correspond to staying in the lane, or staying in the lane while maintaining the current lateral offset to the midline, or changing to a neighbour lane, or entering the highway if the agent is on the entry ramp, or exiting it if the agent is on the slow lane close to an exit ramp. We refer to this collection of

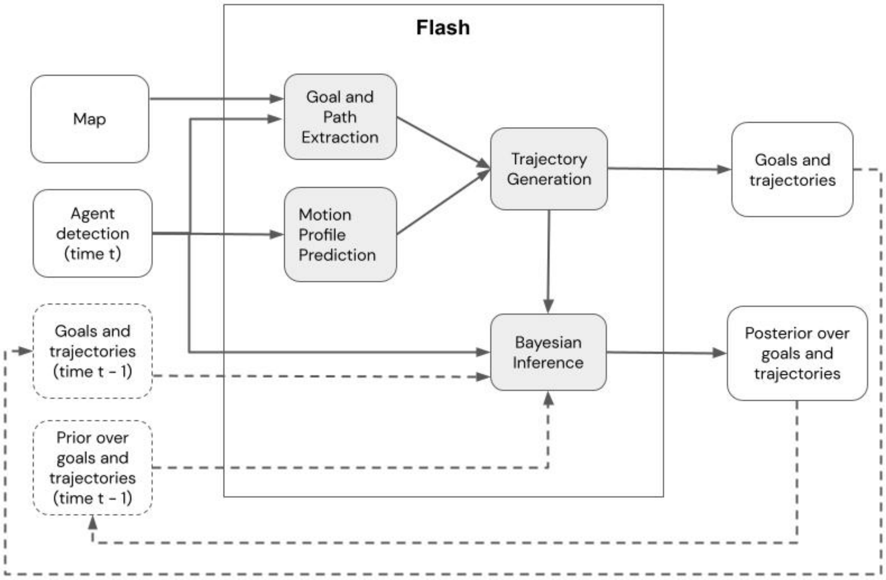

Flash
Map
Agent detection (time t)
Goals and trajectories (time t - 1)
Prior over goals and trajectories (time t - 1)
Goal and Path Extraction
Motion Profile Prediction
Trajectory Generation
Bayesian Inference
Goals and trajectories
Posterior over goals and trajectories

Fig. 1: System overview. The diagram shows the main components in grey boxes and their interconnections. Taking a Bayesian inverse planning approach, outputs becomes inputs at the next timestep, see dotted lines.

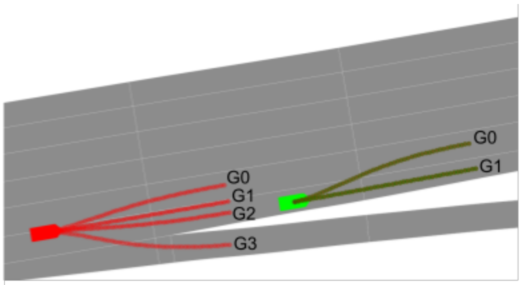


Fig. 2: Portion of the I-80 highway layout with hypothesized goals and path extraction for two agents. Agents' hypothesized goals depend on their positions on the road layout.

goals as the *hypothesized goals*. Figure 2 illustrates the use of lane midlines with an optional offset as reference paths for each goal. The goals correspond to intentions that an agent could have and our system's multimodality derives from this set of possible intentions.

C. Motion Profile Prediction

The predictive performance of different motion profile models varies strongly with the prediction horizon and traffic conditions, such as the congestion level [20]. We chose a neural network based approach for this task as these capture more contextual variations and exploit the availability of training data [20]. Due to our proposed architecture, the motion profile prediction task is narrow and well defined. In addition to this we also further split the data according to the behaviour being performed, i.e. lane follow versus change lane, and context, i.e. number of front and side agents. These aspects permit the use of an ensemble of specialised and lightweight Mixture Density Networks (MDNs) [24].

