Implicit Latent Variable Model for Scene-Consistent Motion Forecasting

Sergio Casas*[1,2], Cole Gulino*[1], Simon Suo*[1,2],
Katie Luo[1], Renjie Liao[1,2], Raquel Urtasun[1,2]

Uber ATG[1], University of Toronto[2]
{sergio.casas, cgulino, suo, katie.luo, rjliao, urtasun}@uber.com

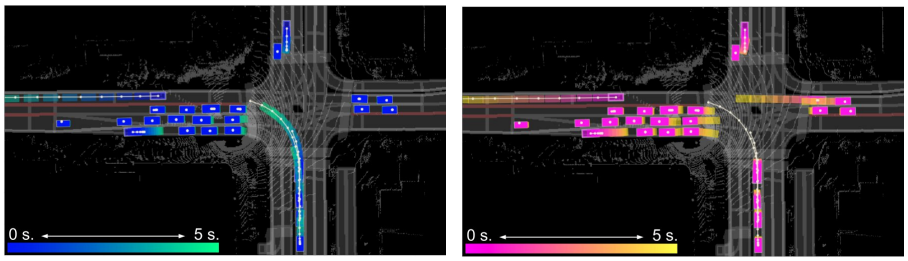
Abstract. In order to plan a safe maneuver an autonomous vehicle must accurately perceive its environment, and understand the interactions among traffic participants. In this paper, we aim to learn scene-consistent motion forecasts of complex urban traffic directly from sensor data. In particular, we propose to characterize the joint distribution over future trajectories via an implicit latent variable model. We model the scene as an interaction graph and employ powerful graph neural networks to learn a distributed latent representation of the scene. Coupled with a deterministic decoder, we obtain trajectory samples that are consistent across traffic participants, achieving state-of-the-art results in motion forecasting and interaction understanding. Last but not least, we demonstrate that our motion forecasts result in safer and more comfortable motion planning.

1 Introduction

Self driving vehicles (SDV) have the potential to make a broad impact in our society, providing a safer and more efficient solution to transportation. A critical component for autonomous driving is the ability to perceive the world and forecast all possible future instantiations of the scene. 3D perception algorithms have improved incredibly fast in recent years [34, 38, 48, 60, 62, 65], yielding very accurate object detections surrounding the SDV. However, producing multi-modal motion forecasts that precisely capture multiple plausible futures consistently for all actors in the scene remains a very open problem.

The complexity is immense: the future is inherently uncertain as actor behaviors are influenced not only by their own individual goals and intentions but also by the other actors' actions. For instance, an actor at an intersection may choose to turn right or go straight due to its own destination, and yield or go if the behavior of a nearby traffic participant is aggressive or conservative. Moreover, unobserved traffic rules such as the future traffic light states heavily affect the traffic (see Fig.1). It is clear that all these aspects cannot be directly observed and require complex reasoning about the scene as a whole, including its geometry, topology and the interaction between multiple agents.

* Denotes equal contribution



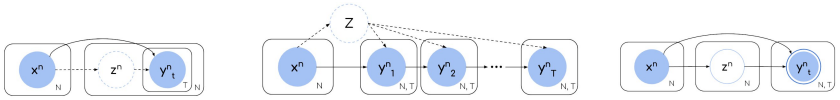
(a) Sample 1: protected left turn (b) Sample 2: horizontal traffic flow

Fig. 1: **Two scene-consistent future trajectory samples from our model**. Ground truth trajectories are shown as white polylines.

In an autonomy system, detections and motion forecasts for other actors in the scene are typically passed as obstacles to a motion-planner [49, 54] in order to plan a safe maneuver. Importantly, the distribution over future trajectories needs to cover the ground-truth for the plan to be safe, but also must exhibit low enough entropy such that a comfortable ride with reasonable progress is achieved. Thus in complex urban environments the SDV should reason about multiple futures separately [17, 23, 32], and plan proactively by understanding how its own actions might influence other actors' behaviors [18,45]. Furthermore, as self-driving vehicles get closer to full autonomy, closed-loop simulation is becoming increasingly critical not only for testing but also for training. In a self-driving simulator [6, 14, 43], smart-actor models [3, 4, 7, 58] are responsible for generating stochastic joint behaviors that are realistic at a scene-level, with actors obeying to underlying scene dynamics with complex interactions.

These applications require learning a joint distribution over actors' future trajectories that characterizes how the scene might unroll as a whole. Since this is generally intractable, many motion forecasting approaches [9, 11, 12, 20] assume marginal independence across actors' future trajectories, failing to get scene-consistent futures. Alternatively, auto-regressive formulations [51, 57] model interactions at the output level, but require sequential sampling which results in slow inference and compounding errors [53].

To overcome these challenges, we propose a novel way to characterize the joint distribution over motion forecasts via an implicit latent variable model (ILVM). We aim to recover a latent space that can summarize all the unobserved scene dynamics given input sensor data. This is challenging given that (i) modern roads present very complex geometries and topologies that make every intersection unique, (ii) the dynamic environment is only partially observed through sensor returns, and (iii) the number of actors in a scene is variable. To address these, we model the scene as an interaction graph [9, 25, 31, 37], where nodes are traffic participants. We then partition the scene latent space into a distributed representation among actors. We leverage graph neural networks (GNN) [2] both to encode the full scene into the latent space as well as to decode latent samples into socially consistent future trajectories. We frame the decoding of all actors'



(a) Independent output (b) Social auto-regressive (c) ILVM (Ours)

Fig. 2: **Graphical models of trajectory distribution**. Dashed arrows/circles denote that only some approaches within the group use those components. Double circle in (c) denotes that it is a deterministic mapping of its inputs.

trajectories as a deterministic mapping from the inputs and scene latent samples, making the latent variables capture all the stochasticity in our generative process. Furthermore, this allows us to perform efficient inference via parallel sampling.

We show that our ILVM significantly outperforms the motion forecasting state-of-the-art in ATG4D [62] and nuScenes [8]. We observe that our ILVM is able to generate scene-consistent samples (see Fig. 1) while producing less entropic joint distributions that also better cover the ground-truth. Moreover, when using our scene-consistent motion forecasts, a state-of-the-art motion planner [54] can plan safer and more comfortable trajectories.

2 Related Work

In this section, we review recent advances in motion forecasting, with a focus on realistic approaches that predict from sensor data, explicitly reason about the multi-modality of the output distribution, or model multi-agent interactions.

In traditional self-driving stacks, an object detection module is responsible for recognizing other traffic participants in the scene, followed by a motion forecasting module that predicts how the scene might unroll given the current state of each actor. However, the actor state is typically a very compact representation that includes pose, velocity, and acceleration. As a consequence, it is hard to incorporate uncertainty due to sensor noise or occlusion.

We follow the works of [10, 40, 63], which unified these two tasks by having a single fully convolutional backbone network predict both the current and future states for each pixel in a bird's eye view grid directly from a voxelized LiDAR point-cloud and semantic raster of an HD map. This approach naturally propagates uncertainty between the two tasks in the feature space, without the need of explicit intermediate representations. While these models reason about uncertainty in sensor observations, they neglect inherent uncertainty in the actors' future behavior. [9, 37] add agent-agent interaction reasoning to this framework. [9] introduces spatially-aware graph neural networks that aggregate features from neighboring actors in the scene to predict a single trajectory per actor with gaussian waypoints, assuming marginal independence across actors. This approach is still limited in expressivity since (i) a uni-modal characterization of the future is insufficient for downstream motion planning to make safe

decisions, and (ii) modeling the marginal distribution per actor cannot provide trajectory samples that are consistent across actors.

Another research stream [1,13,21,31,35,42,50,51,57] has focused on the problem of multi-agent trajectory prediction from perfect perception, i.e., assuming that the ground-truth past trajectory of all actors' is given. Unfortunately, this is not realistic in self-driving vehicles, which rely on imperfect perception with noise that translates into failures such as false positive and false negative detections and id switches in tracking. Nonetheless, these methods have proposed output parameterizations that can predict multi-modal distributions over future trajectories, which are applicable to our end-to-end perception and prediction setting.

Various factorizations of the joint distribution over N actors' trajectories $p(Y|X) = p(y_1, \cdots, y_N|x_1, \cdots, x_N)$ with different levels of independence assumptions have been proposed to sidestep the intractability of $p(Y|X)$. The simplest approximation is to assume *independent futures* across actors and time steps $p(Y|X) = \prod_n \prod_t p(y_n^t|X)$, as shown in Fig. 2a. Some approaches directly regress the parameters of a mixture of Gaussians over time [11, 12, 39], which provides efficient sampling but can suffer from low expressivity and unstable optimization. Non-parametric approaches [26,27,47,52] have also been proposed to characterize the multi-modality of one actor's individual behavior. These approaches either score trajectory samples from a finite set [47, 64] with limited coverage or predict an occupancy grid at different future horizons [26, 27, 52], which is very memory consuming. [50] proposed to learn a one-step policy that predicts the next waypoint based on the previous history, avoiding the time independence assumption. Variational methods [20, 36] inspired by [29, 56] have also been proposed to learn an actor independent latent space to capture unobserved actor dynamics such as goals. Unfortunately, none of these methods can accurately characterize the joint distribution in interactive situations, since the generative process is independent per actor.

An alternative approach to better characterize the behavior of multiple actors jointly is *autoregressive generation* with social mechanisms [1, 51], which predict the distribution over the next trajectory waypoint of each actor conditioned on the previous states of all actors $p(Y|X) = \prod_n \prod_t p\left(y_n^t|Y^{0:t-1}, X\right)$. This approach has been enhanced by introducing latent variables [25, 31, 57], as in Fig. 2b. In particular, [31] introduces discrete latent variables to model pairwise relationships in an interaction graph, while in [25,57] they capture per-actor high-level actions. Autoregressive approaches, however, suffer from compounding errors [33, 46, 53]. During training, the model is fed the ground-truth $Y^{0:t-1}$, while during inference, the model must rely on approximate samples from the learned distribution. While scheduled sampling [5] has been proposed to mitigate this issue, the objective function underlying this method is improper [24] and pushes the conditional distributions $p(y_n^t|Y^{0:t-1})$ to model the marginal distributions $p(y_n^t)$ instead. Moreover, these methods require sequential sampling, which is not amenable to real-time applications such as self-driving.

In contrast to previous works, we propose to model interaction in a scene latent space that captures all sources of uncertainty, and use a deterministic decoder to characterize an implicit joint distribution over all actors' future trajectories without any independence assumptions at the output level, as shown in Fig. 2c. This design features efficient parallel sampling, high expressivity and yields trajectory samples that are substantially more consistent across actors.

3 Scene Level Reasoning for Motion Forecasting

In this section we introduce our approach to model **the joint distribution** $P(Y|X)$ **over** N **actors' future trajectories** $Y = \{y_1, y_2, \cdots, y_N\}$ given each actor's local context $X = \{x_1, x_2, \cdots, x_N\}$ extracted from sensor data and HD maps. An actor's trajectory y_n is composed of 2D waypoints over time y_n^t in the coordinate frame defined by the actor's current position and heading. In the following, we first explain our implicit latent variable model, then introduce our concrete architecture including the actor feature extraction from sensor data, and finally explain how to train our model in an end-to-end manner.

3.1 Implicit Latent Variable Model with Deterministic Decoder

We formulate the generative process of future trajectories over actors with a latent variable model:

$$P(Y|X) = \int_Z P(Y|X,Z)P(Z|X)dZ$$

where Z is a latent variable that captures unobserved scene dynamics such as actor goals and style, multi-agent interactions, or future traffic light states.

We propose to use a **deterministic mapping** $Y = f(X,Z)$ to implicitly characterize $P(Y|X,Z)$, instead of explicitly representing it in a parametric form. This approach allows us to avoid factorizing $P(Y|X,Z)$ (as in Fig. 2a or Fig. 2b) and sidestep the associated shortcomings discussed in Section 2. In this framework, generating scene-consistent future trajectories Y across actors is simple and highly efficient, since it only requires one stage of parallel sampling:

1. Draw latent scene samples from prior $Z \sim P(Z|X)$
2. Decode with the deterministic decoder $Y = f(X,Z)$

We emphasize that this modeling choice encourages the latent Z to capture *all* stochasticity in our generative process. To this end, we leverage a *continuous latent* Z for high expressivity. This stands in contrast to previous methods [25,31,57], where discrete latent Z are employed to model discrete high-level actions or pairwise interactions, and an explicit $P(Y|X,Z)$ to represent continuous uncertainty.

Producing a latent space that can capture all the uncertainties in any scenario is challenging: scenarios vary drastically in the number of actors N, the road topology as well as traffic rules. To mitigate this challenge, we propose to

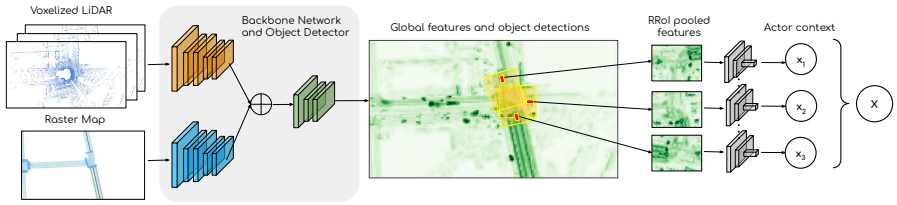


Fig. 3: **Actor Feature Extraction**. Given LiDAR and maps, our backbone CNN detects the actors in the scene, and individual feature vectors per actor are extracted via RRoI Align [41], followed by a CNN with spatial pooling.

partition the scene latent as $Z = \{z_1, z_2, \cdots, z_N\}$, obtaining a distributed representation where z_n is anchored to actor n in an interaction graph with traffic participants as nodes. The distributed representation has the benefit of naturally scaling the capacity of the latent space as the number of actors grow. Furthermore, the anchoring gives the model an inductive bias that eases the learning of a scene latent space. Intuitively, each latent z_n encodes unobserved dynamics most relevant to actor n, including interactions with neighboring actors and traffic rules that apply in its locality. We represent each z_n as a diagonal multivariate gaussian $z_n \sim \mathcal{N}\left(\left[\mu_n^1(X), \cdots, \mu_n^D(X)\right], \text{diag}\left(\left[\sigma_n^1(X), \cdots, \sigma_n^D(X)\right]\right)\right)$, as is common with variational models [29,56]. We emphasize that although factorized, the latent space is not marginally independent across actors since each z_n is conditioned on all $x_1, \cdots, x_N$ as shown in the graphical model in Fig. 2c.

