PLOP: Probabilistic poLynomial Objects trajectory Prediction for autonomous driving

Thibault Buhet[1], **Emilie Wirbel**[1,2], **Andrei Bursuc**[2] **and Xavier Perrotton**[1]
[1] Valeo Driving Assistance Research, [2] Valeo.ai
name.surname@valeo.com

Abstract:
To navigate safely in urban environments, an autonomous vehicle (*ego vehicle*) must understand and anticipate its surroundings, in particular the behavior and intents of other road users (*neighbors*). Most of the times, multiple decision choices are acceptable for all road users (e.g., turn right or left, or different ways of avoiding an obstacle), leading to a highly uncertain and multi-modal decision space. We focus here on predicting multiple feasible future trajectories for both ego vehicle and neighbors through a probabilistic framework. We rely on a conditional imitation learning algorithm, conditioned by a navigation command for the ego vehicle (e.g., “turn right”). Our model processes ego vehicle front-facing camera images and bird-eye view grid, computed from Lidar point clouds, with detections of past and present objects, in order to generate multiple trajectories for both ego vehicle and its neighbors. Our approach is computationally efficient and relies only on on-board sensors. We evaluate our method offline on the publicly available dataset nuScenes, achieving state-of-the-art performance, investigate the impact of our architecture choices on online simulated experiments and show preliminary insights for real vehicle control.

1 Introduction

Operating self-driving cars in the real world is a highly challenging endeavor. The vehicle must interact safely with other road users and stick to the limits and rules of the road. Most human drivers do this effortlessly through visual perception and driving experience allowing them to quickly scan surroundings and perform various micro-decisions needed in traffic. For a vehicle, we consider that over a short enough time interval, the world can be approximated through a snapshot of the current scene in which agents will take actions. The static environment might be hard to understand because of the current topology (*e.g.*, complex intersections) or unusual circumstances (*e.g.*, work zone, absent/inconsistent markings, etc.). Other agents might also be tough to handle because they are out of the autonomous vehicle's control. They can be of very different types (*e.g.*, pedestrian, cyclist, car, truck, robot, etc.) and can be involved in unusual situations that should be managed (*e.g.*, pedestrian on the highway, animal crossing the road, emergency vehicle intervention, etc.). The autonomous vehicle, which we designate as the *ego vehicle* in the following, will also have a destination to reach and will be guided by either a target position or simply by a high-level goal such as keeping its lane, turning at an intersection etc. We refer to the other vehicles close to the ego vehicles as *neighbors*.

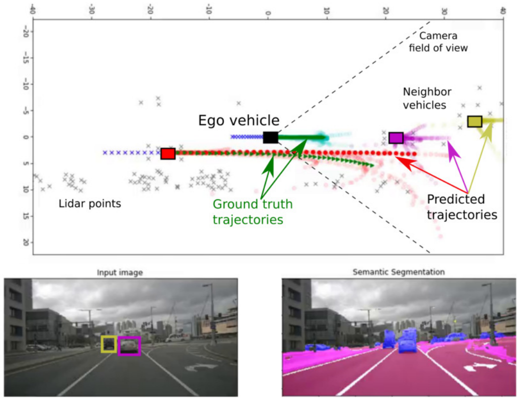


Figure 1: **Qualitative example of trajectory predictions on a test sample from nuScenes dataset.** PLOP processes front camera image, Lidar points, 2s of ego vehicle past track and 1s of neighbor vehicles past tracks to perform trajectory prediction over the next 4s. PLOP handles uncertainty and variability by predicting vehicle trajectories as a probabilistic Gaussian Mixture, constrained by a polynomial formulation.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

Driving algorithms must follow a correct behavior under these varying and evolving parameters. *Trajectory prediction* attempts to mitigate this problem. Here, we define trajectory prediction as the prediction of future positions of all agents in the scene, ego and neighbor vehicles, over a fixed period of time. Typical real-world driving environments involve high uncertainty and multiple possible outcomes. In such contexts, a trajectory prediction algorithm must consider diverse potential future paths for each agent and estimate a measure of uncertainty on the predictions. We assume the ego vehicle has access to information describing the current scene, *e.g.*, sensor data, object bounding boxes and/or past positions (tracks), semantic segmentation maps, depth information, etc. The input representation of the scene is essential and can vary significantly across related methods: using only raw sensor data [1, 2], adding object detections and past positions [3–8], assuming almost complete explicit information about the scene (semantic, lanes, detections, map, etc.) [9]. Similar to previous works, we opt for using raw sensor data (front camera and Lidar) and object detections that can be computed on the fly on the vehicle. We forego using HD maps, which albeit useful, bring a significant cost for updating and processing, while still having blind spots, *e.g.*, recent road works.