Each model in the ensemble consumes a different sized 1D feature vector $\mathbf{z}$. The features capture properties of the target agent being predicted as well as properties of front agents and side agents in the target agent's neighbourhood. These properties include agents' speeds, acceleration values, the agent class $c \in \{car, truck, motorbike, ...\}$. Lane follow models include the headway distances from the target vehicle. Lane change models consume additional features such as the neighbouring side, if the side agents are in front or behind the target vehicle, the centre to centre distance to the target vehicle and the shortest distance between the vehicles' polygons and the target vehicle. Highway entries and exits are considered lane changes. In this work, we conduct experiments with agents within a 60 m radius as in [7], including up to 3 agents in front of target vehicle and up to 3 side agents on each side. Given this, the final ensemble is made of 4 lane follow models, each specialised for a different number of front agents, and 16 lane change models, each specialised for a different number of front and side agents.

We define a target motion profile $\mathbf{y}^{mp}_{t=0} = [r_t]^N_{t=1}$ as a sequence of distances at multiple timesteps, where N is the total number of predicted timesteps. We define a predicted motion profile as $\hat{\mathbf{y}}^{mp}_{t=0}$. N is a function of horizon and delta time, respectively 5 s and 1 s, hence 5. Each MDN model learns the joint distribution $p(\mathbf{z}, \mathbf{y}^{mp})$ in the form of a multivariate Gaussian mixture model with M multivariate Gaussian functions:

$$p(\mathbf{z}, \mathbf{y}^{mp}) = \sum_{m=1}^{M} \pi_m N(\mathbf{z}, \mathbf{y}^{mp} | \boldsymbol{\mu}_m, \Sigma_m), \text{ where} \quad (1)$$

π_m, $\boldsymbol{\mu}_m$ and Σ_m are the probability, mean vector, and diagonal covariance matrix of the m^{th} multivariate Gaussian mixture component. In this work, we set M to 1 and rely on goal extraction and the ensemble to handle multimodality. Therefore, the vector of predicted means $\boldsymbol{\mu}_m$ represent the predicted motion profile $\hat{\mathbf{y}}^{mp}$. MDNs model motion profile uncertainty, predicting the mixture parameters including variances instead of single outputs. We denote each m component's prediction error as $\boldsymbol{\epsilon}_m$ and train the model with the Negative Log Likelihood (NLL) loss function:

$$NLL = -\ln \sum_{m=1}^{M} \pi_m e^{-\frac{1}{2}\boldsymbol{\epsilon}_m^T \Sigma_m^{-1} \boldsymbol{\epsilon}_m - \ln\left(\sqrt{(2\pi)^N |\Sigma_m|}\right)} \quad (2)$$

In this architecture, each MDN consists of 2 fully connected layers, 64 and 32 neurons each, with $relu$ activations. Each model is trained on a specialised dataset split, augmented with samples from the other splits. For instance, lane follow networks considering two front agents are trained with lane follow samples with at least 2 front agents and, similarly, lane change networks considering two side agents are trained with lane change samples with at least 2 side agents. Lane follow networks are trained with a learning rate of 0.001 and a batch size of 1024 for 10 epochs. Lane change networks are trained with the same learning rate and a batch size of 32 for 20 epochs.

D. Pure Pursuit Trajectory Generator

Despite their strong ability to model context, neural networks are not interpretable and can perform poorly when conditions change [33], [34]. Similarly to previous work [5], we also use pure pursuit [19] to address these challenges and generate physically feasible trajectories describing how each agent might reach any of its hypothesized goals. Pure pursuit is a path tracking algorithm that uses a controller and a vehicle model to generate a trajectory that minimises the error from the target motion profile and the target path. In our implementation, the neural networks provide the target motion profile while the target path is extracted from the map as described before. The path tracking approach imitates how humans drive; they look at a point they want to go to and control the car to reach it. As shown in Figure 3, the algorithm chooses a goal position $(x_t^{g_i}, y_t^{g_i})$ using a predefined lookahead distance parameter l_d and calculates the curvature that will move an agent from its current position to the target position while maintaining a fixed steering angle. An agent's state at time t is described using the centre position (x_t^i, y_t^i), current orientation θ_t^i and speed s_t^i. Since all agents are vehicles, we use a kinematic bicycle model to represent their motion. In addition to the state of the vehicle, we also require the distance between the rear axle and the centre position L_r^i and the wheelbase length L^i. Given a control input $u_t^i = (a_t^i, \sigma_t^i)$ at time t composed of acceleration and steering angle respectively, the Equation 3 and Figure 4 describe the motion of the vehicle. Since we use the centre position to describe the state of a vehicle, we need to compute the side slip angle $\beta_t^i = \tan^{-1}(\frac{L_r^i}{L^i} tan(\sigma_t^i))$. Finally, Δt represents the time difference between two time steps.