Since integration over Z is intractable, we exploit amortized variational inference [29,56]. By introducing an encoder distribution $Q(Z|X,Y)$ to approximate the true posterior $P(Z|X,Y)$, the learning problem can be reformulated as a maximization of the Evidence Lower BOund (ELBO). Please visit the supplementary for a more thorough description of variational inference.

3.2 Joint Perception and Motion Forecasting Architecture

Our architecture consists of an actor feature extractor that detects objects in the scene and provides rich representations of each actor (Fig. 3), encoder/prior modules that infer a scene latent space at training/inference respectively, and a decoder that predicts the actors' future trajectories (Fig. 4). To implement the prior, encoder and decoder modules, we leverage a flexible scene interaction module (SIM) as our building block for relational reasoning (Alg. 1).

Actor Feature Extractor: Fig. 3 shows how we extract per actor features $X = \{x_1, x_2, \cdots, x_N\}$ from raw sensor data and HD maps in a differentiable manner, such that perception and motion forecasting can be trained jointly end-to-end. We use a CNN-based perception backbone network architecture inspired by [10,62] to extract rich geometrical and motion features about the whole scene from a past history of voxelized LiDAR point clouds and a raster map. We then detect [62] the traffic participants in the scene, and apply Rotated Region of Interest Align [41] to the backbone features around each object detection,

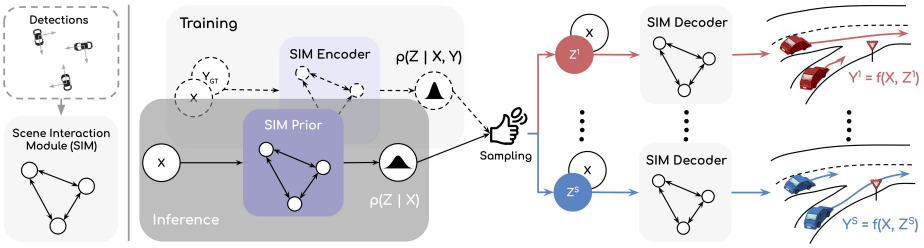


Fig. 4: **Our Implicit Latent Variable Model** encodes the scene into a latent space, from which it can efficiently sample multiple future realizations in parallel, each with socially consistent trajectories.

providing the local context for all actors, as proposed by [9]. As mentioned at the beginning of Section 3, this will be the input to our motion forecasting module. This contrasts with previous approaches (e.g., [11, 13, 51, 57]) that assume past trajectories for each actor are given. We refer the reader to the supplementary material for more details about our perception module, including the backbone architecture and detection parameterization.

Scene Interaction Module (SIM): This is a core building block of our encoder, prior, and decoder networks, as shown in Fig. 4. Once we have extracted individual actor features, we can frame the scene as a fully-connected interaction graph where each traffic participant is a node. Inspired by [9], we use a spatially-aware graph neural network to model multi-agent dynamics, as described in Alg. 1. Our SIM performs a single round of message passing to update the nodes' representation, taking into account spatiotemporal relationships.

Encoder: To approximate the true posterior latent distribution $P(Z|X,Y)$, we introduce an approximate posterior $q_\phi(Z|X,Y)$, implemented by our SIM and parameterized by ϕ. This network is also commonly known as recognition network, since it receives the target output variable Y as an input, and thus it can *recognize* the scene dynamics that are unobserved by the prior $p_\gamma(Z|X)$. Note that the encoder can only be used during training, since it requires access to the ground-truth future trajectories. We initialize the node representations as $h_n = \text{MLP}(x_n \oplus \text{GRU}(y_n))$, where $\oplus$ denotes concatenation along the feature dimension. After running one round of message passing, the scene interaction module predicts the distribution over scene latent variables $Z = \{z_1, z_2, \cdots, z_N\}$. We stress that despite anchoring each partition of the scene latent to an actor, each individual z_n contains information about the full scene, since each final node representation is dependent on the whole input X because of the message propagation in the fully-connected interaction graph.

Prior: The prior network $p_\gamma(Z|X)$ is responsible for approximating the prior distribution of the scene latent variable Z at inference time. Similar to the encoder, we model the scene-level latent space with our SIM, where the only difference is that the initial node representations in the graph propagation are $h_n = \text{MLP}(x_n)$, since y_n is not available at inference time.

Algorithm 1 Scene Interaction Module (SIM)

Input: Initial hidden state for all of the actors in the scene $\{h_0, h_1, \cdots, h_N\}$. BEV coordinates (centroid and heading) of the detected bounding boxes $\{c_0, c_1, ..., c_N\}$.
Output: Feature vector per node $\{o_0, o_1, \cdots, o_N\}$.

Construct fully-connected interaction graph $G = (V, E)$ from detections
Compute pairwise coordinate transformations $\mathcal{T}(c_u, c_v)$, $\forall (u, v) \in E$
for $(u, v) \in E$ **do** ▷ Compute message for every edge in the graph in parallel
    $m_{u \to v} = \text{MLP}\left(h_u, h_v, \mathcal{T}(c_u, c_v)\right)$
for $v \in V$ **do** ▷ Update node states in parallel
    $a_v = \text{MaxPooling}\left(\{m_{u \to v} : u \in \mathbf{N}(v)\}\right)$ ▷ Aggregate messages from neighbors
    $h'_v = \text{GRU}\left(h_v, a_v\right)$ ▷ Update the hidden state
    $o_v = \text{MLP}\left(h'_v\right)$ ▷ Compute outputs
return $\{o_0, o_1, \cdots, o_N\}$

Deterministic Decoder: Recall that our scene latent has been partitioned into a distributed representation $Z = \{z_1, z_2, \cdots, z_N\}$. To leverage actor features and distributed latents from the whole scene, we parameterize the decoder with another SIM. We can then predict the s-th realization of the future at a scene level via message passing, where each actor trajectory y_n^s takes into account a sample from all the partitions of the scene latent $Z^s = \{z_1^s, \cdots, z_n^s\}$ as well as all actors' features X, enabling reasoning about multi-agent interactions such as car following, yielding, etc. More precisely, given each actor context x_n, we initialize its node representation for the decoder graph propagation as $h_n^s = \text{MLP}(x_n \oplus z_n^s)$. After a round of message passing in our SIM, $h_n^{'s}$ contains an updated representation of actor n that takes into account the underlying dynamics of the whole scene summarized in Z^s. Finally, the s-th trajectory sample for actor n is deterministically decoded $y_n^s = \text{MLP}(h_n^{'s})$ by the SIM output function, without additional sampling steps. The trajectory-level scene sample is simply the collection of all actor trajectories $Y^s = \{y_1^s, \ldots, y_N^s\}$. We can generate S possible futures for all actors in the scene in parallel by batching S scene latent samples.

In this fashion, our model implicitly characterizes the joint distribution over actors' trajectories, achieving superior scene-level consistency. In the experiments section we ablate the design choices in the encoder, prior and decoder, and show that although all of them are important, the deterministic decoder is the key contribution towards socially-consistent trajectories.

3.3 Learning

Our perception and prediction model can be trained end-to-end using stochastic gradient descent. In particular, we minimize a multi-task loss for detection and motion forecasting: $\mathcal{L} = \mathcal{L}_{\text{det}} + \lambda \cdot \mathcal{L}_{\text{forecast}}$

Detection: For the detection classification branch we employ a binary cross entropy loss with hard negative mining $\mathcal{L}_{\text{cla}}$. We select all positive examples from the ground-truth and 3 times as many negative examples. For box fitting, we apply a smooth ℓ_1 loss $\mathcal{L}_{\text{reg}}$ to each of the 5 parameters $(x_i, y_i, w_i, h_i, \phi_i)$ of the bounding boxes anchored to a positive example i. The overall detection loss is a linear combination $\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{cla}} + \alpha \cdot \mathcal{L}_{\text{reg}}$.

Motion Forecasting: We adapt the variational learning objective of the CVAE framework [56] and optimize the evidence-based lower bound (ELBO) of the log-likelihood $\log P(Y|X)$. In our case, due to the deterministic decoder leading to an implicit distribution over Y, we use Huber loss ℓ_δ as the reconstruction loss, and reweight the KL term with β as proposed by [19]:

$$\mathcal{L}_{\text{forecast}} = \sum_{n}^{N} \sum_{t}^{T} \ell_\delta(y_n^t - y_{n,GT}^t) + \beta \cdot \text{KL}\left(q_\phi\left(Z|X, Y_{GT}\right) || p_\gamma\left(Z|X\right)\right)$$

where the first term minimizes the reconstruction error between all the trajectories in the scene $Y = \{y_n^t | \forall n, t\} = f_\theta(Z)$, $Z \sim q_\phi\left(Z|X, Y_{GT}\right)$ and their corresponding ground-truth Y_{GT}, and the second term brings the privileged *posterior* $q_\phi(Z|X, Y_{GT})$ and the approximate *prior* $p_\gamma(Z|X)$ distributions closer.

4 Experimental Evaluation

In this section, we first explain the metrics and baselines we use for evaluation. Next, we compare our model against state-of-the-art motion forecasting algorithms on predicting the future 5 second trajectories on two real-world datasets: ATG4D [62] and nuScenes [8] (see supplementary for details). Then, we measure the impact on motion planning. Finally, we carry out an ablation study to understand which part of our model contributes the most.

4.1 Scene Level Motion Forecasting Metrics

Previous methods use sample quality metrics at the actor level such as the popular minimum/mean average displacement error (minADE/meanADE). However, these metrics only evaluate the quality of the underlying marginal distribution per actor. For instance, minADE takes the trajectory sample that best fits the ground-truth of each actor independently, which does not measure the consistency between different actors sample trajectories and can be easily cheated by predicting high entropy distributions that cover all the space but are not precise.

We propose scene-level sample quality metrics to evaluate how well the models capture the joint distribution over future outcomes. To this end, we define a scene-level counterpart of the popular minimum/mean average displacement error. We emphasize that in this context, each scene sample $s \in 1, ..., S$ is a

Type	Model	SCR_{5s} (%)	min SFDE(m)	min SADE(m)	mean SFDE(m)	mean SADE(m)
Indep. Output	SpAGNN [9]	8.19	2.83	1.34	4.37	1.92
	RulesRoad [20]	6.66	2.71	1.32	4.21	1.84
	MTP [12]	3.98	1.91	0.95	3.11	1.37
	MultiPath [11]	4.41	1.97	0.95	3.14	1.36
	R2P2-MA [50]	4.63	2.13	1.09	3.27	1.49
Social Auto-regressive	SocialLSTM [1]	6.13	2.75	1.38	4.05	1.83
	NRI [31]	7.00	2.68	1.43	3.81	1.74
	ESP [51]	2.67	1.91	0.97	2.84	1.29
	MFP [57]	5.15	2.35	1.13	3.35	1.45
	ILVM	**0.70**	**1.53**	**0.76**	**2.27**	**1.02**

Table 1: **[ATG4D] Scene-level motion forecasting** ($S = 15$ samples)

collection of N future trajectories, one for each actor in the scene.

$$\text{minSADE} = \min_{s \in 1 \dots S} \frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} ||y_{n,GT}^{t} - y_{n,s}^{t}||^2$$

$$\text{meanSADE} = \frac{1}{NTS} \sum_{s=1}^{S} \sum_{n=1}^{N} \sum_{t=1}^{T} ||y_{n,GT}^{t} - y_{n,s}^{t}||^2$$

We also compute their final counterparts minSFDE and meanSFDE, which evaluate only the motion forecasts at the final timestep (i.e. at 5 seconds).

Furthermore, to evaluate the consistency of the motion forecasts we propose to measure the scene collision rate (SCR). It measures the percentage of trajectory samples that collide with any other trajectory in the same scene sample s. Two trajectory samples are considered in collision if the overlap between their future bounding boxes at any time step is higher than a small IOU threshold ε_{IOU}. To compute this, we first obtain the bounding boxes for future time steps $\{b_{i,s}^{t}\}$. The size of the bounding boxes are the same as their object detections and the future headings are extracted by finite differences on the trajectory samples.

$$\text{SCR}_T = \frac{1}{NS} \sum_{s=1}^{S} \sum_{i=1}^{N} \min \left(1, \sum_{j>i}^{N} \sum_{t=1}^{T} \mathbb{1} \left[IoU(b_{i,s}^{t}, b_{j,s}^{t}) > \varepsilon_{IOU} \right] \right)$$

Finally, to perform a fair comparison in motion forecasting metrics, which are evaluated on true positive detections, we follow [9] and operate the object detector at 90% recall point for all models in ATG4D and 80% in nuScenes.

4.2 Baselines

In this section, we discuss the state of the art motion forecasting models that we use as baselines. It is important to note that most baselines are designed for motion forecasting given perfect perception, i.e., ground-truth past trajectories. However, this is not realistic in self-driving vehicles, which rely on imperfect noisy

Type	Model	SCR_{5s} (%)	min SFDE(m)	min SADE(m)	mean SFDE(m)	mean SADE(m)
Indep. Output	SpAGNN [9]	7.54	2.07	1.00	3.85	1.82
	RulesRoad [20]	5.67	2.10	1.01	3.55	1.67
	MTP [12]	8.68	1.86	0.91	3.86	1.85
	MultiPath [11]	7.31	2.01	0.95	3.50	1.65
	R2P2-MA [50]	4.56	2.25	1.08	3.47	1.67
Social Auto-regressive	SocialLSTM [1]	6.45	2.71	1.33	4.20	2.05
	NRI [31]	5.98	2.54	1.28	3.91	1.88
	ESP [51]	5.09	2.16	1.07	3.46	1.67
	MFP [57]	4.94	2.74	1.30	4.11	1.95
	ILVM	**1.91**	**1.84**	**0.86**	**2.99**	**1.43**

Table 2: **[nuScenes] Scene-level motion forecasting** ($S = 15$ samples)

perception. Thus, we adapt them to the realistic setting by replacing their past trajectory encoders with our extracted actor features (see Fig. 3) and training end-to-end with our perception backbone (see supplementary for details).