To this effect, we propose an architecture for multi-modal trajectory prediction, dubbed PLOP (**P**robabilistic po**L**ynomial **O**bjects trajectory **P**rediction). PLOP predicts multiple plausible trajectories separately for ego vehicle and neighbors, in a single step fashion, through a formalism based on Mixture Density Networks (MDN) [10]. Makansi et al. [11] showed that MDNs can provide stable results with careful training and sampling, in a pedestrian future position prediction context. Here, MDNs coupled with a polynomial formulation for trajectories naturally deal with the ambiguity of agents' behaviours offering multiple realistic futures. PLOP leverages scene information from sensor data (Lidar and front camera) and detection tracks of nearby vehicles, which can be computed from side cameras and/or Lidar point clouds. We show that adding an auxiliary perception task, e.g. semantic segmentation on front image, contributes to further improvement of prediction accuracy through injection of scene layout information in the internal representations of the network. In Figure 1 we illustrate typical input data and results for our method. Additional results can be found in the supplementary material video[1].

The main contributions of this work are the following: **(1)** We propose a single-shot, anchor-less trajectory prediction method, based on MDNs and polynomial trajectory constraints, relying only on on-board sensors (no HD map requirement). The polynomial formulation ensures that the predicted trajectories are coherent and smooth, while providing more learning flexibility through the extra parameters. We find that this mitigates training instability and mode collapse that are common to MDNs [12]; **(2)** We extensively evaluate PLOP and show its effectiveness across datasets and settings. We conduct a comparison showing the improvement over state-of-the-art PRECOG [3] on the public dataset nuScenes [13]; **(3)** We study closed loop performance for the ego vehicle, on simulation and with preliminary insights for real vehicle control.

2 Related Work

In contrast to explicit mathematical models [14], data-driven methods offer the advantage that the models no longer need to be specified explicitly. Recent deep learning methods address this problem mostly through Imitation Learning (IL). We focus here on a subset of IL, behavioral cloning (BC), a supervised learning approach where expert samples are used as ground truth. Several datasets such as nuScenes [13] or Waymo [15] enable the development of BC algorithms for trajectory prediction. We note that for dealing with multiple possible outcomes, there is practically no dataset offering such situations (except for a recent synthetic pedestrian dataset [16]), further increasing the difficulty of this endeavor. Most of the existing literature assumes that the tracked past positions are known, often by accumulating past object detections. Approaches are usually recursive and attempt to encode the relationships between agents of the road.

There are two main fields of trajectory prediction methods: predicting vehicle behavior, or pedestrians. In most pedestrian prediction problems, the scene is observed from a fixed position with actors entering and leaving. A recent survey by Rudenko et al. [17] presents of very complete panorama of the state of the art. Since precursor work such as SocialLSTM [6], later extended for vehicle applications by CS-LSTM [5], pedestrian oriented literature often focuses on interactions between agents. Pedestrian trajectory predictions are often based on graph structures such as the recent Trajectron [18, 19] or Social-BiGAT [20]. Here, we will concentrate on vehicle trajectory prediction.

[1] https://youtu.be/94FwahFmc5A

For road users, the scene is usually centered around an ego vehicle. There is a fundamental distinction between predicting the trajectory for the ego vehicle, and predicting the neighbors. In the last case, there is no information available on the goal of the vehicles, and they cannot be controlled. Here, benchmarks such as Argoverse [21] or nuScenes [13] based on offline datasets can be used, although global metrics do have their limits because they do not distinguish between critical use cases, *e.g.* braking at a red light, and non-critical use cases, *e.g.* lane keeping without obstacle. In this context, graph representations can also be leveraged for better modeling of interactions between agents for trajectory prediction [22–24]. SpAGNN [25] extend graph representations with detections of different objects in the scene and generate trajectories through iterative message passing instead of a classic RNN-based decoder. In most cases, it is assumed that neighbors are detected tracked, but some approaches, such as PnPNet [26] learn jointly perception, object track association, and future trajectory prediction from HD maps and Lidar frames.

Our work is focused on predicting the ego vehicle trajectory, with a closed loop control application in mind, along with the neighbors trajectories. Some approaches such as ChauffeurNet [9] use a high-level scene representation (road map, traffic lights, speed limit, route, dynamic bounding boxes, etc.). More recently, MultiPath [27] uses trajectory anchors, used in one-step object detection, extracted from the training data for ego vehicle prediction. Hong et al. [28] use a high level representation which includes some dynamic context. Cui et al. [12] produce multi-modal Gaussian representations, however contrary to our work the modes are constrained by the possible manoeuvres, e.g. turn at an intersection.