$$\begin{aligned} d &= s_t^i \Delta t + \frac{a_t^i \Delta t^2}{2} \\ x_{t+1}^i &= x_t^i + d cos(\theta_t^i + \beta_t^i) \\ y_{t+1}^i &= y_t^i + d sin(\theta_t^i + \beta_t^i) \\ \theta_{t+1}^i &= \theta_t^i + \frac{d}{L} cos(\beta_t^i) tan(\sigma_t^i) \\ s_{t+1}^i &= s_t^i + a_t^i \Delta t \end{aligned} \quad (3)$$

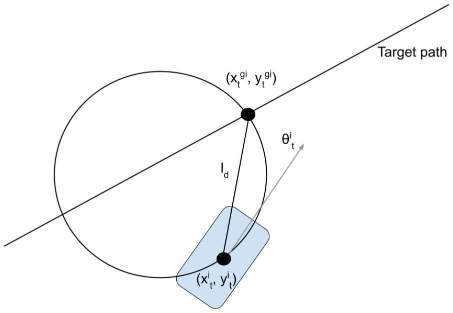


Fig. 3: Goal selection for pure pursuit trajectory generator.

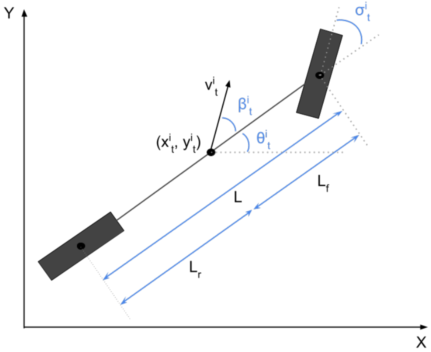


Fig. 4: Kinematic bicycle model.

The control input u_t^i is computed using a proportional controller with a gain $k_p = 2$ for each of the two components independently. The acceleration input at time t is the speed error to the future target speeds at $t' = t + 5\Delta t$. The final speed is used as target for the final $5\Delta t$ of the prediction horizon. The target speeds are retrieved from the motion profile prediction output $\hat{\mathbf{y}}^{mp}$, in particular we predict at 1 Hz and linearly interpolate them at 10 Hz. We cap acceleration using a maximum acceptable absolute acceleration $M_a = 6.0$ and absolute jerk $M_j = 10.0$ limits. Similarly, we use the orientation error for computing the steering angle and we cap it using a predefined maximum curvature $M_c = 0.3$. The orientation error $\theta_{\epsilon t}^{\ i}$ is the difference between the current vehicle orientation and the target orientation, i.e. the vehicle pointing directly to the goal position on the path. We then limit the curvature of the circle that the vehicle would describe to the maximum accepted one as shown in Equation 4.

$$\begin{aligned} \kappa &= \min\left(2\frac{sin(\theta_{\epsilon t}^{\ i})}{l_d}, M_c\right) \\ \sigma_t^i &= \tan^{-1}(\kappa L) \end{aligned} \quad (4)$$

E. Bayesian Inference

The remaining characteristics of our system are *i.)* processing the history of observations and *ii.)* performing multimodal trajectory prediction. The first of these is important when noise level is high in individual observations, while the latter is necessary to handle the uncertainty due to the unknown intention of the target agent. We achieve both by recursively consuming the observations in the history input and estimating the latent goal $g_t^i \in \mathbf{g}_t^i$ of a visible agent i via online Bayesian inference, see Equation 5. $P(g_{t=k-1}^i|\mathbf{y}_{t=k-1}^i)$ represents the previous posterior and is used as a prior in the current timestep. At $t = 0$ a uniform prior $U(g_{t=k-1}^i)$ is used.

$$\hat{P}(g_{t=k}^i|\mathbf{y}_{t=k}^i) \propto P(\mathbf{y}_{t=k}^i|g_{t=k-1}^i)P(g_{t=k-1}^i|\mathbf{y}_{t=k-1}^i) \quad (5)$$