Independent output: We benchmark against SpAGNN [9], MTP [12], MultiPath [11], RulesRoad [20], and R2P2-MA [50]. Since the trajectory sampling process from these models is independent per actor, we define a scene sample s by drawing one sample for each actor in the scene.

Social auto-regressive: We compare against SocialLSTM [1], ESP [51], MFP [57], and NRI [31]. It is worth sharing that for these baselines to achieve competitive results we had to perturb the ground-truth trajectories with white noise during training. This is because these models suffer from a distributional shift between training and inference, as explained in Section 2. We note that white noise was more effective than teacher forcing [33] or scheduled sampling [5].

4.3 Motion Forecasting Results

Experimental results for motion forecasting in the ATG4D dataset (with $S = 15$ samples) are shown in Table 1. Our ILVM outperforms the baselines across all metrics. Very notably, it **achieves a 75% reduction in collision rate** with respect to the strongest baseline in this metric (ESP [51]), thus highlighting the better characterization of the joint distribution across actors (which also translates into scene-consistent samples). Our model is also much more precise (20% reduction in meanSFDE) while exhibiting better coverage of the ground-truth data (19% reduction in minSFDE). We include an analysis of how the minSADE and minSFDE vary across different number of samples S in the supplementary.

Fig. 5 shows individual samples. We heuristically select the two most distinct samples for visualization to show diverse realizations of the future. The baseline models capture variations in individual actors' future, but do not capture the yielding interaction at the intersection, which our model does. In addition, Fig. 6

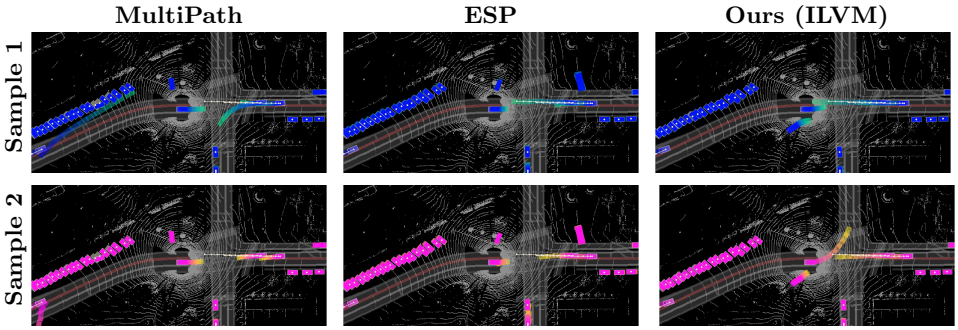


Fig. 5: **Scene-level samples**. Our latent variable model captures underlying scene dynamics at the intersection level (i.e. yield vs. go)

showcases the full distribution learned by the models. More concretely, this plot shows a Monte Carlo estimation of the marginal distribution per actor, where 50 samples are drawn from each model. Transparency in the plots illustrates the probability density at a given location. These examples support the same conclusion taken from the quantitative results and highlight the ability of our model to understand complex road geometries and the multi-modal behaviors they induce. This is particularly interesting since all models share the same representation of the environment and backbone architecture.

To show that our improvements generalize to a dataset with a different distribution of motions and road topologies, we validate our method on nuScenes. Table 2, shows that ILVM brings improvements over the baselines across all metrics. In particular, we observe significant gains in scene-consistency (SCR) and precision metrics (meanSADE and meanSFDE).

4.4 Motion Planning Results

To validate the system-level impact of different perception and prediction models, we use the state-of-the-art learnable motion planner of [55] to plan a trajectory for the SDV (τ_{SDV}):

$$\tau_{\text{SDV}} = \arg\min_{\tau \in \text{T}} \mathbb{E}_{p(Y|X)} \left[c(\tau, Y \setminus y_{\text{SDV}}) \right] \approx \arg\min_{\tau \in \text{T}} c(\tau, \left\{ Y^s \setminus y^s_{\text{SDV}} : \forall s \in 1 \dots S \right\})$$

where $p(Y|X)$ is the distribution over future trajectories output by the perception and prediction model, T is a predefined set of SDV trajectories given the map and high-level route, and c is a costing function that measures safety and comfort taking into account the motion forecasts for the rest of the vehicles. More concretely, the motion planner receives a Monte Carlo estimate of the future trajectory distribution with $S = 50$ sample trajectories (see Fig.6) for every detected vehicle (excluding the SDV), which are considered obstacles in order to approximate the expected cost of plans $\tau \in \text{T}$.

The experiments in Table 3 measure how different motion forecasts translate into the safety and comfort of the SDV trajectory (τ_{SDV}), an impact often

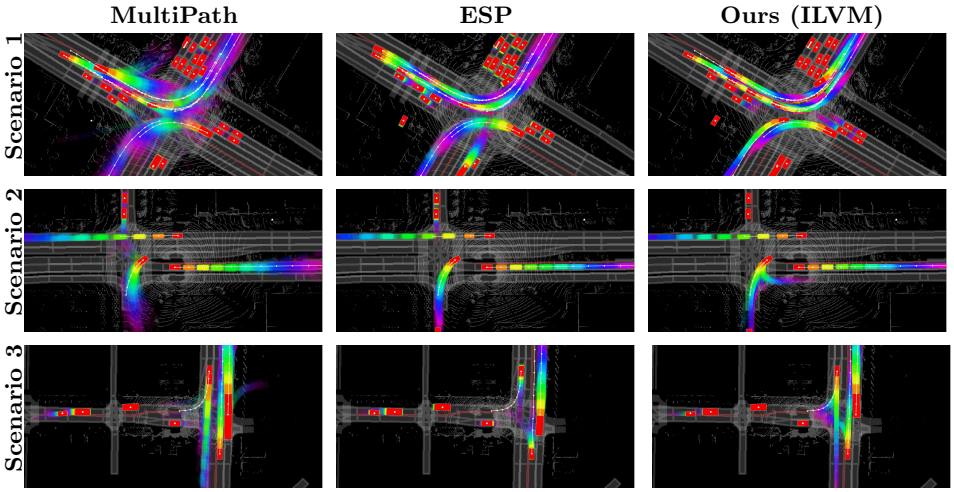


Fig. 6: **Motion forecasting visualizations of 50 samples**. Time is encoded in the rainbow color map ranging from red (0s) to pink (5s).

overlooked by previous works. Our motion forecasts (ILVM) enable the motion planner to execute significantly safer and more comfortable trajectories. We notice that the ego-motion plans make similar progress across models, but our approach produces the closest trajectories to the expert demonstrations (lowest ℓ_2 distance at 5 seconds into the future), while yielding much fewer collisions. We include planning qualitative results in our supplementary material.

4.5 Ablation Study

Implicit vs. Explicit Decoder: We ablate ILVM ($\mathcal{M}_0$ in Table 4) by replacing the proposed implicit decoder with an explicit decoder that produces a full covariance bi-variate Gaussian per waypoint, and the reconstruction loss with Negative Log Likelihood. This gives us $\mathcal{M}_1$, where ancestral sampling is used for inference: first sample latent, then sample output. Here, we can see that assuming conditional independence across actor at the output level significantly degrades all aspects of the motion forecasting performance. Most notably, the high scene collision rate shows that the samples are no longer socially consistent.

Learned vs. Fixed Prior: A comparison between $\mathcal{M}_0$ and $\mathcal{M}_2$ in Table 4 shows that using a learned prior network $P(Z|X)$ achieves a better precision diversity trade-off compared to using a fixed prior distribution of isotropic Gaussians.

ILVM architecture: In Table 4, $\mathcal{M}_3$ ablates the SIM encoder and prior networks by replacing them with MLPs that model $p(z_n|x_n)$ and $p(z_n|x_n, y_n)$ at the actor-level, respectively. $\mathcal{M}_4$ replaces the SIM decoder by an MLP per actor $y_n^s = \text{MLP}(X, z_n^s)$. Finally, $\mathcal{M}_5$ applies the changes in $\mathcal{M}_3$ and $\mathcal{M}_4$. These experiments show that both the graph based prior/encoder and decoder are important

Type	Model	Collision (% up to 5s)	L2 human (m @ 5s)	Lat. acc. (m/s^2)	Jerk (m/s^3)	Progress (m @ 5s)
Indep. Output	SpAGNN [9]	4.19	5.98	2.94	2.90	32.37
	RulesRoad [20]	4.04	5.83	2.84	2.76	32.50
	MTP [12]	3.10	5.67	2.83	2.66	33.14
	MultiPath [11]	3.30	5.58	2.73	2.57	32.99
	R2P2-MA [51]	3.71	5.65	2.84	2.53	33.90
Social Auto-regressive	SocialLSTM [1]	4.22	5.92	2.76	2.66	32.60
	NRI [31]	4.94	5.73	2.78	2.55	33.43
	ESP [51]	3.13	5.48	2.76	2.44	33.74
	MFP [57]	4.14	5.57	2.61	2.43	32.94
	ILVM	**2.64**	**5.33**	**2.59**	**2.30**	33.72

Table 3: **[ATG4D] System Level Performance (ego-motion planning)**

ID	Learned Prior	Implicit Output	SIM Encoder	SIM Decoder	SCR_{5s}	min SFDE	min SADE	mean SFDE	mean SADE
$\mathcal{M}_0$	✓	✓	✓	✓	**0.70**	**1.53**	**0.76**	**2.27**	**1.02**
$\mathcal{M}_1$	✓		✓	✓	8.46	2.66	1.31	4.17	1.80
$\mathcal{M}_2$		✓	✓	✓	1.10	1.53	0.76	2.43	1.08
$\mathcal{M}_3$	✓	✓		✓	1.03	1.57	0.78	2.42	1.08
$\mathcal{M}_4$	✓	✓	✓		1.52	1.67	0.81	2.44	1.09
$\mathcal{M}_5$	✓	✓			1.74	1.66	0.81	2.43	1.08

Table 4: **[ATG4D] Motion Forecasting Ablation Study** ($S = 15$ samples)

for our latent variable model. In particular, the large gap in scene level collision demonstrates that our proposed SIM encoder and decoder capture scene-level understanding that is not present in the ablations with independent assumptions at the latent or output level.

5 Conclusion and Future Work

We have proposed a latent variable model to obtain an implicit joint distribution over actor trajectories that characterizes the dependencies over their future behaviors. Our model achieves fast parallel sampling of the joint trajectory space and produces scene-consistent motion forecasts. We have demonstrated the effectiveness of our method on two challenging datasets by significantly improving over state-of-the-art motion forecasting models on scene-level sample quality metrics. Our method achieves much more precise predictions that are more socially consistent. We also show that our method produces significant improvements in motion planning, even though the planner does not make explicit use of the strong consistency of our scenes. We leave it to future work to design a motion planner to better utilize joint distributions over trajectories.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: Proceedings of the IEEE CVPR (2016)
2. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks (2018)
3. Behbahani, F., Shiarlis, K., Chen, X., Kurin, V., Kasewa, S., Stirbu, C., Gomes, J., Paul, S., Oliehoek, F.A., Messias, J., et al.: Learning from demonstration in the wild. 2019 International Conference on Robotics and Automation (ICRA) (May 2019). https://doi.org/10.1109/icra.2019.8794412
4. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: Sumo–simulation of urban mobility: an overview. In: Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind (2011)
5. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in Neural Information Processing Systems. pp. 1171–1179 (2015)
6. Best, A., Narang, S., Pasqualin, L., Barber, D., Manocha, D.: Autonovi-sim: Autonomous vehicle simulation platform with weather, sensing, and traffic control. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1161–11618 (2018)
7. Bhattacharyya, R.P., Phillips, D.J., Wulfe, B., Morton, J., Kuefler, A., Kochenderfer, M.J.: Multi-agent imitation learning for driving simulation. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Oct 2018). https://doi.org/10.1109/iros.2018.8593758
8. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)
9. Casas, S., Gulino, C., Liao, R., Urtasun, R.: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. arXiv preprint arXiv:1910.08233 (2019)
10. Casas, S., Luo, W., Urtasun, R.: Intentnet: Learning to predict intention from raw sensor data. In: Conference on Robot Learning (2018)
11. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449 (2019)
12. Cui, H., Radosavljevic, V., Chou, F.C., Lin, T.H., Nguyen, T., Huang, T.K., Schneider, J., Djuric, N.: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. arXiv preprint arXiv:1809.10732 (2018)
13. Djuric, N., Radosavljevic, V., Cui, H., Nguyen, T., Chou, F.C., Lin, T.H., Schneider, J.: Motion prediction of traffic actors for autonomous driving using deep convolutional networks. arXiv preprint arXiv:1808.05819 (2018)
14. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)

15. Engelcke, M., Rao, D., Wang, D.Z., Tong, C.H., Posner, I.: Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In: 2017 ICRA (2017)
16. Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., Carin, L.: Cyclical annealing schedule: A simple approach to mitigating. Proceedings of the 2019 Conference of the North (2019). https://doi.org/10.18653/v1/n19-1021, http://dx.doi.org/10.18653/V1/N19-1021
17. Hardy, J., Campbell, M.: Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. IEEE Transactions on Robotics (2013)
18. Henaff, M., Canziani, A., LeCun, Y.: Model-predictive policy learning with uncertainty regularization for driving in dense traffic. arXiv preprint arXiv:1901.02705 (2019)
19. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework.
20. Hong, J., Sapp, B., Philbin, J.: Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
21. Hoshen, Y.: Vain: Attentional multi-agent predictive modeling. In: Advances in Neural Information Processing Systems. pp. 2701–2711 (2017)
22. Huang, J., Sivakumar, V., Mnatsakanyan, M., Pang, G.: Improving rotated text detection with rotation region proposal networks (2018)
23. Hubmann, C., Schulz, J., Becker, M., Althoff, D., Stiller, C.: Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. IEEE Transactions on Intelligent Vehicles **3**(1), 5–17 (2018)
24. Huszár, F.: How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101 (2015)
25. Ivanovic, B., Pavone, M.: The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2375–2384 (2019)
26. Jain, A., Casas, S., Liao, R., Xiong, Y., Feng, S., Segal, S., Urtasun, R.: Discrete residual flow for probabilistic pedestrian behavior prediction. arXiv preprint arXiv:1910.08041 (2019)
27. Kim, B., Kang, C.M., Kim, J., Lee, S.H., Chung, C.C., Choi, J.W.: Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 399–404. IEEE (2017)
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
29. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2013)
30. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
31. Kipf, T., Fetaya, E., Wang, K.C., Welling, M., Zemel, R.: Neural relational inference for interacting systems. arXiv preprint arXiv:1802.04687 (2018)
32. Klingelschmitt, S., Damerow, F., Eggert, J.: Managing the complexity of inner-city scenes: An efficient situation hypotheses selection scheme. In: 2015 IEEE intelligent vehicles symposium (IV). pp. 1232–1239. IEEE (2015)
33. Lamb, A.M., ALIAS PARTH GOYAL, A.G., Zhang, Y., Zhang, S., Courville, A.C., Bengio, Y.: Professor forcing: A new algorithm for training recurrent networks. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon,