In contrast, we choose to leverage also low level sensor data, here Lidar point clouds and camera image. In that domain, recent approaches address the variation in agent behaviors by predicting multiple trajectories, often in a stochastic way. Many works, *e.g.*, PRECOG [3], R2P2 [4], MFP [29], SocialGAN [30] and others [31], focus on this aspect through a probabilistic framework on the network output or latent representations, producing multiple trajectories for ego vehicle, nearby vehicles or both. Phan-Minh et al. [32] generate a trajectory set, then classify correct trajectories. Marchetti et al. [33] generate multiple futures from encodings of similar trajectories stored in a memory. Ohn-Bar et al. [34] learn a weighted mixture of expert policies trained to mimic agents with specific behaviors. In PRECOG, Rhinehart et al. [3] advance a probabilistic formulation that explicitly models interactions between agents, using latent variables to model their plausible reactions, with the possibility to pre-condition the trajectory of the ego vehicle by a goal.

Compared to related works, an advantage of our approach is that we rely on on-board raw sensor data, differently from others relying on high precision maps and GPS positioning. PLOP is trainable end-to-end from imitation learning, where data is relatively easier to obtain. PLOP is computationally efficient during both training and inference as it predicts trajectory coefficients in a single step, without requiring a RNN-based decoder. The polynomial function trajectory coefficients eschew the need for anchors [27], whose quality can vary across datasets.

3 Network Structure

Our main goal is to predict future ego vehicle trajectories along with the neighbors trajectories. To this end, we use a single multi-input multi-output neural network to produce a probabilistic representation of these trajectories. In this section, we first describe the input structure, then the architecture of the network, and finally the formulation of the outputs and the associated losses.

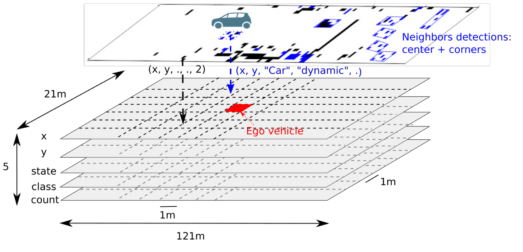


Figure 2: **Structure and construction of the input bird-eye view (BEV):** vehicles (corners and center) and Lidar points are reprojected in the ego vehicle coordinate system. The BEV ranges from -60.5m to $+60.5$m on the longitudinal axis and -10.5m to $+10.5$m on the lateral axis using 1m× 1m cells. Each cell contains if applicable the position (resp. mean position) (x,y) of neighbor vehicle (resp. Lidar points), state (parked, stopped, dynamic), class (2wheels, car, truck) and Lidar points count. Each BEV is represented by a $[121 \times 21 \times 5]$ tensor.

3.1 Inputs: Past Trajectories, Camera and Lidar

We assume all input data is sampled at 10Hz. We note N the maximum number of neighbors that are considered as input ($N = 10$ in this work). Past inputs are accumulated over 2s. For simplicity, we only consider vehicles as neighbors, excluding other road users such as pedestrians or bikes from the input object detections.

Past trajectories are represented as time series, over the last 2s for the ego vehicle and the neighbors. We use a frontal RGB camera (with a field of view of 70° on nuScenes).The metric surroundings (Lidar point cloud and neighbors detections) are projected into a grid bird-eye view (BEV), inspired by [2, 3, 35], as summarized in Figure 2. The ground Lidar points are filtered out for unequivocal obstacle representation. This map is accumulated over the past 2s, *i.e.*, 20 frames.

Finally, we use a navigation command input for the conditional part of our network [36]. There are four navigation commands, one when the ego vehicle is far away from an intersection: *follow* and three when the ego vehicle is close to an intersection: *left, straight, right*.

3.2 Network Architecture

We illustrate the structure of our neural network in Figure 3 and detail its design here. The architecture has two main sections: an encoder to synthesize information and a predictor to process it. Overall, the structure of the neural network is designed to guide latent representations towards encompassing relevant information about the environment and its geometry.

The encoder consists of three parts: the front camera image of the ego vehicle is encoded by a VGG16 [37] network, the BEVs are encoded by a CNN with 3D convolutional layers, and the trajectories are encoded using an LSTM layer [38]. The LSTM weights are shared between the neighbors, while the ego vehicle has its own LSTM weights. The predictor can also be divided in three parts. First, we consider an auxiliary U-net semantic segmentation decoder [39] to inject an awareness of scene layouts and human interpretability in the network features and to improve learning stability of the entire architecture. Note that we are not actually interested in the semantic segmentation prediction itself, but rather use it as pretext task to enable the network to learn useful features. We can remove this module after training. Then, the neighbors' trajectories prediction takes as input, for each vehicle, a concatenation of the BEV encoding and the considered neighbor vehicle past trajectory encoding and apply 3 fully connected (FC) layers to output a multivariate Gaussian mixture for a fixed number of possible trajectories K. We train this module with the negative log-likelihood loss [10] (see Section 3.3). The FC weights are shared between all neighbors vehicles. Finally the ego vehicle trajectory prediction uses the same principle as for the neighbors vehicles while adding the image encoding as an input and a conditional dimension to the FC layers. The 3 FC layers are replaced by 4×3 FC layers conditioned by the 4 navigation commands.