The current implementation uses a single trajectory for a specific goal at a certain timestep. The likelihood of each goal and its corresponding trajectory is estimated by comparing previously predicted trajectories $\hat{\mathbf{y}}_{t=k-1}^i$ with the observed agent state at current timestep $t = k$. We extract the velocity direction ϕ_t^i from the velocity vector $\mathbf{v}_t^i$ and use it in our likelihood estimation as it is more robust to noise compared to using the vehicle orientation θ. Variations between predicted states and observed states are captured assuming a normal distribution on the position and velocity direction of the agent at the current timestep with fixed variances σ^2 for each term. There is a weight ω for each one of the three likelihood terms that allows us to tune their importance. The values we used are $\omega_x = 0.4$, $\omega_y = 0.4$ and $\omega_\phi = 0.15$

$$\begin{aligned} P(\mathbf{y}_{t=k}^i|g_{t=k-1}^i) &\approx \omega_x \mathcal{N}(x_{t=k}^i|\hat{x}_{t=k}^i, \sigma_x^2) \\ &\times \omega_y \mathcal{N}(y_{t=k}^i|\hat{y}_{t=k}^i, \sigma_y^2) \times \omega_\phi \mathcal{N}(\phi_{t=k}^i|\hat{\phi}_{t=k}^i, \sigma_\phi^2) \quad (6) \end{aligned}$$

In certain situations, goals can only be reached if an agent executes an uncomfortable maneuver which should imply that those goals are less likely. To capture this aspect, trajectories with lateral accelerations a_t^i above a predefined threshold max_a, 0.0 in our implementation, are penalised proportionally to the amount of violation. We integrate this as a bias term and multiply the probability of the corresponding goal with a weight that is computed using the maximum lateral acceleration $max_a^{g^i}$ of the trajectory and an exponential decay function. See Equation 7, where μ is a normalisation factor and λ, 0.5 in our implementation, is the parameter that determines the amount of punishment. Lateral acceleration values are computed as $a_t^i = \frac{{v_t^i}^2}{r_t^i}$ where r_t^i is the radius of the circle that the vehicle is currently describing given its current steer angle and assuming a kinematic bicycle model.

$$\bar{P}(g_{t=k}^i|\mathbf{y}_{t=k}^i) = \mu\hat{P}(g_{t=k}^i|\mathbf{y}_{t=k}^i) \times e^{-\lambda(max_a^{g^i} - max_a)} \quad (7)$$

An agent's goal may change in time. For example, one agent has finished a lane change and wishes to perform a lane change back in order to complete an overtake. Similarly to [13], we add a forgetting step which has the effect of smoothing the posterior, balancing recent evidence and past evidence. We mix the output of the Bayes update with a uniform distribution to get the final posterior, see Equation 8, where the parameter γ, 0.1 in our implementation, determines the amount of smoothing.

$$P(g_{t=k}^i|\mathbf{y}_{t=k}^i) = (1-\gamma)\bar{P}(g_{t=k}^i|\mathbf{y}_{t=k}^i) + \gamma U(g_{t=k}^i) \quad (8)$$

Finally, we need to account for goals changing due to the agent motion through the layout which will result in a change in the number of hypothesized goals. If a goal is no longer achievable, e.g. the agent has passed the exit ramp, then the goal is removed and its mass is distributed equally among the remaining goals. Similarly, a new goal is added with mass equal to the value that it would have if we assume a uniform distribution over the complete set of hypothesized goals. This new mass $P(g_{new}) = \frac{1}{goals\,count}$ is equally drawn from the masses of the other already existing goals. Otherwise, there is a 1:1 mapping between goals at $t = k$ and those at $t = k-1$ and our map definition makes it straightforward to perform the matching.

V. EXPERIMENTS

We evaluate our system on the Next Generation Simulation (NGSIM) dataset [12], which is seen as a standard for highway scenarios. NGSIM includes vehicle trajectory data acquired from two US highways, US-101 and I-80, using CCTV cameras and a semi-automatic annotation process. Each dataset part was captured at 10 Hz over a time span of 45 minutes and consists of 15 min segments of mild, moderate and congested traffic conditions. The dataset provides the coordinates of vehicles in UTM coordinates and a local coordinate system. We used UTM coordinates for alignment with our geo-referenced OpenDrive map annotations. As in previous related works [6], we split the datasets into train (70 %), validation (10 %) and test (20 %) based on the vehicle ID. Each vehicle from each split is chosen as the target vehicle, defining one sample. We split the trajectories of the target vehicle into segments of 8 s, where we use 3 s of history and a 5 s prediction horizon.