I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6099-professor-forcing-a-new-algorithm-for-training-recurrent-networks.pdf
34. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)
35. Le, H.M., Yue, Y., Carr, P., Lucey, P.: Coordinated multi-agent imitation learning (2017)
36. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: Desire: Distant future prediction in dynamic scenes with interacting agents. In: Proceedings of the IEEE CVPR (2017)
37. Li, L., Yang, B., Liang, M., Zeng, W., Ren, M., Segal, S., Urtasun, R.: End-to-end contextual perception and prediction with interaction transformer. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020)
38. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE CVPR (2019)
39. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: ECCV (2020)
40. Luo, W., Yang, B., Urtasun, R.: Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In: Proceedings of the IEEE CVPR (2018)
41. Ma, J., Shao, W., Ye, H., Wang, L., Wang, H., Zheng, Y., Xue, X.: Arbitrary-oriented scene text detection via rotation proposals. IEEE Transactions on Multimedia (2018)
42. Ma, W.C., Huang, D.A., Lee, N., Kitani, K.M.: Forecasting interactive dynamics of pedestrians with fictitious play. In: Proceedings of the IEEE CVPR (2017)
43. Martinez, M., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A., Kornhauser, A.: Beyond grand theft auto v for training, testing and enhancing deep learning in self driving cars (2017)
44. Mohamed, S., Lakshminarayanan, B.: Learning in implicit generative models. arXiv preprint arXiv:1610.03483 (2016)
45. Okamoto, M., Perona, P., Khiat, A.: Ddt: Deep driving tree for proactive planning in interactive scenarios. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp. 656–661. IEEE (2018)
46. Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., Peters, J., et al.: An algorithmic perspective on imitation learning. Foundations and Trends® in Robotics (2018)
47. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: Multimodal behavior prediction using trajectory sets. arXiv preprint arXiv:1911.10298 (2019)
48. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE CVPR (2017)
49. Ratliff, N.D., Bagnell, J.A., Zinkevich, M.A.: Maximum margin planning. In: Proceedings of the 23rd international conference on Machine learning. pp. 729–736 (2006)
50. Rhinehart, N., Kitani, K.M., Vernaza, P.: R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 772–788 (2018)

51. Rhinehart, N., McAllister, R., Kitani, K., Levine, S.: PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings. arXiv e-prints arXiv:1905.01296 (May 2019)
52. Ridel, D., Deo, N., Wolf, D., Trivedi, M.: Scene compliant trajectory forecast with agent-centric spatio-temporal grids. IEEE Robotics and Automation Letters (2020)
53. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635 (2011)
54. Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.C., Yumer, E., Urtasun, R.: Jointly learnable behavior and trajectory planning for self-driving vehicles. arXiv preprint arXiv:1910.04586 (2019)
55. Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.C., Yumer, E., Urtasun, R.: Jointly learnable behavior and trajectory planning for self-driving vehicles. arXiv preprint arXiv:1910.04586 (2019)
56. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: Advances in neural information processing systems. pp. 3483–3491 (2015)
57. Tang, C., Salakhutdinov, R.R.: Multiple futures prediction. In: Advances in Neural Information Processing Systems. pp. 15398–15408 (2019)
58. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E (2000)
59. Wu, Y., He, K.: Group normalization. In: Proceedings of the ECCV (ECCV) (2018)
60. Yang, B., Guo, R., Liang, M., Sergio, C., Urtasun, R.: Exploiting radar for robust perception of dynamic objects. In: ECCV (2020)
61. Yang, B., Liang, M., Urtasun, R.: Hdnet: Exploiting hd maps for 3d object detection. In: Conference on Robot Learning. pp. 146–155 (2018)
62. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: Proceedings of the IEEE CVPR (2018)
63. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-end interpretable neural motion planner. In: Proceedings of the IEEE CVPR (2019)
64. Zeng, W., Wang, S., Liao, R., Chen, Y., Yang, B., Urtasun, R.: Dsdnet: Deep structured self-driving network. In: ECCV (2020)
65. Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in lidar point clouds. arXiv preprint arXiv:1910.06528 (2019)

Supplementary Material

In the following supplementary materials, we provide: additional discussions of our method in the broader context of implicit generative models in Section A, details about the datasets used in Section B, more implementation details in Section C, more in-depth evaluation details and quantitative results in Section D, and finally more visualizations in Section E.

A Further Discussion on Implicit Generative Models

A.1 Implicit Generative Models

Typical probabilistic models for motion forecasting define an *explicit* parameterized output distribution over each actor n and trajectory waypoints across time t, y_t^n. Examples are the methods proposed in [9,11,12], which parameterize their output distribution as a mixture of Gaussians, which can be sampled efficiently and provides a likelihood evaluation but assumes 1) independence across actors, and 2) a particular shape of the output distribution. In contrast, implicit generative models define a output distribution $p_\theta(Y)$ *implicitly* by specifying a latent distribution $p(Z)$ from which we can sample, followed by a mapping $f_\theta : \mathcal{Z} \to \mathcal{Y}$, which we refer to as the decoder.

In particular, we can characterize the decoder in two ways:

1. via a specified and tractable conditional likelihood $p_\theta(Y|Z)$. In this case, many tools are available for inference and learning. Variational inference, and in particular the variational auto-encoder (VAE) [30], is a common choice.
2. via a stochastic sampling procedure where $p(Y|Z)$ is not specified. In this case, likelihood-free inference methods are required for learning. Density estimation by comparison has been proposed [44] using either density ratio (GAN) or density difference (MMD). These methods, however, are generally more difficult to optimize.

In our model, we define f_θ as a deterministic function (parameterized by a graph neural network), since we wish the latent Z to capture all the uncertainty in a scene and have Y be deterministic given Z. To sidestep the difficulty of likelihood-free inference, particularly in a complex model where we optimize both perception and motion forecasting end-to-end, we make a mild assumption to leverage variational inference for learning. In the following sections, we provide a preliminary on variational inference before providing a detailed analysis of our model and learning approach.

A.2 Variational Inference

The variational auto-encoder (VAE) [29] specifies a directed graphical model with latent variables z and output variables y. Conditional variational auto-encoder (CVAE) [56] extends this formulation to the conditional generative setting, with additional input variables x. Now, for a given observation x, z is drawn

from the conditional prior distribution $p(z|x)$ and output y is generated from the distribution $p(y|x,z)$.

The learning objective for the conditional generative model is maximizing the conditional log likelihood $\log p(y|x)$. But since marginalizing over continuous latent variables z is intractable, it is typical to apply the Stochastic Gradient Variational Bayes (SGVB) framework and optimize the following evidence variational lower bound (ELBO) instead:

$$\log p(y|x) \geq \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(y|x,z)] - KL(q_\phi(z|x,y)||p_\gamma(z|x))$$

Here, $q_\phi(z|x,y)$ is the learned approximate posterior, $p_\gamma(z|x)$ the learned approximate prior, and $p_\theta(y|x,z)$ the learned decoder.

Followup works have proposed further modifications to this objective to encourage disentanglement, prevent posterior collapse, and improve training stability. In this work, we follow [19] in extending the ELBO objective with an additional hyperparameter β:

$$L_{ELBO} = -\mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(y|x,z)] + \beta KL(q_\phi(z|x,y)||p_\gamma(z|x))$$

A.3 Analysis of Our Method

We recall that to capture the joint distribution over all the actor trajectories we employ a deterministic decoder $Y = f(X,Z)$, letting the latent variable Z capture all the stochasticity. Thus, instead of optimizing the likelihood based reconstruction objective that appears in the ELBO, we opted for a Huber loss on the trajectory waypoints. This choice can be interpreted as an assumption of $p_\theta(Y|X,Z)$ being a Gaussian/Laplacian with fixed diagonal covariance. For simplicity, let's assume our Huber loss is always active within the L2 segment, but the following derivation could be easily done with a Laplacian as well. In this view, we can further interpret β as the variance of the underlying Gaussian, as follows:

$$\begin{aligned}
\mathcal{L}_{forecast} &= ||Y - Y_{GT}||_2^2 + \beta KL(q_\phi(Z|X,Y)||p_\gamma(Z|X)) \\
&\propto \frac{1}{\beta}||Y - Y_{GT}||_2^2 + KL(q_\phi(Z|X,Y)||p_\gamma(Z|X)) \\
&\propto \mathbb{E}_{q_\phi(Z|X,Y)}\left[\log \mathcal{N}(Y_{GT}|f_\theta(X,Z), \frac{\beta}{2}I)\right] + KL(q_\phi(Z|X,Y)||p_\gamma(Z|X))
\end{aligned}$$

To see this, recall the log likelihood of Gaussian with diagonal covariance:

$$\log \mathcal{N}(Y_{GT}|\mu, \sigma^2 I) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{t=1}^{T}(y_{n,GT}^t - \mu^t)^2 - c_\sigma$$

$$\log \mathcal{N}(Y_{GT}|\mu, \sigma^2 I = \frac{\beta}{2}I) = -\frac{1}{\beta}\sum_{n=1}^{N}\sum_{t=1}^{T}(y_{n,GT}^t - \mu^t)^2 - c_\sigma$$

where c_σ can be neglected since it is constant relative to μ, and thus does not contribute to its gradient.

Empirically we found $\beta = 0.05$ to yield the best results. Under the interpretation above, this would translate into using a fixed variance of $2.5cm$ while learning our model. We note that this is extremely small in the context of motion forecasts (where vehicles can easily travel 50 meters in 5 seconds), and thus consistent with our goal of approximating Y as a deterministic mapping from X and Z, letting Z capture (nearly) all the uncertainty at a scene level.

B Datasets

We benchmark our approach on two datasets: ATG4D [62] and nuScenes [8]. This allow us to test the effectiveness of our approach in two vehicle platforms with different LiDAR sensors and maps, driving in multiple cities across the world.

ATG4D Our dataset contains more than one million frames collected over several cities in North America with a 64-beam, roof-mounted LiDAR. Our labels are very precise 3D bounding box tracks with a maximum distance from the self-driving vehicle of 100 meters. There are 6500 snippets in total, each 25 seconds long. In each city, we have access to high definition maps capturing the geometry and the topology of each road network. Following previous works in joint perception and motion forecasting [9, 10, 40] we consider a rectangular region centered around the self-driving vehicle that spans 144 meters along the direction of its heading and 80 meters across. In these experiments, the model is given one second of LiDAR history and has to predict 5 seconds into the future.

nuScenes This dataset consists of 1,000 snippets of 20 seconds each, collected in Boston and Singapore (right-side vs. left-side driving). Their 32-beam LiDAR captures a sparser point cloud than the one in ATG4D, making object detection more challenging. High definition maps are also provided. We use the evaluation setup proposed in their perception benchmark, where the previous 10 LiDAR sweeps (0.5 seconds) are fed to the model, and the region of interest is a circle of 50 meters radius around the SDV. The prediction horizon is 5 seconds.

C Implementation Details

In this section, first we provide implementation details about each component in our joint perception and motion forecasting model. We then discuss the required adaptations for the baselines.

C.1 ILVM Details

LiDAR Pointcloud Parameterization: Following [62], we use a voxelized representation of the 3D LiDAR point cloud in Bird's Eye View (BEV) as

the main input to our model. As in its follow-up work [61], we normalize the height dimension with dense ground-height information provided by HD maps for ATG4D dataset only (nuScenes does not provide this information). To exploit motion cues, we leverage multiple LiDAR sweeps by compensating the ego-motion (i.e. projecting the past sweeps to the coordinate frame of the current sweep), as proposed by [40]. Following [10], we ravel the height and time dimension into the channel dimension, to use 2D convolution to process spatial-temporal information efficiently. The final representation is a 3D occupancy tensor of dimensions $(\frac{L}{\Delta L}, \frac{W}{\Delta W}, \frac{H \cdot T}{\Delta H \cdot \Delta T})$. Here, $L = 144$, $W = 80$, and $H = 5$ are the spatial dimensions in meters. $\Delta L = \Delta W = \Delta H = 0.2$ m/pixel are the resolutions for the spatial dimensions, $T = 5$ seconds is the prediction horizon, and $\Delta T = 0.5$ seconds/time-step is the time resolution.

High-Definition Maps Parameterization: We use a rasterized map representation encoding traffic elements such as intersections, lanes, roads, and traffic lights. Elements with different semantics are encoded into different channels in the raster, as proposed by [10].

The map elements we rasterize are the following: drivable surface polygons, road polygons, intersection polygons, vehicle lane polygons going straight, dedicated left and right vehicle lane polygons, dedicated bike lane polygons, dedicated bus lane polygons, centerline markers for all lanes, lane dividers for all lanes with semantics (allowed to cross, not allowed to cross, might be allowed to cross). This gives us a total of 13 different map channels combining these elements.