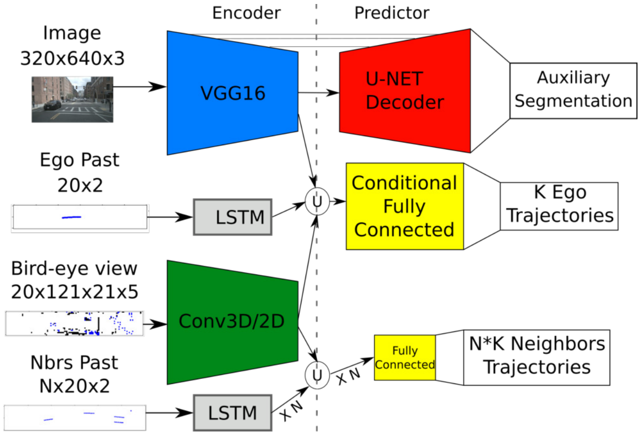


Figure 3: **Neural network architecture for PLOP:** the encoder is on the left and the predictor is on the right. ∪: Concatenation. PLOP is compatible with an arbitrary number of neighbors N, though for implementation constraints maximal N is fixed to 10 for training. PLOP architecture is presented in greater details in Section 1 of the supplementary material.

We note that each vehicle prediction does not have direct access to the sequence of past positions of other vehicles. The bird's eye view (BEV) encoding implicitly encapsulates the interactions between vehicles. It allows our architecture to be agnostic to the number of considered neighbors, an advantage compared to methods like Social-LSTM or PRECOG. A bird's eye view is also a lightweight representation for point cloud data, and can be easily complemented with additional information as described in Figure 2.

3.3 Network Outputs and Training

Trajectory prediction Our network has two main outputs, the possible future trajectories for both ego vehicle and nearby vehicles. For each vehicle we want to predict multiple trajectories to mimic the stochasticity of human behavior and cope with ambiguities in a given situation. We want to predict a fixed number K of possible trajectories for each vehicle, and associate them to a probability distribution over x and y (x is the longitudinal axis, y the lateral axis, pointing left). For the ego vehicle, we estimate the probability distribution conditioned by the navigation command c. For simplification, we consider that x and y are independent. We make the following assumptions and simplifications: **(1) The distribution is expressed only at fixed points**, sampled at a fixed rate in the future (indexed by $t \in [0,T]$). We forecast trajectories over a 4s horizon, *i.e.*, $T = 40$ here; **(2)** For x and y respectively, for each point in time, **the distribution is modeled by a mixture with K Gaussian components** $\mathcal{N}\left(\hat{\mu}_{k,x}(t), \sigma_{k,x,t}\right)$. By default, we use $K = 12$, following the choice of Rhinehart et al. [3]; **(3) The mixture weights π_k are shared** for all sampled points belonging to the same trajectory (over x and y). This makes it possible to associate a weight to a whole trajectory and reduces the number of parameters; **(4)** For x and y respectively, for each component, the means of the distribution $\hat{\mu}_{k,x}(t)$ (resp. $\hat{\mu}_{k,y}(t)$) are **generated polynomials of degree 4 of time**, following Buhet et al. [7]. We denote the coefficients of these polynomials $\boldsymbol{a}_{k,d,x}$ (resp. $\boldsymbol{a}_{k,d,y}$), the constant coefficient is set to zero. This reduces the number of parameters and constrains the dynamics of the points produced by the trajectories.

Finally, for each sampled point in the future at time $t \in [0,T]$, the probability density function for the point position, $p(x,t)$ is expressed as follows, for each vehicle ($p(\cdot|c)$ for the ego vehicle, for each command): $p(x,t) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\hat{\mu}_{k,x}(t), \sigma_{k,x,t}\right)$ where $\hat{\mu}_{k,x}(t) = \sum_{d=0}^{d=3} a_{k,d,x} t^{4-d}$. Expressions for $p(y,t)$ are analogous.

The outputs of the network are: the vectors $\boldsymbol{a}_{k,x}$ (resp. $\boldsymbol{a}_{k,y}$), of dimension d, the variance $\sigma_{k,x,t}$ (resp. $\sigma_{k,y,t}$) and global trajectory coefficients π_k for each vehicle, and for each command c in the case of the ego vehicle. In a nutshell, this representation can be interpreted as predicting K trajectories, each associated with a confidence π_k, with sampled points following a Gaussian distribution centered on $\left(\hat{\mu}_{k,x}(t), \hat{\mu}_{k,y}(t)\right)$ (generated and constrained by polynomials) and with variance $\left(\sigma_{k,x,t}, \sigma_{k,y,t}\right)$. We illustrate the idea through a simplified example in Figure 4.