A. Overall System Evaluation

We compare our system with other multimodal prediction methods using two standard trajectory error metrics, Root Mean Squared Error (RMSE) and Final Displacement Error (FDE). As in [7], they are calculated by comparing the ground truth trajectory with the most likely trajectory. Lower scores are better. Table I includes the numerical comparison. Both RMSE and FDE scores of our system are lower than that of a simpler baseline, the Constant Velocity (CV) model [25], as well as that of other deep learning methods [6], [8] for all time horizons. We outperform the closest competitor [7] for most time horizons (3 s, 4 s, 5 s). Our system is comparable to the best state-of-the-art for shorter and easier horizons (1 s, 2 s, 3 s) and significantly improve over the longer, more difficult horizons (4 s, 5 s).

TABLE I: Overall system comparison on the NGSIM test set using RMSE and FDE of the most likely trajectory. Lower scores are better.

Time Horizon		1 s	2s	3s	4s	5s
RMSE [m]	CV [25]	0.76	1.82	3.17	4.80	6.70
	CSP(M) [6]	0.59	1.27	2.13	3.22	4.64
	PiP [8]	0.55	1.18	1.94	2.88	4.04
	SAMMP [7]	**0.51**	**1.13**	1.88	2.81	3.98
	Flash	0.52	1.15	**1.84**	**2.64**	**3.63**
FDE [m]	CV [25]	0.46	1.24	2.27	3.53	4.99
	CSP(M) [6]	0.39	0.91	1.55	2.36	3.39
	SAMMP [7]	**0.31**	**0.78**	1.35	2.04	2.90
	Flash	0.33	0.82	**1.34**	**1.91**	**2.62**

B. Motion Profile Prediction Analysis

The previously reported overall error can be caused by several factors. In this section, we focus on what we observed to be the most significant contributor: motion profile prediction. In addition to the previously reported dataset pre-processing steps, we split the dataset based on the observed behaviour of the agent being predicted: lane follow and lane change. We evaluate the motion profile prediction component using the RMSE and Mean Negative Log Likelihood (MNLL) errors on the predicted future distance. MNLL takes uncertainty into account [6]. We adapt the RMSE to compute it on a single dimension. The NLL is already defined in Equation 2, where the displacement error at time t for a Gaussian component m centered at $\tilde{\boldsymbol{d}}_{m,t}$ is $\boldsymbol{\epsilon}_{m,t} = \mathbf{d}_{m,t} - \tilde{\mathbf{d}}_{m,t}$. We average across the dataset to compute the MNLL value.

Figure 5 show the relative performance of different lane follow motion profile prediction models. We compared physics-based methods, Constant Velocity (CV) and Decaying Acceleration (DA), and each neural network in the ensemble. We do not report Constant Acceleration since it consistently obtains the largest errors. We model the physics-based model uncertainty at each predicted timestep using standard deviations and modelling the errors of the CV or DA assumption with a centered Gaussian distribution at each timestep. Neural networks outperform physics-based models. Considering more agents leads to lower errors since traffic dynamics, such as safe distancing, stop and go motion and motion initiation, can be modelled.

Tables II and III show the performance of the lane change networks. We report RMSE and MNLL at 5 s for brevity. The ensemble always improves over physics-based models, whose RMSE and MNLL errors are always above 7.34 m and 3.44. The performance of the lane change networks is not influenced by the number of front agents as much as the performance of the lane follow networks. Indeed, the attention of a driver performing a change lane should be on what happens in the target lane. The most difficult cases for all lane change models, including physics-based models, involve fewer number of side agents. Our interpretation is that the number of ways one can perform a change lane is reduced in congested situations, simplifying the prediction task. Furthermore, the RMSEs values vary a lot with number of side agents in comparison to the MNLL values. The cases with few agents have more variability, which is captured by our system as confirmed by the MNLL values.