Shared Perception Backbone: We use a lightweight backbone network adapted from [62] for feature extraction. In particular, we instantiate two separate streams such that the voxelized LiDAR and rasterized map are processed separately first. The resulting features from both streams are then concatenated feature-wise since they share the same spatial resolution, and finally fused by a convolutional header. Our LiDAR backbone uses 2, 2, 3, and 6 layers in its 4 residual blocks. The convolutions in the residual blocks of our LiDAR backbone have 32, 64, 128 and 256 filters with a stride of 1, 2, 2, 2 respectively. The backbone that processes the high-definition maps uses 2, 2, 3, and 3 layers in its 4 residual blocks. The convolutions in the residual blocks of our map backbone have 16, 32, 64 and 128 filters with a stride of 1, 2, 2, 2 respectively. For both backbones, the final feature map is a multi-resolution concatenation of the outputs of each residual block, as explained in [15]. This gives us 4x down-sampled features with respect to the input. The header network consists of 4 convolution layers with 256 filters per layer. We use GroupNorm [59] because of our small batch size (number of frames) per GPU. These extracted features inform both the downstream detection and motion forecasting networks, explained next.

Object Detection: We use two convolutional layers to output a classification (i.e. confidence) score and a bounding box for each anchor location following the output parameterization proposed in [62], which are finally reduced to the

final set of candidates by applying non-maximal suppression (NMS) with an IoU of 0.1, and finally thresholding low probability detections (given by the desired common recall).

Actor Feature Extraction: To arrive at the final actor level features x_n, we apply rotated ROI Align [22] to extract fixed size spatial feature maps for bounding boxes with arbitrary shapes and rotations from our global feature map extracted by the backbone. We pool a region around each actor in its frame with an axis defined by the actor's centroid orientation. The region in BEV space spans for 10m backwards, 70m in front, and 40m to both sides of the actor. After applying the rotated ROI Align operator, we get a feature map for each actor of size 40 x 40 x 256. We then apply a 4-layer down-sampling convolutional network followed by max-pooling along the spatial dimensions to reduce the feature map to a 512-dimensional feature vector per actor. The convolutional network uses a dilation factor of 2 for the convolutional layers to enlarge the receptive field for the per-actor features, which we found to be important. We use ReLU as the non-linearity and GroupNorm for normalization.

Scene Interaction Module: Our scene interaction module is inspired by [9], and is used in our Prior, Encoder, and Decoder networks. Our edge or message function consists of a 3-layer MLP that takes as input the hidden states of the 2 terminal nodes at each edge in the graph at the previous propagation step as well as the projected coordinates of their corresponding bounding boxes. We use feature-wise max-pooling as our aggregate function in order to be more robust to changes in the graph topology between training and inference, since at training we use the ground-truth bounding boxes but at inference employ the detected bounding boxes. To update the hidden states we use a GRU cell. Finally, to output the results from the graph propagations, we use a 2-layer MLP.

Motion Forecasting: The inference of our motion forecasting model is explained step-by-step in Algorithm 2. Our Prior_γ and Encoder_ϕ modules are both composed of 2 SIMs with different parameters, one that predicts the latent means Z_μ and one that predicts the latent sigmas Z_σ. We use the same input, hidden, and output dimension of 64 for these SIMs. To obtain the 64-dimensional input H^0 to the $Prior_\gamma$ SIMs, we use a 2-layer MLP to summarize the 512-dimensional actor feature X. For Encoder_ϕ, we use an additional 2-layer MLP to embed the ground truth future trajectories Y_{GT} into a 64-dimensional embedding first, then summarize the concatenated 576-dimensional vector into the 64-dimensional input H^0 to the SIMs. The Decoder_θ is implemented with a single SIM, which takes a 576-dimensional input H^0 (i.e. direct concatenation of 512-dimensional actor features X and 64-dimensional latent sample Z^s), and outputs a 20-dimensional vector Y^s (i.e. 10 waypoints with (x, y) coordinates) for each actor. Although we have described the algorithm as sequential over scenes $1 \dots S$ for clarity in the algorithm, the sampling and decoding of all scenes can be done in parallel.

Algorithm 2 Motion Forecasting

Input: Actor features $X = \{x_1, x_2, \cdots, x_N\}$. BEV locations of object detections $C = \{c_0, c_1, ..., c_N\}$ Number of scene samples to generate S.

Output: Scene trajectory samples in bird's-eye-view space $\{Y^1, Y^2, \cdots, Y^S\}$, where $Y^s = \{y_1^s, y_2^s, ..., y_N^s\}$ (N is the number of detected actors).

$\{Z_\mu, Z_\sigma\} \leftarrow \text{Prior}_\gamma(X, C)$ ▷ Use SIM modules to output latent distribution

for $s = 1, ..., S$ **do** ▷ Run for all requested number of samples

  $Z^s \sim \mathcal{N}(\{Z_\mu, Z_\sigma \cdot I\})$ ▷ Sample a scene latent from diagonal gaussian

  $H^s = \{\text{MLP}(x_n \oplus z_n^s) : \forall n \in 1 \dots N\}$

  $Y^s = \text{Decoder}_\theta(H^s, C)$ ▷ Use SIM module to decode trajectory sample

return $\{Y^1, Y^2, \cdots, Y^S\}$

Optimization Details: We use the Adam optimizer [28] with an initial learning rate of 1.25e-5 and no weight decay. To weigh the multi-task objective, we use $[\alpha, \lambda, \beta] = [0.1, 0.5, 0.05]$. We follow [16] in using a cyclic annealing schedule for β. More specifically, we perform warmup for 40k steps in 10k step cycles.

C.2 Baseline Details

Here we provide the implementation details behind how we updated our baseline models to meet our perception and prediction setting. There are basically two options for comparison:

(a) use an off-the-shelf detector and tracker to provide past trajectories, or
(b) replace their past trajectory encoders by our backbone and per actor feature extraction.

SpAGNN [9] showed that (a) using an off-the-shelf tracker (Unscented Kalman Filter + Hungarian matching) results in much worse performance than option (b), so we stick to the latter for a fair comparison where all methods use the same architecture to extract actor features X from sensor data, which is trained end-to-end with the motion forecasting module for each baseline.

Explicit Marginal Likelihood Models: SpAGNN [9] was originally proposed in the joint perception and prediction setting and therefore does not require any adaptation. We adapt MTP [12] and MultiPath [11] to use our backbone network, object detection and per actor feature extraction and then apply their proposed mixture of trajectories output parameterization, where each way-point is a gaussian.

A detail worth noting is that these baselines do not propose a way to get temporally consistent samples, since the gaussians are independent across time (the models are not auto-regressive). Thus, we introduce a heuristic sampler to get temporally consistent samples from this model. The sampled trajectories are extracted using the re-parameterization trick for a bi-variate normal:

$$y_{n,t}^s = \mu_{n,t} + A_{n,t} \cdot \varepsilon_n^s$$

where the model predicts a normal distribution $\mathcal{N}\left(y^s_{n,t}|\mu_{n,t}, \Sigma_{n,t}\right)$ per waypoint t, $(A_{n,t})^T \cdot A_{n,t} = \Sigma_{n,t}$ is the cholesky decomposition of the covariance matrix, and $\varepsilon^s_n \sim \mathcal{N}(0, I)$ is the noise sampled from a standard bi-variate normal distribution. Note that the noise ε^s_n is constant across time t for a given sample s and actor n. Intuitively, having a constant noise across time steps allows us to sample waypoints whose relative location with respect to its predicted mean and covariance is constant across time (i.e. translated by the predicted mean and scaled by the predicted covariance per time).

Autoregressive Models While many papers that utilize auto-regressive models [50,51,57] use fully observed states as dynamic inputs, we extend these models to the joint detection and motion forecasting task. For all auto-regressive models, we use the *detection backbone* and the *actor feature extraction* modules that we use for all models, including ours.

Due to the compounding error problem [53] found in auto-regressive models, we had to make some adjustments to the training procedure to account for the noise in the $t-1$ conditioning space. Typically during training for auto-regressive models, a one-step prediction distribution given the previous ground-truth value $p\left(y_{t+1}|x, y_{t,GT}\right)$ is learned. This can cause a catastrophic mismatch between the input distribution that the model sees during training and that it sees during inference. To help simulate the noise it sees during inference, we add gaussian noise to the conditioning state $\tilde{y} = y_{GT} + \epsilon$ where $\epsilon \sim \mathcal{N}\left(0, I \cdot \alpha\right)$. The parameter α defines to the amount of noise we expect in meters between time-steps (we use a value of 0.2m in our experiments.) We note that we also tried scheduled sampling [5], but adding white noise worked better.

For ESP [51], we extended our *R2P2-MA* implementation with the "whisker" indexing technique to get added context into the feature map at the location of the conditioning state $\tilde{y}_t$. Due to memory constraints, we had to limit the radii of the whiskers to $[1m, 2m, 4m]$ while keeping the seven angle bins. We also reproduced the social context conditions but with a minor modification. While the original paper specified a fixed number of actors, we used k-nearest neighbors to select a set amount of $M = 4$ neighbors to gather social features and model the distribution $p\left(y_{n,t+1}|x, y_{n,t}, y_{1,t}, y_{2,t}, \cdots, y_{M,t}\right)$. Lastly, We note that the originally proposed SocialLSTM and NRI do not leverage any sensor or map data, but since we share the feature extraction architecture for all models, their adaptations do have access to these cues making their methods more powerful than originally proposed.

D Additional Evaluation Details and Results

In this section, we present additional evaluation details and quantitative results on *detection* and *motion forecasting*.

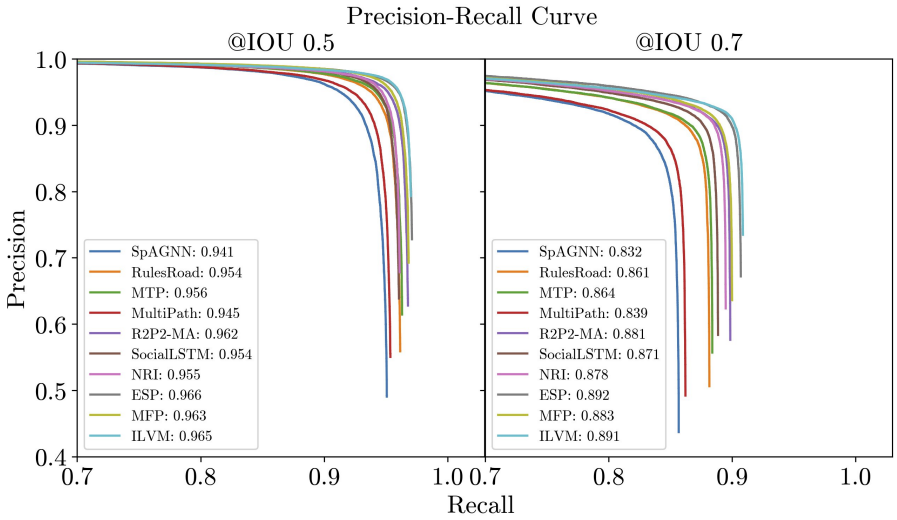


Fig. 7: Precision-Recall curve at IoU 0.5 and 0.7. Legend shows mAP (mean Average Precision) for each model. Note: horizontal axis starts at 0.7 recall.

D.1 Detection

Fig. 7 shows that our model achieves the best detection performance at both IoU thresholds. Since all models have the same backbone and detection header, we conjecture that our learning objective eases the joint optimization of both detection and motion forecasting.

D.2 Motion Forecasting

For ATG4D experiments, we operate the object detector at a 90% common recall point. In nuScenes experiments, we operate the object detector at an 80% common recall point, since detection is more challenging in this dataset due to the sparser 32-beam LiDAR sensor (as opposed to 64-beam in ATG4D).

Diversity vs. Precision in multimodal prediction Fig. 8 showcases three different predictions that exhibit different qualities, which we use to illustrate the language used throughout the paper. On the left we show an accurate model that can nicely capture the bimodal distribution due to the branching map topology. On the middle, a prediction model predicts high diversity samples (high recall), but has low precision as it predicts unrealistic samples that are out of distribution. On the right, we show a high-precision prediction, meaning that all the samples are within the true data distribution, but low recall or diversity, meaning that it misses some modes of the ground-truth distribution.

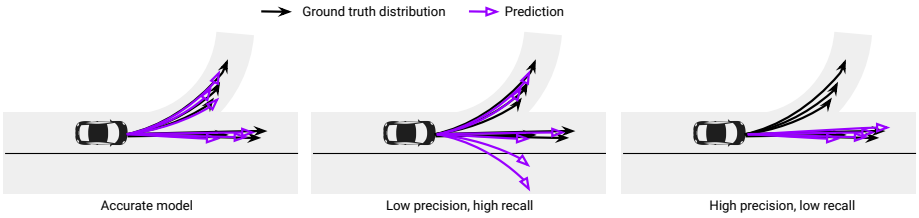


Fig. 8: Diversity vs. Precision for an individual actor

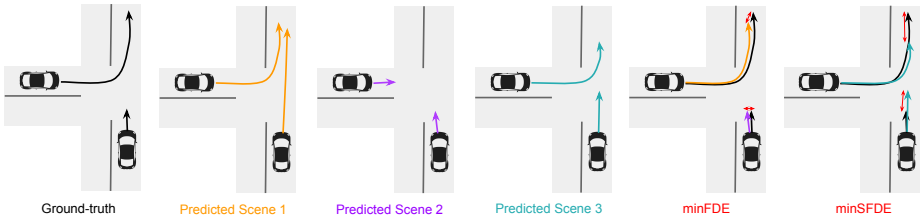


Fig. 9: Coverage metrics: actor-level (minFDE) vs. scene-level (minSFDE). The first column shows the groun-truth trajectories, the next 3 are possible futures predicted by the model, the last 2 show the trajectory samples selected by each metric (which are different), as well as their error (red arrows).

Actor-level vs. Scene-level metrics Fig. 9 motivates the need for scene-level metrics to evaluate the characterization of the joint distribution over actors. In particular, minFDE (actor-level minimum displacement error) will take the minimum error trajectory for each actor regardless of which scene prediction it belongs to. In contrast, minSFDE (our proposed scene-level counterpart) takes the trajectories from the predicted scene with less average error across vehicles, thus selecting the scene that is most consistent with the ground-truth as a whole.