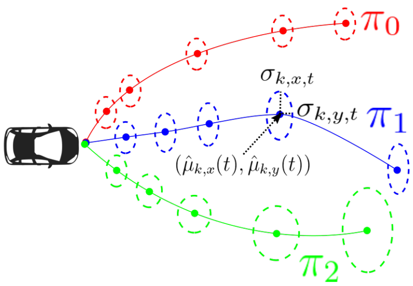


Figure 4: **Simplified representation of the trajectory prediction model** ($K = 3, T = 5$)**.** Each sampled Gaussian component is represented by an ellipse of center $\left(\hat{\mu}_{k,x}(t), \hat{\mu}_{k,y}(t)\right)$ and shape $\left(\sigma_{k,x,t}, \sigma_{k,y,t}\right)$.

Auxiliary semantic segmentation output The segmentation module predicts over 7 typical classes for visual perception-based driving (*Void, Vehicle, Pedestrian, Traffic Sign/Signal, Lane Marking, Road, Sidewalk*). Our objective here is to make sure that the features of the image encoder comprise additional information about the road position and availability, the applicability of the traffic rules (traffic sign/signal), the vulnerable road users (pedestrians, cyclists, etc.) position, etc. This information is useful for trajectory prediction and can potentially contribute to the explainability of the model.

Loss To train the network, the main objective of predicting the trajectories distribution is achieved by minimizing negative log-likelihood (NLL) over all sampled points of the ground truth ego and neighbor vehicles trajectories (see Equation 1, where p_{ego} is the distribution for the ego vehicle and p_n the distribution for the n-th neighbor). To improve the results for the lateral component of the trajectories, which is harder to predict, we add a weight α to the loss related to the y coordinate ($\alpha = 3$ in our tests). In the end, the loss $L_{\texttt{NLL}}$ is expressed by Equation 1, where $(\overline{x}(t), \overline{y}(t))$ represents the ground truth position of the ego vehicle at time t, respectively $(\overline{x}_n(t), \overline{y}_n(t))$ for the n-th neighbor, and c is the current navigation command for the ego vehicle:

$$L_{\texttt{NLL}}(c) = -\sum_{t=1}^{T} \left(\log\left(p_{\texttt{ego}}(\overline{x}(t)|c)\right) + \alpha \log\left(p_{\texttt{ego}}(\overline{y}(t)|c)\right) + \sum_{n=1}^{N} \left[\log\left(p_n(\overline{x}_n(t))\right) + \alpha \log\left(p_n(\overline{y}_n(t))\right) \right] \right). \tag{1}$$

For the auxiliary semantic segmentation module, we use the cross-entropy loss $L_{\texttt{seg}}$. Since the considered datasets do not offer both semantic segmentation and object tracking annotations, we train the network by mixing samples containing either one of the objectives, and backpropagate using either $L_{\texttt{NLL}}$ or $L_{\texttt{seg}}$. For training, we used Radam optimizer [40], which is an improvement of Adam, with learning rate 10^{-5}. The mini-batch size was set to 8 and each mini-batch was split in two: 4 semantic segmentation task inputs and 4 trajectory task inputs.

4 Experiments on Offline Data

4.1 Metrics

We use two main metrics: minMSD (minimum Mean Squared Deviation) as in [3, 4, 41] and minADE (minimum Average Displacement Error): minMSD $= \frac{1}{TN}\sum_{n=1}^{N}\min_{k\in K}\sum_{t=1}^{T}||\mu_{n,k}(t)-\mu_n^*(t)||^2$ and minADE $= \frac{1}{TN}\sum_{n=1}^{N}\min_{k\in K}\sum_{t=1}^{T}||\mu_{n,k}(t)-\mu_n^*(t)||$.

These metrics consider only the best predicted trajectory for the selected metric regardless of its confidence score. The purpose is to avoid penalizing valid possible future trajectories for each agent when it does not correspond to the actual recorded path. For example, penalizing a prediction that turns right with high confidence and goes straight with smaller but still high confidence when the vehicle went straight during the recording is not appropriate. minMSD will emphasize high errors while minADE is neutral regarding the error magnitude.