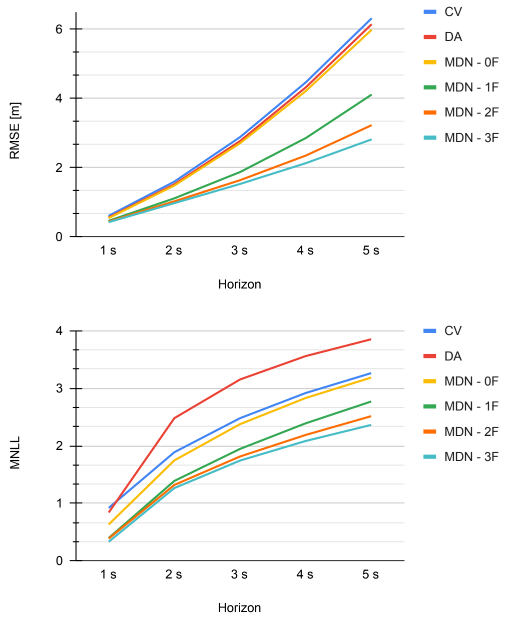


Fig. 5: RMSE and MNLL comparison of lane follow models for motion profile prediction. Models considering more front agents are more accurate.

TABLE II: Comparison of lane change motion profile prediction models on the NGSIM test set using RMSE.

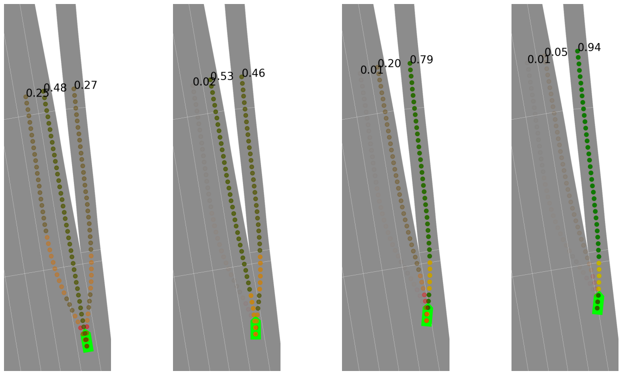
# Side # Front	0	1	2	3
0	10.05	7.54	6.45	5.13
1	9.49	7.71	6.44	5.13
2	9.50	7.05	6.15	5.06
3	8.98	8.09	6.14	5.25

TABLE III: Comparison of lane change motion profile prediction models on the NGSIM test set using MNLL.

# Side # Front	0	1	2	3
0	3.82	3.44	3.24	3.04
1	3.83	3.52	3.39	3.00
2	3.74	3.34	3.33	3.01
3	3.56	3.77	3.68	3.28

C. Qualitative Analysis

One major advantage of our system is the ability to inspect each component, allowing to debug and understand their contributions to the overall performance. Here, we show how we can debug and interpret the output of the Bayesian inference component. We also provide insight in the advantages of combining the neural networks with a classical trajectory generator.



(a) Time 0.0 (b) Time 0.2 (c) Time 0.4 (d) Time 0.6

Fig. 6: Consecutive calls of the Bayesian inference component on an example of a highway exit.

We illustrate the Bayesian inference component's process through an example of a vehicle exiting the I-80 highway, see Figure 6. For ease of visualisation, we use a constant velocity model instead of the motion profile prediction networks. We run the method with $\Delta t = 0.1$. We only show four observations and calls to the method at a $\Delta t = 0.2$ to capture a longer horizon in this illustration. The green box represents the observed position and orientation of the vehicle. The sequences of points starting from the vehicle are the predicted trajectories and these are annotated with their posterior probability. The colour of the points represent the lateral acceleration: green is low (< 2.0), yellow is medium (< 5.0) and red is high (> 5.0). Transparency is proportional to the probability of trajectories. In the first observation, the most likely behaviour is a follow lane as the change lane and exit contain some points with high lateral acceleration pushing their likelihood down. After a couple of observations, the vehicle has turned towards the exit causing the change left to be predicted as highly unlikely. Since the weight ω_ϕ is lower than the position terms, follow lane is still very probable as the vehicle has not gone far from the midline of the lane despite the relative large change in orientation. After another pair of observations, the follow lane becomes unlikely due to the large lateral acceleration values in the first few points of the corresponding trajectories. The final observation results in a large confidence in the exit behaviour, while the other behaviour have a low probability. The forgetting factor aids in quickly responding in the event of a goal change as these values do not go below a minimum.