Scene Consistency – Sample Collisions Here, we demonstrate that our models produce scene-level samples that are more socially consistent regardless of which recall point we operate our object detector. Figure 10 shows our Scene Collision Rate (SCR) at different detection recall points (also known as operating point). As the recall point is chosen to be higher and higher there are more low probability actors in the scene which greatly increases the chances of a predicted collision, as expected. When analyzing the results, it is clear that just sharing social features as [9] does is not enough to create scene consistent samples. Models that do joint sampling such as ILVM and ESP [51] do markedly better on this measure. Interestingly, ILVM barely sees an increase in the amount of collisions as recall increases, which shows that our model is able to generate scene consistent samples no matter the complexity of the scene.

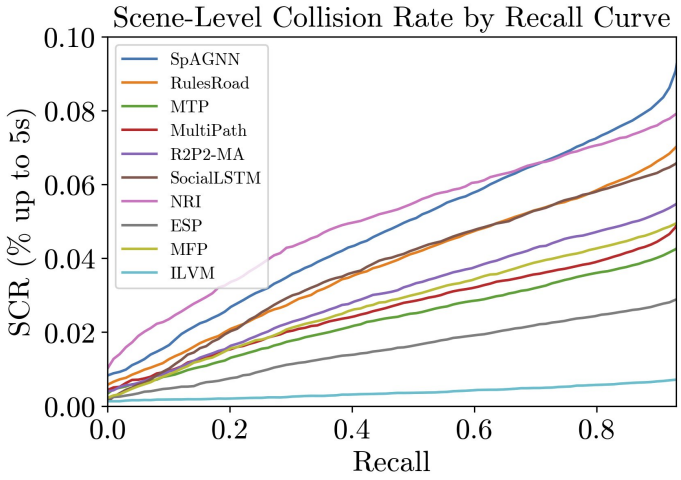


Fig. 10: ILVM models social interaction consistently well at different recall levels.

Sample Quality – Cumulative Hit Rate: So far in the motion forecasting literature, actor-level precision (meanFDE) and recall (minFDE) metrics of the trajectory samples have been proposed, but no attempt has been made to combine them in a single metric, despite the fact that the previous metrics have evident drawbacks. For instance, meanFDE disregards the fact that multiple plausible futures could be very far apart and overly penalizes multi-modality, while a low minFDE can be achieved by just predicting fanned out distributions that cover big spaces. Here, we propose to use a cumulative Hit Rate curve, where the horizontal axis corresponds to the L2 error threshold, and the vertical axis to the percentage of samples that fall under such error. Moreover, we extend this notion to also capture failures in the object detector, by considering that false positive and false negative detections always have error higher than the threshold, thus obtaining a holistic metric for joint perception and prediction. We now define this metric mathematically. We use $\hat{y}$ to denote the ground truth future y_{GT}.

$$\text{Hit Rate}\left(y, \hat{y}, t, \epsilon\right) = \frac{1}{NS} \sum_{n=1}^{N} \sum_{s=1}^{S} \text{Hit}\left(y_n^{t,s}, \hat{y}_n^t, y_n^0, \hat{y}_n^0, \epsilon\right)$$

$$\text{Hit}\left(y_n^{t,s}, \hat{y}_n^t, y_n^0, \hat{y}_n^0, \epsilon\right) = \begin{cases} 1 & \text{if IoU}\left(y_n^0, \hat{y}_n^0\right) > 0.5 \text{ and } ||y_n^{t,s} - \hat{y}_n^t||_2 < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

Thus Hit Rate finds the percentage of samples that are true positive detections and have an L2 error below a threshold ϵ. We sweep ϵ values of 0.0m to

5.0m to get the broader curve which gives us the distribution on how likely each model is to get a detection and sample close to the ground-truth. We do not compute the metric above 5 meter error since we consider that to always be a bad sample or "miss". Fig. 11 shows that Our ILVM significantly outperforms all baselines in cumulative hit rate across across all time steps.

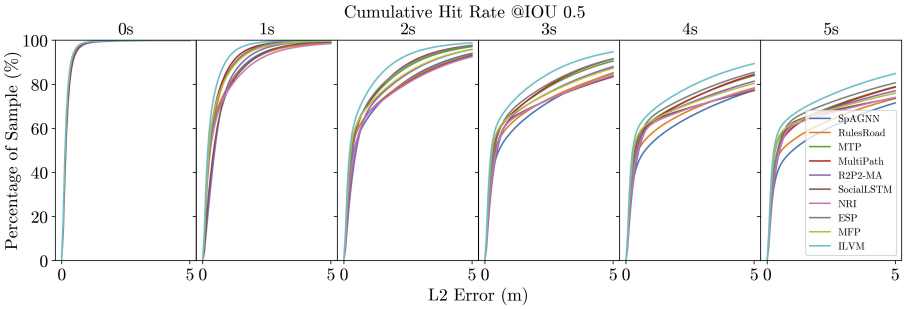


Fig. 11: ILVM obtains the best hit rate at all time-steps in the prediction horizon.

Sample Quality – Breakdown We define Along-Track and Cross-Track distance as the longitudinal and lateral distance after projecting motion forecasts into the ground-truth actor trajectory coordinate. This breakdown is important, since lateral error is semantically more significant than longitudinal error for the downstream task of ego-motion planning.

Fig. 12 provides an in-depth analysis of the sample quality of our motion forecasts by examining the error breakdown between Along-Track and Cross-Track. Furthermore, we highlight the robustness of our model in recovering ground-truth scenes when given different number of samples.

The results showcase that model rankings may not be consistent in the breakdown of Along-Track vs. Cross-Track. In particular, we find the main contributor to ILVM's advantage is better Along-Track forecast, while having equal or better Cross-Track. This implies that our model is able to better estimate overall current and future velocity of the actors while having the same precision on their path as ESP, and significantly better than the other baselines.

Sample Quality – Precision Diversity Tradeoff In Fig. 13, we showcase the progression of scene-level sample quality metrics of Our ILVM during training. While the precision metric (meanSFDE) continues to improve past 50k iterations, the diversity metric (minSFDE) reaches its optimum. This sheds light on the inherent tradeoff between the diversity and precision aspect of sample quality measure, particularly when we only have access to a single ground truth realization of the multiple plausible futures.

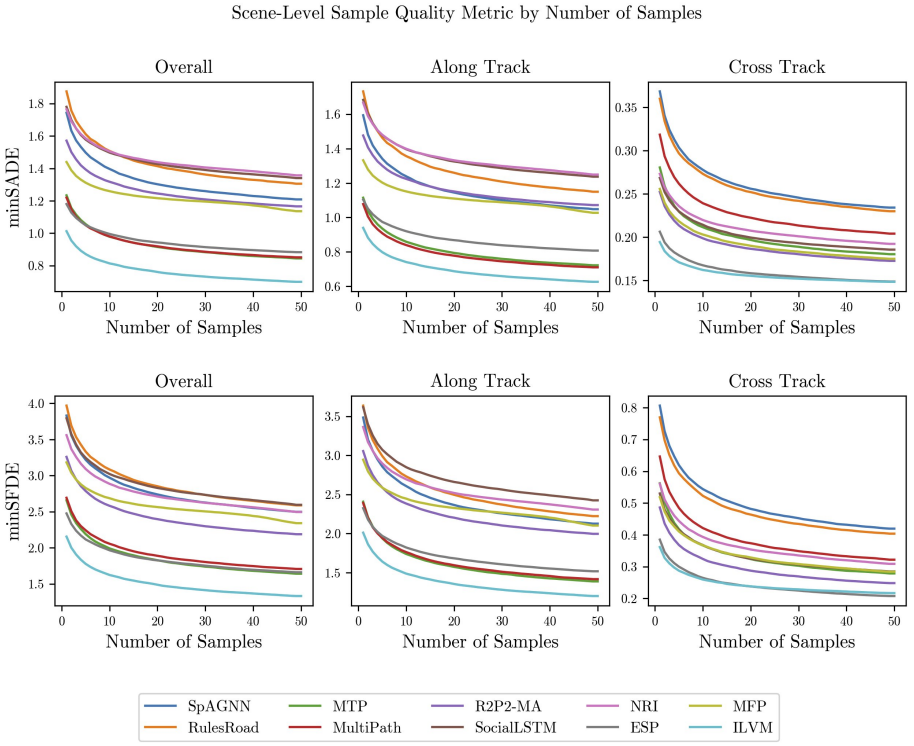


Fig. 12: Our ILVM outperforms more significantly in along track error than cross track error for both scene-level minADE and minFDE metrics.

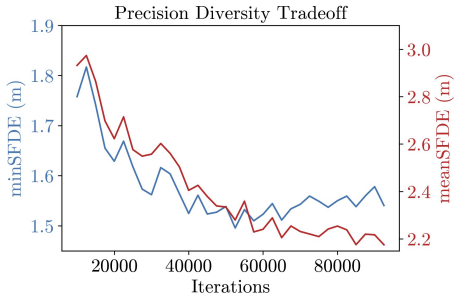


Fig. 13: Tradeoff between minSFDE and meanSFDE as model training progresses. Note that the *minSFDE curve* follows the y-axis on the left, and the *meanSFDE curve* follows the y-axis on the right.

Ablation – KL Term We include an analysis of how the *beta* weight on the KL term trades off diversity and precision in Fig. 5.

We observe that:

β	SCR_{5s} (%)	min SFDE (m)	min SADE (m)	mean SFDE (m)	mean SADE (m)
0.01	1.41	1.70	0.84	2.58	1.17
0.03	0.89	1.59	0.79	2.37	1.06
0.05	0.70	**1.53**	**0.76**	2.27	1.02
0.5	**0.64**	1.85	0.85	**1.90**	**0.86**
1	0.64	1.87	0.87	1.95	0.88

Table 5: **[ATG4D] KL loss ablation study**

1. *High beta*: model loses multimodality and predicts a single future without variance. Low recall (minSADE) and high precision (meanSADE, collision). High KL loss constrains the posterior to be close to the prior, thus limiting the flexibility to encode useful information.
2. *Low beta*: model produces very high entropy distributions that try to cover all possible futures at the expense of producing unrealistic samples. High recall (minSADE) and low precision (meanSADE, collision). Low KL loss allows the posterior to diverge from the prior, which creates a gap between training and inference. Then at inference, the decoder struggles to interpret latent samples from the prior distribution, which its not trained on.

E Additional Visualizations

Scene Consistency: In Figure 14, 15, 16, 17 we showcase the scene consistency of the samples generated from our model. For these visualizations, each row corresponds to a model, and we show 2 scene-level samples for each model to characterize the joint distribution.

More concretely, we show the two *most distinct* samples by averaging the pairwise Euclidean distance between all samples. We empirically find that this selection methodology yields representative samples and insight into how well the models learn scene-level social interaction between agents.

Latent Space Interpolation: In Figs. 18 and 19, we take the 2 most distinct samples as in the previous scene sample visualizations, and show the resulting futures when performing linear interpolation in the latent space. We show that the interpolated latent points still produce semantically meaningful trajectories for all the actors in the scene, and capture scene level variations including multi-agent interactions. More precisely, Z^1 and Z^2 are the latent samples that map into the most distinct futures out of 50. The rows in between correspond to the linear interpolation of the latent space, and different columns to different scenarios.

Overall Sample Quality: In Figs. 20, 21, 22, we show additional qualitative results for motion forecasting, comparing our method to the baselines in a wide

range of urban scenarios, one per column. We blend 50 scene sample trajectories with transparency. Time is encoded in the rainbow color map ranging from red (0s) to pink (5s). This can be seen as a sample-based characterization of the per-actor marginal distributions. We can see that our method generally produces more accurate and less entropic distributions that better understand the map topology and multi-agent interactions.

Ego-motion Planning: In Figs. 24, 23, 25, we show qualitative comparison between ego-motion planning open-loop results when using motion forecast from some of the strongest baselines and our model. Open-loop means that the SDV acts as if it does not receive new sensor information for the future horizon of 5 seconds, and thus needs to rely completely on the motion forecasts at the start of these scenarios. Thus, many of the collisions on these results could be potentially avoided by obtaining more accurate information in subsequent time steps and re-planning, but closed-loop experiments are out of scope of this paper.

The predicted bounding box samples into the future for other traffic participants are shown in yellow. The ground-truth future trajectories are shown in white if not in collision with the ground-truth SDV trajectory (shown as an empty black box) and in red if colliding with the SDV plan. Overall, we can see how ego-vehicle harmful events are avoided with more precise motion forecasts from our model. In particular, we observe that the main reason the baseline motion forecasting models tend to cause more collisions than our predictions is because the entropy of their distributions is too high, leaving the motion planner no space to plan a safe trajectory.