4.2 Comparison to State of the Art

We conduct all experiments on two recent datasets: nuScenes [13] and A2D2 [42]. nuScenes is used as the trajectory prediction dataset, it consists of 850 scenes of 20s driven in Boston and Singapore. The 2Hz annotation of the scenes (tracked bounding boxes) is extended to 10Hz using interpolation. We use the nuScenes train set as a train + validation set and the nuScenes validation set as a test set. We use only a small part of the Audi A2D2 dataset, namely image semantic segmentation data. We take the 41,280 annotated frames and reduce the available classes to the 7 classes we consider. At training time, the nuScenes examples are used only with trajectory losses and the examples from Audi with semantic segmentation loss. We evaluate our results only on the nuScenes dataset since the semantic segmentation task is auxiliary. The goal of semantic segmentation here is not accuracy, but rather to implicitly inject awareness about the scene layout into the network.

	minMSD (m^2)					
Number of agents	1	2	3	4	5	6+
DESIRE-plan [41]	2.26	6.64	6.18	9.20	8.52	-
ESP [3]	1.86	2.37	2.81	3.20	4.36	-
PRECOG [3]	**0.149**	2.32	2.65	3.16	4.25	-
PLOP	1.89	**1.97**	**2.39**	**2.74**	**2.84**	2.53

Table 1: Comparison with published results of DESIRE-plan, ESP and PRECOG from [3] (results from their Table II, with a fixed 5 agents training), over minMSD metric.

For comparison, we use the results published in PRECOG [3], with both ESP and PRECOG itself, and also report some of their baselines for comparison. We distinguish the evaluation of the trajectory prediction regarding the number of agents in the current scene, from one agent (ego vehicle only) up to 10 agents (9 neighbors). There is information on the goal of the ego vehicle but not for the others. Note that in our setup, the goal is given as a high-level navigation command, whereas PRECOG gives the goal as the target position 4s ahead, which puts PLOP at a slight disadvantage here. The comparison is fairer for neighbor trajectories, and the performance is relevant as they are by definition open loop.

Results are presented in Table 1. The comparison is made using minMSD, as reported in PRECOG [3]. We note that ESP and PRECOG are not fully flexible concerning the number of agents considered, we choose to compare to the published results trained for a maximum of 5 neighbors. To reach maximum performance using ESP and PRECOG, it is required to train specifically for each situation: 1 agent, 2 agents, etc. For a fair comparison, all results presented in Table 1 are computed for $K = 12$. Considering ego trajectory prediction only (1 agent), PRECOG outperforms our architecture significantly. This is expected, since the navigation goal is given as an exact target position, compared with our high-level navigation command. However, PLOP performs as well as ESP, even if ESP has access to the same goal as PRECOG. For situations with up to 4 agents, PLOP slightly outperforms others Finally, for 5 agents or more, our method outperforms by a large margin all other approaches. This shows that, unlike compared methods, our model is more robust to the varying number of neighbors: note that with 3 or more agents, the minMSD metric, regarding the neighbors vehicles only, varies little. This result might be explained by our interaction encoding which is robust to the variations of N using only multiple BEV projections and our non-iterative single step trajectory generation. We believe this is a valuable results as running one model per situation in a real car poses major technical challenges. The metric improves while going from 5 to 6 or more agents, such crowded situations often involve slow or stopped vehicles which are easier to predict.

4.3 Finer analysis on metrics and architecture

Table 2 reports minADE. We notice that minADE values for the different situations are overall close to each other. We note that, due to the square factor, small variations in the minADE metric can induce higher variations in the minMSD metric. We also note that low minADE value along high minMSD values tends to represent a distribution with few very incorrect trajectories and a great number of correct trajectories, whereas high minADE value along low minMSD value tends to represent a distribution constituted of a lot of moderately correct trajectories.

Number of agents	1	2	3	4	5	6+
minADE	0.85	0.84	0.90	0.94	0.90	0.76

Table 2: Comparison of minADE (m) metric according to the number of agents.

The number of predicted trajectories K is fixed in the network architecture and we need to estimate the best value for this parameter. The presented results so far used $K=12$ for a fair comparison with other methods, however in other works up to one hundred or more trajectories per agent are predicted [41]. Observed results tend to show that increasing K improves the metrics results. However, to keep a reasonable number of parameters in the output layer considering our trajectory generation we keep K under 12. The results of this hyperparameter study are presented in Table 3. It confirms that adding more trajectory components improves the performance, although we can see that the contribution diminishes for $K \geq 6$.

	Ego vehicle		Neighbor vehicles	
	minMSD	minADE	minMSD	minADE
$K=12$	**1.65**	**0.79**	**2.82**	**0.88**
$K=6$	2.28	0.90	3.77	1.05
$K=3$	2.55	0.93	6.81	1.44
$K=1$	4.13	1.26	9.91	1.83

Table 3: Influence of number of components for the Gaussian mixture K over the metrics.