In addition to the known unpredictability of neural networks performance, the training data can be noisy. Due to this combination, their output is likely to violate our acceleration and jerk limits. Previous work [5] has also reported that the path output of neural networks can violate the constraints of a vehicle model. These concerns are important since a prediction system that produces infeasible trajectories can negatively impact the performance of the components down the line, e.g. planning. Figure 7 shows that the combination of neural networks with kinematic and motion profile constraints can address these concerns. Even though the ground truth trajectory is not feasible, the prediction output is. Furthermore, the green trajectory is the one with the highest probability.

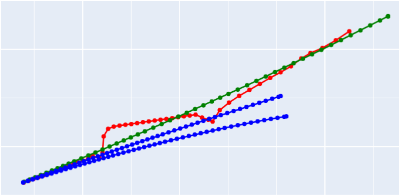


Fig. 7: Violations of kinematic and motion profile constraints. The ground truth trajectory in red presents high curvature and large variations in point distances. The predicted trajectories in blue and green (most likely) are well-formed.

We also performed a final analysis on the effects of the acceleration and jerk limits. We count the number of cases when acceleration or jerk limits have been violated, and the magnitude of violations. The calculations are performed on the predicted trajectory closest to the ground truth trajectory. On a subset of the dataset made of 20k samples, we observed 193 acceleration violations and 11.6k jerk violations that were corrected by the limits. The mean of the acceleration violation is $2.6\,m/s^2$ and standard deviation of $2.23\,m/s^2$. The mean of the jerk violation is $13.48\,m/s^3$ and standard deviation of $13.86\,m/s^3$. We also noticed that the RMSE value was at most $0.02\,m$ higher than without imposing the limits. These statistics show the importance of these limits in producing feasible trajectories without damaging the overall performance of the system.

D. Runtime Analysis

The runtime of the system was measured with a desktop PC equipped with an Intel Core i7-7800X CPU 3.50GHz over a set of 1000 samples chosen arbitrarily from the dataset. Our full system which runs on CPU is implemented in C++ and Python. It takes approximately 5.5 ms per call for each agent and we provide a breakdown of the components cost in Table IV. Given our code structure, the time reported for the Bayesian inference component includes the time for generating trajectories with the pure pursuit algorithm. The feature extraction step is currently the bottleneck as it is Python code which can be further optimised. Training the neural network ensemble in Tensorflow [35] requires approximately 32.5 minutes while other methods report training times of several hours using comparable machines and the same dataset, e.g. 17.5 hours [9] or 1 day [36].

TABLE IV: Running times [ms] per component call.

Data processing	Motion Profile Prediction	Bayesian Inference
3.3	1.8	0.4

VI. CONCLUSIONS

We present a novel motion prediction system, based on a modular architecture which involves both data-driven and analytically modelled components. We demonstrate that this achieves state-of-the-art results in the highway driving setting. The proposed system covers multiple aspects of interest, namely multi-modality, physical feasibility, motion profile uncertainty, system maintainability and efficiency.

REFERENCES

[1] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1617–1624, 2018.

[2] S. V. Albrecht, C. Brewitt, J. Wilhelm, B. Gyevnar, F. Eiras, M. Dobre, and S. Ramamoorthy, "Interpretable goal-based prediction and planning for autonomous driving," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1043–1049, IEEE, 2021.

[3] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "A two-stage optimization-based motion planner for safe urban driving," *IEEE Transactions on Robotics*, 2021.

[4] N. Rhinehart, J. He, C. Packer, M. A. Wright, R. McAllister, J. E. Gonzalez, and S. Levine, "Contingencies from observations: Tractable contingency planning with learned behavior models," in *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pp. 13663–13669, IEEE, 2021.

[5] H. Girase, J. Hoang, S. Yalamanchi, and M. Marchetti-Bowick, "Physically feasible vehicle trajectory prediction," *arXiv preprint arXiv:2104.14679*, 2021.

[6] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1468–1476, 2018.

[7] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-head attention for multi-modal joint vehicle motion forecasting," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9638–9644, IEEE, 2020.