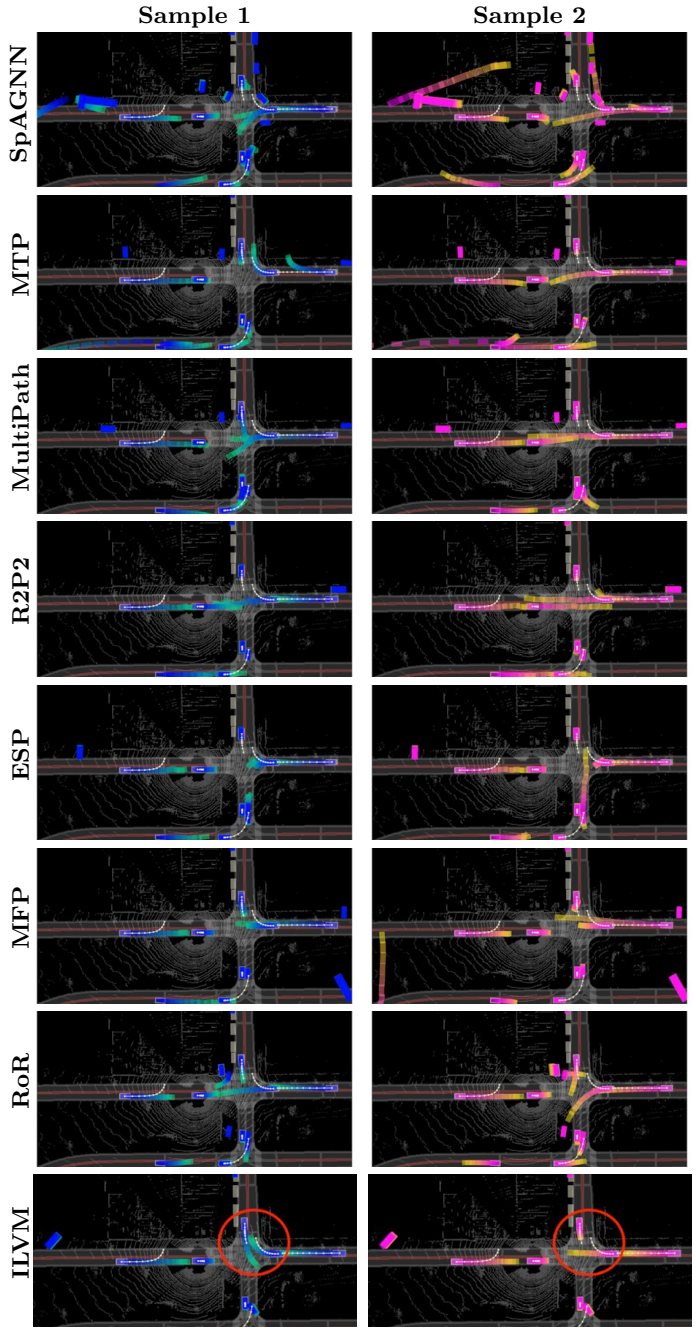


Fig. 14: **Scene-level samples**. Our latent variable model captures complex interactions at intersections. In this example, the car facing south will yield/go if the car facing west goes straight/turns right, respectively. The baselines do not capture this complex interaction, and most show inconsistent (colliding) samples for the 2 highlighted actors.

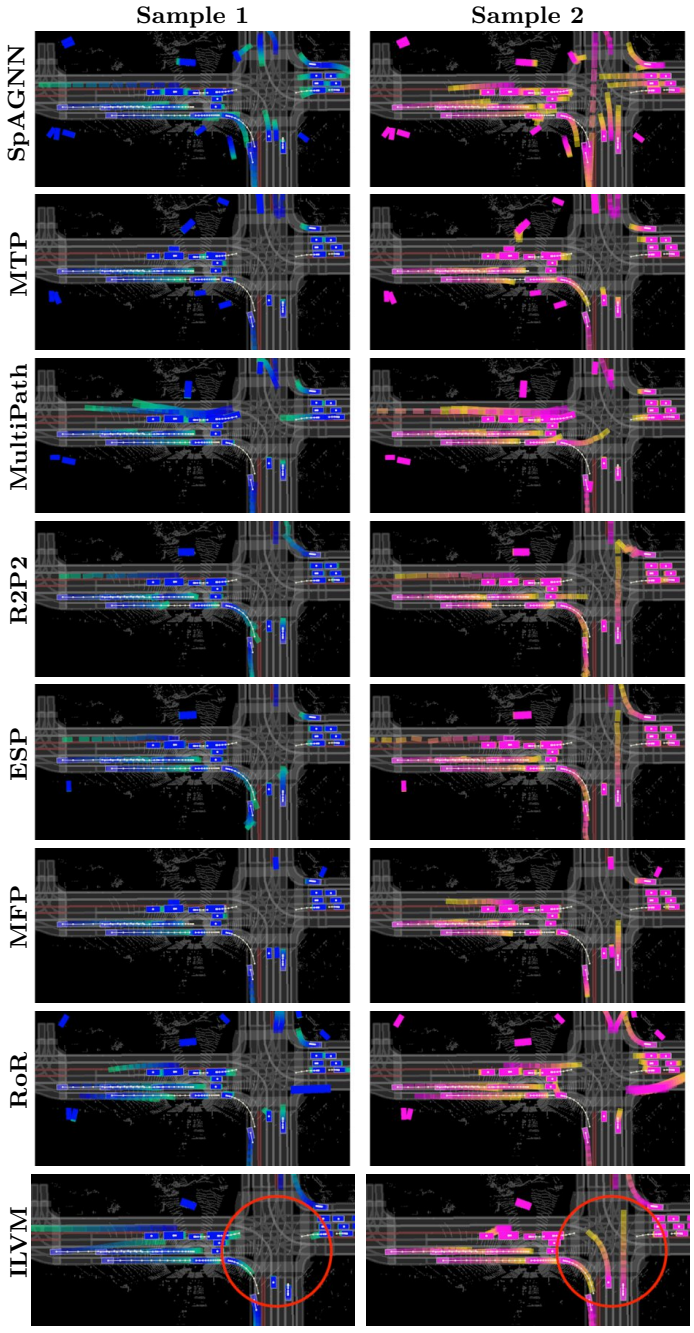


Fig. 15: **Scene-level samples**. Our latent variable model captures the different scene outcomes for possible states of a given traffic light intersection (vertical vs. horizontal traffic).

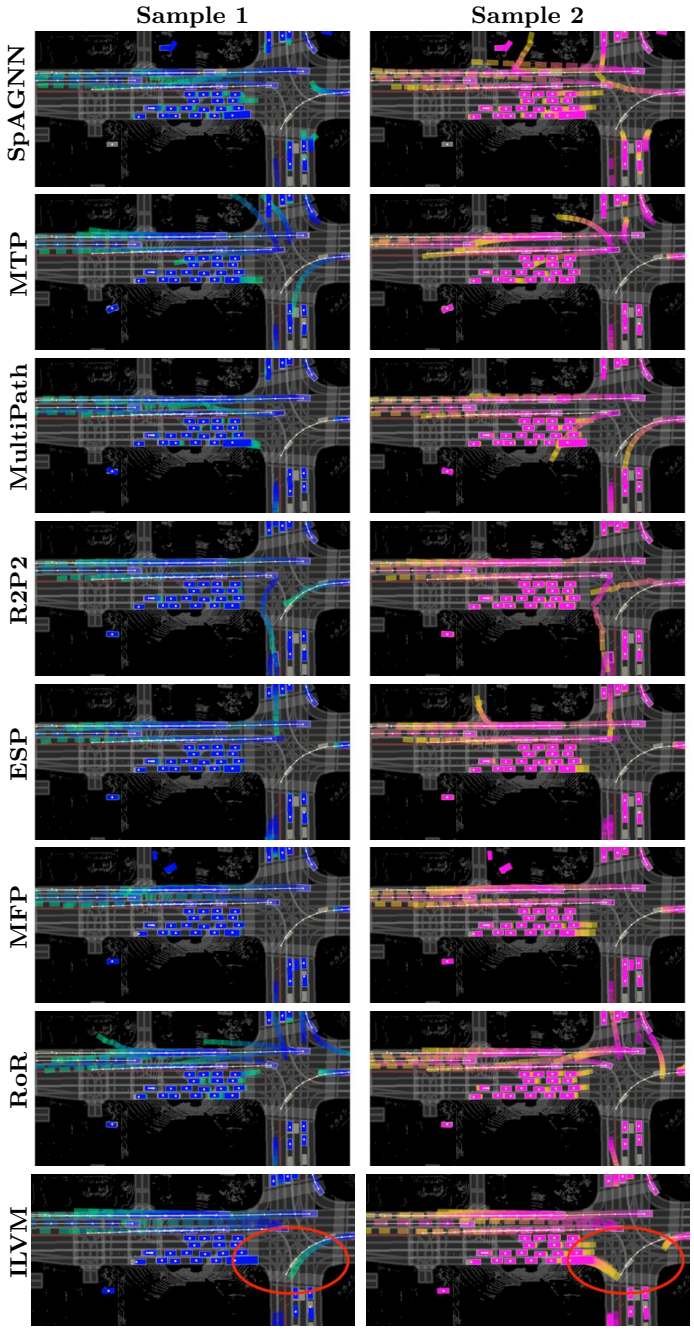


Fig. 16: **Scene-level samples**. Our latent variable model captures whether the bus will proceed with the right turn, or the left-turning vehicle will.

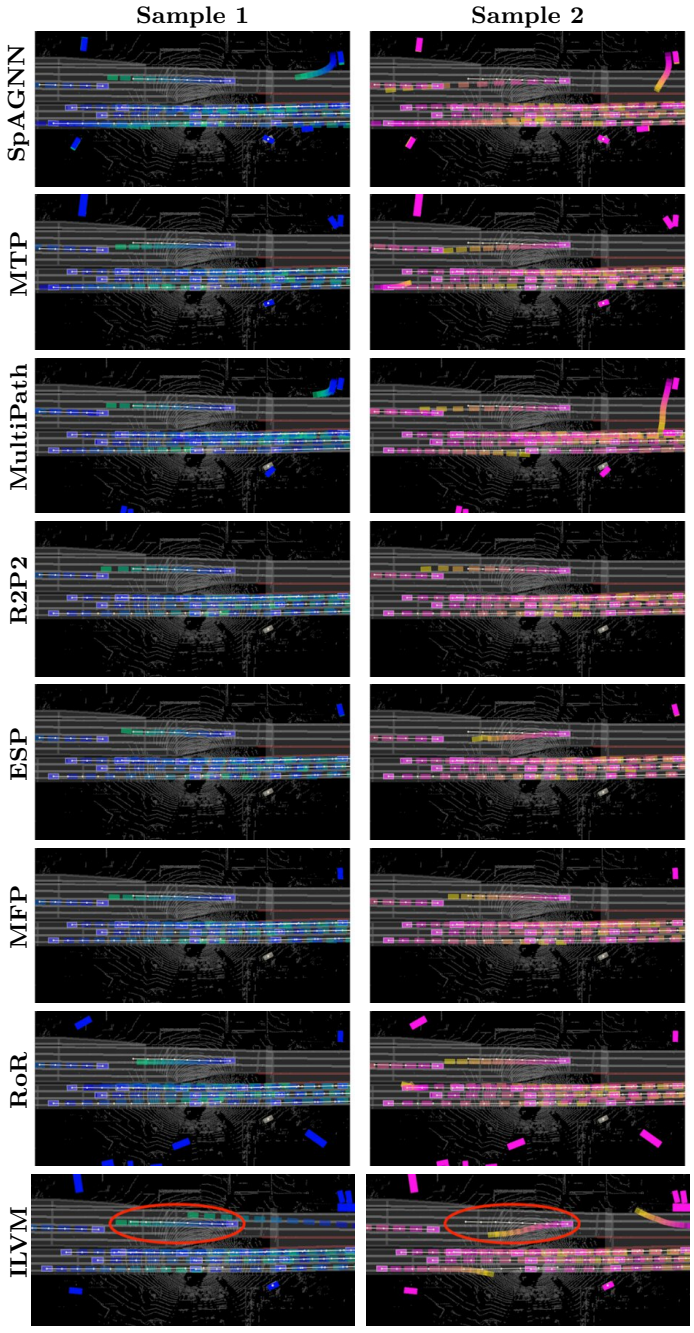


Fig. 17: **Scene-level samples**. Our latent variable model captures multiple realistic futures (including lane changes) that respect the map geometries and are dynamically feasible.

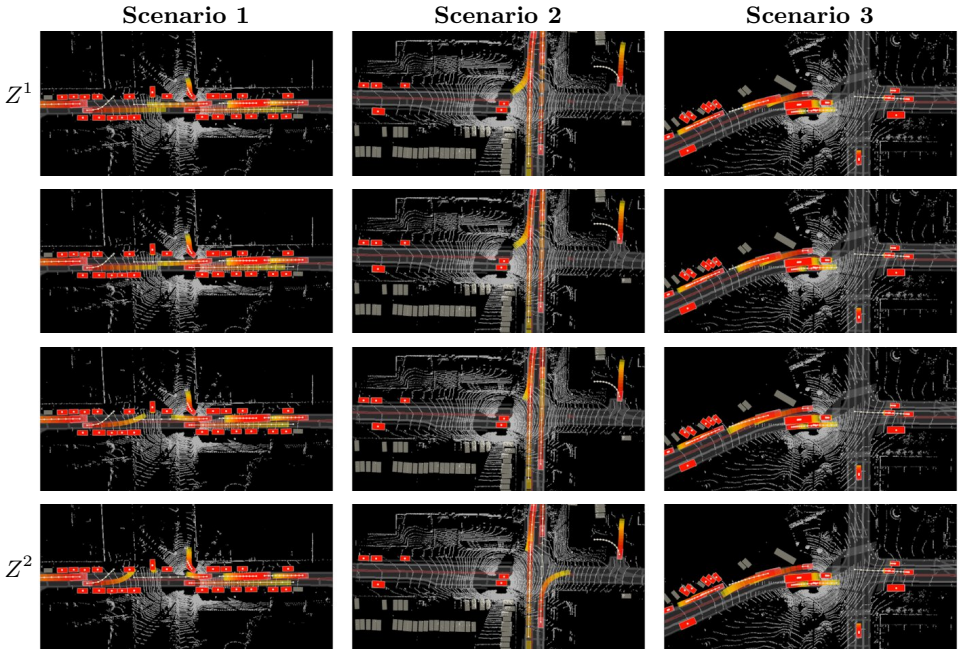


Fig. 18: **Latent space interpolation:** *Scenario 1* showcases a complex interaction between 3 vehicles: when the 2 vehicles in the road predict turning or slow moving trajectories, the third vehicle pulls out of the driveway, and when the 2 vehicles in the road keep constant velocity to go straight, the vehicle in the driveway yields. *Scenario 2* turning vs. going straight behavior with smooth transitions. *Scenario 3* we can see how the speed of 2 cars that follow each other vary consistently.