5 Closed Loop on Real Vehicle

5.1 Setup

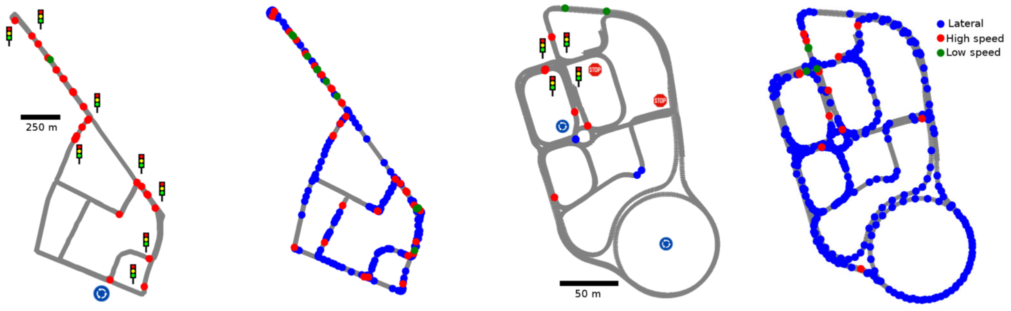
To alleviate the limitations of IL, it is necessary to use multiple data augmentation techniques. The goal is to produce scenarios outside of the training data, such as, lateral shift, lane departure, breaking anticipation or recovery. The methods are out of the scope of this work, but are detailed extensively in the literature [7, 9, 36, 43]. We also add noise in the vehicles positions (ego and neighbors) to improve our method robustness following a similar approach to [7, 44]. We use a test vehicle equipped with high precision GPS, Lidar and surround-view cameras (replacing the frontal camera from nuScenes). GPS tracks are used as ground truth for the ego vehicle trajectories. For neighbors, we use an off-the-shelf vehicle detector [45] and tracker, running at 25Hz, which is not manually annotated and thus contains noise. The dataset contains two types of scenes: a busy urban driving scenario and a test track covering multiple urban situations (traffic light, roundabout, yield, stop, etc) with only one other vehicle. The characteristics of this internal dataset are summarized in Table 4 and detailed in the supplementary material. The prediction horizon is reduced to 2s ($T=20$) to correspond better to the control algorithm.

Scenario	$\overline{S}$	$\overline{N}$	Σ_N	Σ_D	Σ_T
Urban	19	5.6	13619	102	5h20
Track	17	0.5	369	43	2h26

Table 4: Datasets characteristics: mean speed $\overline{S}$ (kph), mean number of neighbors per frame $\overline{N}$, total tracked neighbhors Σ_N, total distance Σ_D (km) and time Σ_T

5.2 Evaluation

A closed loop setup is critical to evaluate trajectory prediction, especially for a behavioral cloning approach (see [46]). For example, on an offline benchmark, a trajectory prediction that entirely ignores traffic lights would be penalized only in the few frames where the vehicle braking should be anticipated. There are two possible ways to evaluate trajectory prediction for vehicle control: simulation, and running tests on the actual target vehicle. Amini et al. [47] present a thorough example: they use a data-driven simulator to train and evaluate driving policies offline, then for online behavior near-crash recoveries, trajectory variance and distance to mean trajectories are reported over a 3km test track. Our online experiments are conducted using a in-house simulator where we approximate real driving using a bicycle model for ego displacement and sensor transformations [43, 48]. We have also performed preliminary tests in a real vehicle with the ego trajectory prediction on the urban test track. More information about our online tests is presented in the supplementary material.



(a) PLOP, urban (b) Const. vel., urban (c) PLOP, track (d) Const. vel., track

Figure 5: Closed loop error locations for urban and track test data, visualized for PLOP and constant velocity baseline. We note that braking behind a vehicle can induce multiple high speed errors and stack multiple red dots on the same location. Points of interest (traffic lights, roundabout, stop signs) are highlighted on the map.

	Urban			Track		
Failures	*lat*	*high*	*low*	*lat*	*high*	*low*
PLOP	0	40	1	4	7	2
PLOP no seg.	4	80	0	7	18	0
Const. vel.	185	35	10	381	14	4
MLP	431	44	15	302	53	12

Table 5: Quantitative comparison of failure cases, on the test set: PLOP, PLOP without semantic segmentation auxiliary loss (no seg.) vs two baselines, one assuming constant velocity (Const. vel.) and the other with a multi-layer Perceptron (MLP)

To evaluate performance in the simulator, we rely on 3 metrics: lateral, high speed and low speed errors count. Lateral errors (*lat*) occur when the simulated vehicle deviates from the expert trajectory from more than 1m. High and low speed errors (*high* and *low*) occur when the simulated vehicle is too fast (catching up to a vehicle 15% faster than the real vehicle up to 0.6s in the future) or too slow (simulated speed is 20kph or lower under the expert one). As expected, offline metrics such as minADE are not discriminating enough for the online behavior. Removing raw sensor data results roughly in a 10% drop in minADE on nuScenes, and training a multi-layer perceptron (MLP) using only past trajectory data gives a final ADE of 0.4m on the test data. However these approaches are absurd online: they cannot access mandatory information such as traffic lights, road intersections etc. Similarly, the impact of semantic segmentation auxiliary loss can only be seen online. Quantitative results in the simulator over the test recordings, 8km of busy urban scenario and 5kms of test track, can be seen in Table 5, and errors locations can be visualized in Figure 5. More details are available in the supplementary materials.