[8] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "Pip: Planning-informed trajectory prediction for autonomous driving," in *European Conference on Computer Vision*, pp. 598–614, Springer, 2020.

[9] B. Mersch, T. Höllen, K. Zhao, C. Stachniss, and R. Roscher, "Maneuver-based trajectory prediction for self-driving cars using spatio-temporal convolutional networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4888–4895, IEEE, 2021.

[10] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine learning: The high interest credit card of technical debt," 2014.

[11] A. R. Srinivasan, M. Hasan, Y.-S. Lin, M. Leonetti, J. Billington, R. Romano, and G. Markkula, "Comparing merging behaviors observed in naturalistic data with behaviors generated by a machine learned model," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 3787–3792, IEEE, 2021.

[12] J. Colyar and J. Halkias, "Us highway 101 dataset," *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, pp. 27–69, 2007.

[13] C. L. Baker, R. Saxe, and J. B. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 113, no. 3, pp. 329–349, 2009.

[14] M. Ramírez and H. Geffner, "Probabilistic plan recognition using off-the-shelf classical planners," in *24th AAAI Conference on Artificial Intelligence*, pp. 1121–1126, 2010.

[15] J. P. Hanna, A. Rahman, E. Fosong, F. Eiras, M. Dobre, J. Redford, S. Ramamoorthy, and S. V. Albrecht, "Interpretable goal recognition in the presence of occluded factors for autonomous vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7044–7051, IEEE, 2021.

[16] Y. Luo, P. Cai, Y. Lee, and D. Hsu, "Gamma: A general agent motion model for autonomous driving," *arXiv:1906.01566 [cs]*, 2022.

[17] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[18] T. Buhet, É. Wirbel, A. Bursuc, and X. Perrotton, "PLOP: probabilistic polynomial objects trajectory prediction for autonomous driving," in *CoRL*, vol. 155 of *Proceedings of Machine Learning Research*, pp. 329–338, PMLR, 2020.

[19] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[20] S. Lefèvre, C. Sun, R. Bajcsy, and C. Laugier, "Comparison of parametric and non-parametric approaches for vehicle speed prediction," in *2014 American Control Conference*, pp. 3494–3499, IEEE, 2014.

[21] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[22] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[23] J. Bernhard and A. C. Knoll, "Risk-constrained interactive safety under behavior uncertainty for autonomous driving," in *IV*, pp. 63–70, IEEE, 2021.

[24] C. M. Bishop, "Mixture density networks," 1994.

[25] J. Mercat, N. E. Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Kinematic single vehicle trajectory prediction baselines and applications with the ngsim dataset," *arXiv preprint arXiv:1908.11472*, 2019.

[26] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.

[27] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, pp. 86–99, 2019.

[28] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding HD maps and agent dynamics from vectorized representation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 11522–11530, 2020.

[29] F. Chou, T. Lin, H. Cui, V. Radosavljevic, T. Nguyen, T. Huang, M. Niedoba, J. Schneider, and N. Djuric, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," in *IEEE Intelligent Vehicles Symposium, IV 2020, Las Vegas, NV, USA, October 19 - November 13, 2020*, pp. 1655–1662, IEEE, 2020.

[30] L. Zhang, P. Su, J. Hoang, G. C. Haynes, and M. Marchetti-Bowick, "Map-adaptive goal-based trajectory prediction," in *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA* (J. Kober, F. Ramos, and C. J. Tomlin, eds.), vol. 155 of *Proceedings of Machine Learning Research*, pp. 1371–1383, PMLR, 2020.

[31] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *CoRR*, vol. abs/1708.06374, 2017.

[32] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks," in *Proc. of the Driving Simulation Conference Europe*, pp. 231–242, 2010.

[33] A. Filos, P. Tigkas, R. Mcallister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?," in *International Conference on Machine Learning*, vol. 119, pp. 3145–3153, PMLR, Nov. 2020.

[34] H. Pulver, F. Eiras, L. Carozza, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "PILOT: Efficient planning by imitation learning and optimisation for safe autonomous driving," *arXiv:2011.00509 [cs]*, Nov. 2020.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[36] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," *Advances in Neural Information Processing Systems*, vol. 32, pp. 15424–15434, 2019.