5.3 Runtime and Optimization

We implemented the proposed architecture using Tensorflow. We used an embedded device as computing platform (32Tops at best) for on-board computation and optimized our model using the TF-TRT library to use the full potential of the embedded GPU. We achieved 13-15 FPS online while returning all outputs (ego trajectories, neighbors trajectories and semantic segmentation) and were able to push the numbers to 22-25 FPS with only ego trajectories and neighbors trajectories outputs.

6 CONCLUSIONS

In this work, we demonstrate the interest of our multi-input multimodal approach PLOP for vehicle trajectory prediction in an urban environment. Our architecture leverages frontal camera and Lidar inputs, to produce multiple trajectories using reparameterized Mixture Density Networks, with an auxiliary semantic segmentation task. We show that we can improve open loop state-of-the-art performance in a multi-agent system, by evaluating the vehicle trajectories from the nuScenes dataset. We also provide a simulated closed loop evaluation, to go towards real vehicle online application.

In future work, we would be interested to include object detections as part of the architecture, to make it truly end-to-end and to use only raw sensor data. Including other types of road users, such as pedestrians or cyclists, could also make the system safer in a busy urban environment. Finally, it would be relevant to address the problem of generalization, and domain shift in particular due to weather conditions.

Acknowledgments

The authors would like to thank the Valeo.ai team for their comments on the article, and people supporting the vehicle operation, in particular Romain Desmarets, Gaetan Le-Gall and Brouk Wedajo.

References

[1] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018.

[2] S. Casas, W. Luo, and R. Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018.

[3] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *ICCV*, 2019.

[4] N. Rhinehart, K. M. Kitani, and P. Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018.

[5] 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*, pages 1468–1476, 2018.

[6] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016.

[7] T. Buhet, E. Wirbel, and X. Perrotton. Conditional vehicle trajectories prediction in carla urban environment. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[8] S. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. *CoRR*, 2018.

[9] M. Bansal, A. Krizhevsky, and A. S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, 2018.

[10] C. M. Bishop. Mixture density networks. 1994.

[11] O. Makansi, E. Ilg, O. Cicek, and T. Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *ICRA*, 2019.

[13] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[14] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, 1995.

[15] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019.

[16] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *CVPR*, 2020.

[17] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[18] B. Ivanovic and M. Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2375–2384, 2019.

[19] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. *arXiv preprint arXiv:2001.03093*, 2020.

[20] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Advances in Neural Information Processing Systems*, pages 137–146, 2019.

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

[22] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *ICCV*, 2019.

[23] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. 33:6120–6127, 2019.

[24] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha. Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms. *arXiv preprint arXiv:1912.01118*, 2019.

[25] S. Casas, C. Gulino, R. Liao, and R. Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint arXiv:1910.08233*, 2019.

[26] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *CVPR*, 2020.

[27] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction, 10 2019.

[28] J. Hong, B. Sapp, and J. Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. *CoRR*, 2019.

[29] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15424–15434, 2019.

[30] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: socially acceptable trajectories with generative adversarial networks. *CoRR*, 2018.

[31] N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. *CoRR*, 2018.

[32] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *CVPR*, 2020.

[33] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *CVPR*, 2020.

[34] E. Ohn-Bar, A. Prakash, A. Behl, K. Chitta, and A. Geiger. Learning situational driving. In *CVPR*, 2020.

[35] S. Hoermann, M. Bach, and K. Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. *CoRR*, 2017.

[36] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun. End-to-end driving via conditional imitation learning. *CoRR*, 2017.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

[38] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, 2015.

[40] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the variance of the adaptive learning rate and beyond. *ArXiv*, 2019.

[41] N. Lee, W. Choi, P. Vernaza, C. Choy, P. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. pages 2165–2174, 07 2017. doi:10.1109/CVPR.2017.233.

[42] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, and P. Schuberth. A2d2: Aev autonomous driving dataset. http://www.a2d2.audi, 2019.

[43] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

[44] M. Toromanoff, E. Wirbel, and F. Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *CVPR*, 2020.

[45] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017.

[46] F. Codevilla, A. M. Lopez, V. Koltun, and A. Dosovitskiy. On offline evaluation of vision-based driving models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 236–251, 2018.

[47] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020.

[48] M. Toromanoff, É. Wirbel, F. Wilhelm, C. Vejarano, X. Perrotton, and F. Moutarde. End to end vehicle lateral control using a single fisheye camera. *CoRR*, abs/1808.06940, 2018. URL http://arxiv.org/abs/1808.06940.