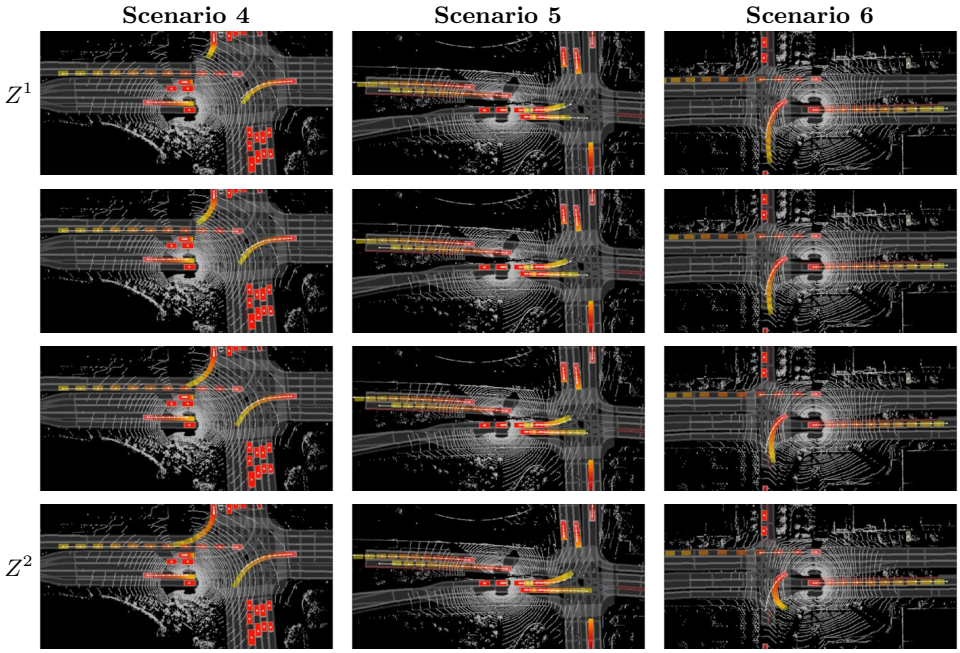


Fig. 19: **Latent space interpolation:** *Scenario 4* and *Scenario 5* showcase smooth transitions between different speed profiles when turning and going straight at an intersection. *Scenario 6* we can see all the range of possibilities from a left-turn to a u-turn, which is a pretty rare event.

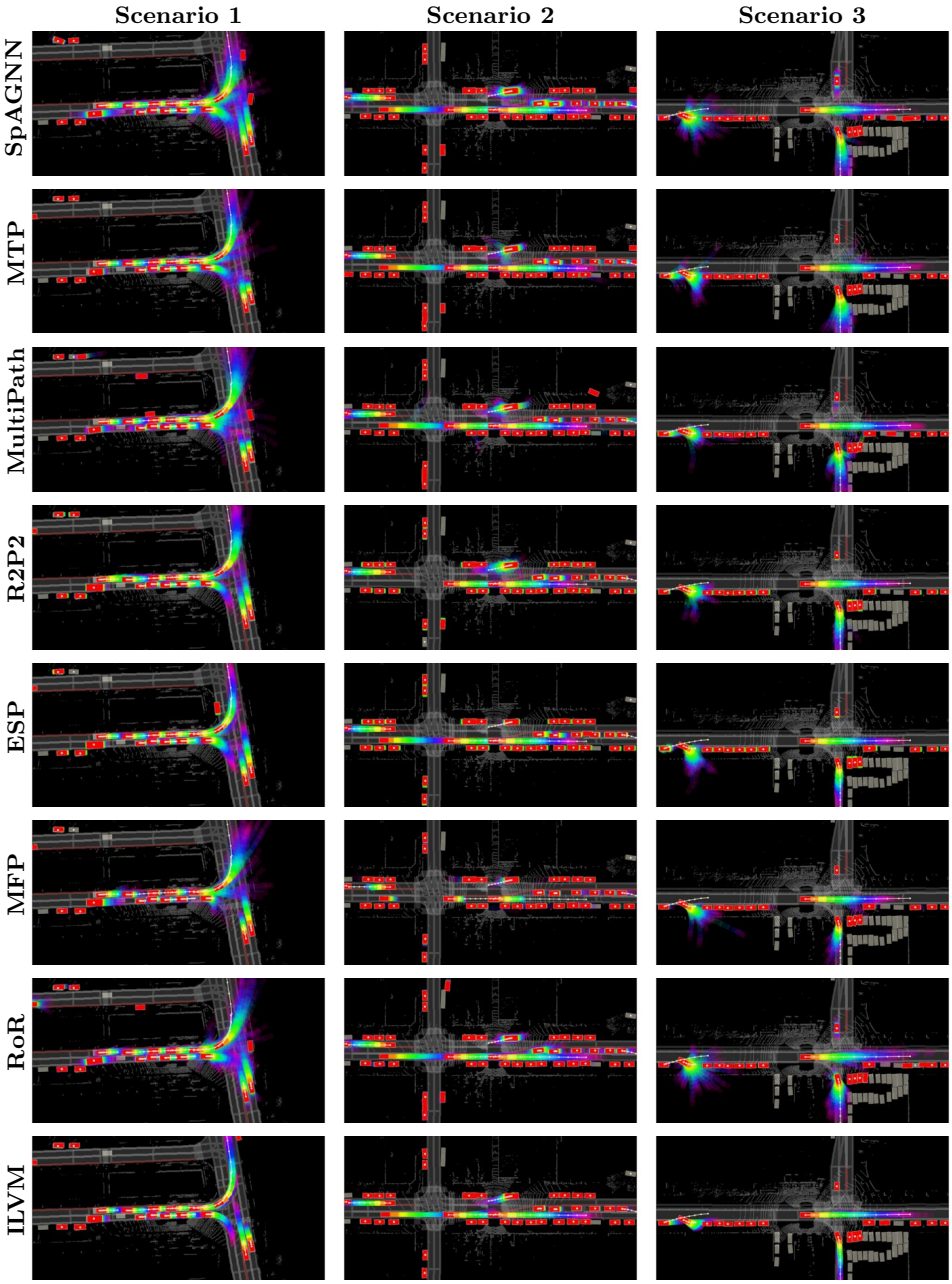


Fig. 20: **Overall Sample Quality:** *Scenario 1* showcases a T-intersection with fast-moving turns. *Scenario 2* is interesting because there is a vehicle coming out of a parking spot, which is not very frequent in driving logs. *Scenario 3* captures a vehicle maneuvering into a parking spot, also an unusual event.

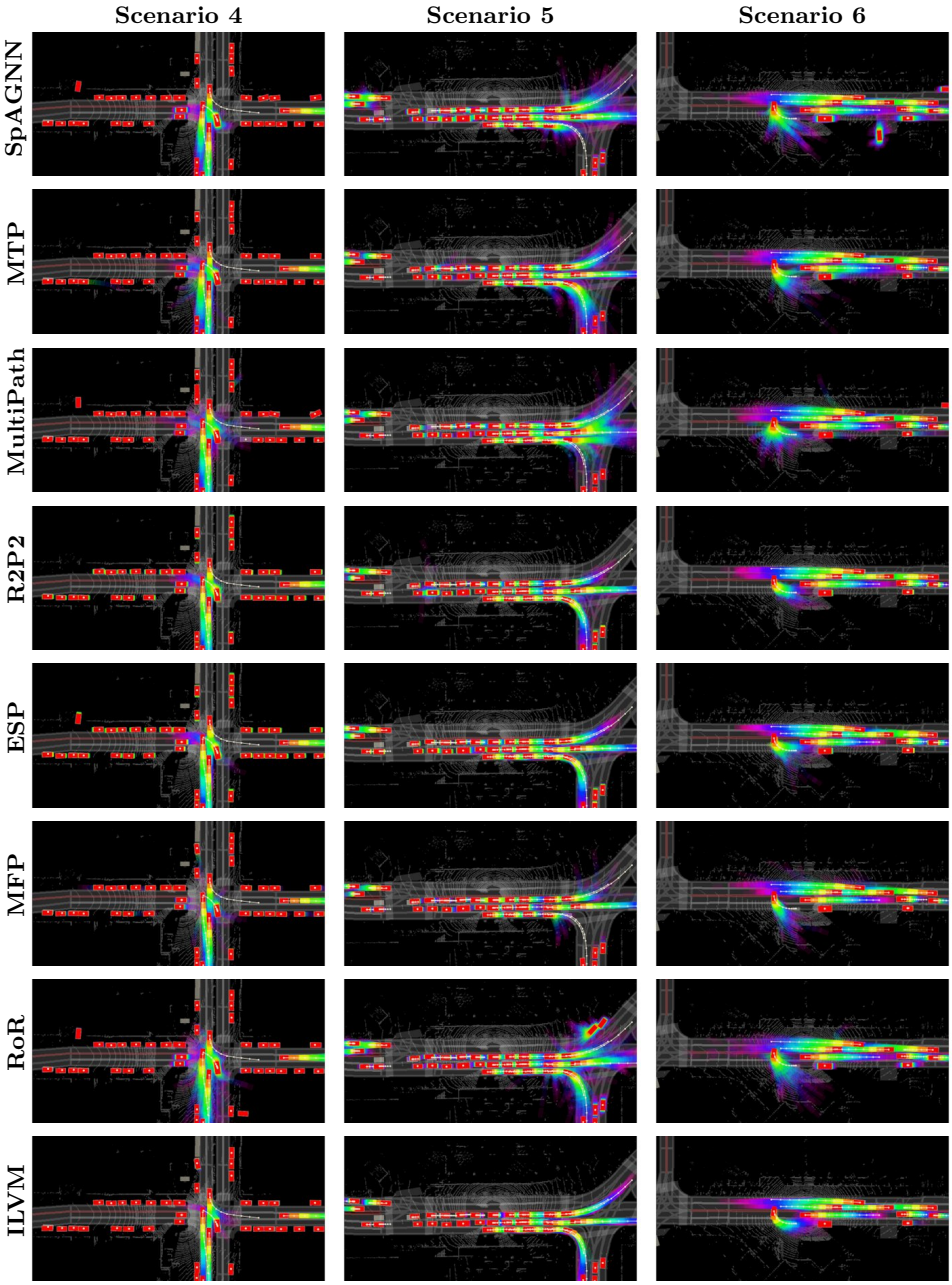


Fig. 21: **Overall Sample Quality:** *Scenario 4* showcases a complex interaction between 3 vehicles at a 4-way intersection, where our model identifies sharp "modes". However, like all other baselines, it misses (or predicts with very low probability) the true mode of the vehicle facing south and left-turning. *Scenario 5* showcases fast moving traffic, where our model can predict an accurate distribution with very low entropy even at 5 seconds into the future. *Scenario 6* Our model captures a vehicle performing a U-turn.

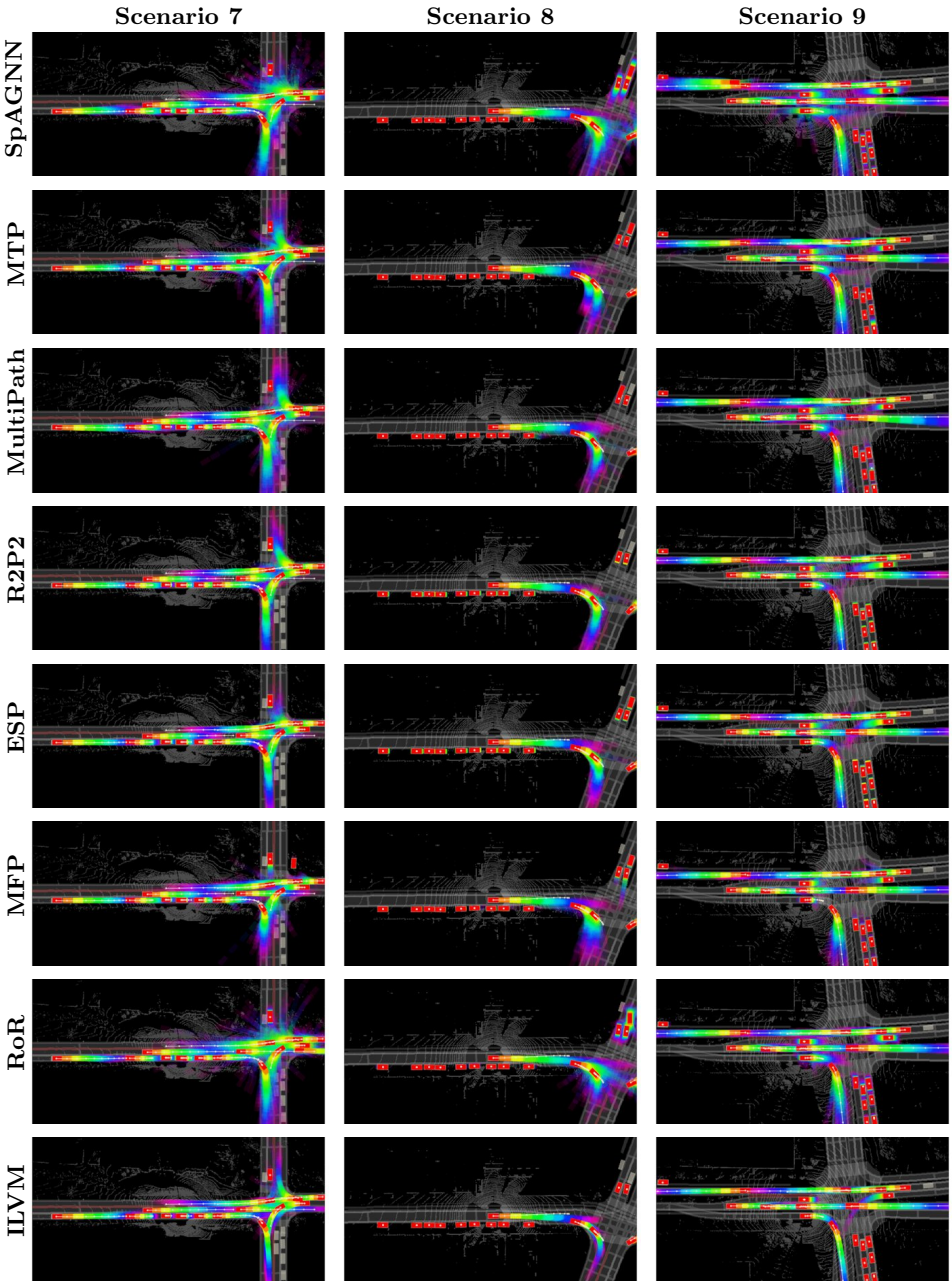


Fig. 22: **Overall Sample Quality:** We highlighted the accuracy and sharpness of our predictions in *Scenarios 7, 8 and 9*.

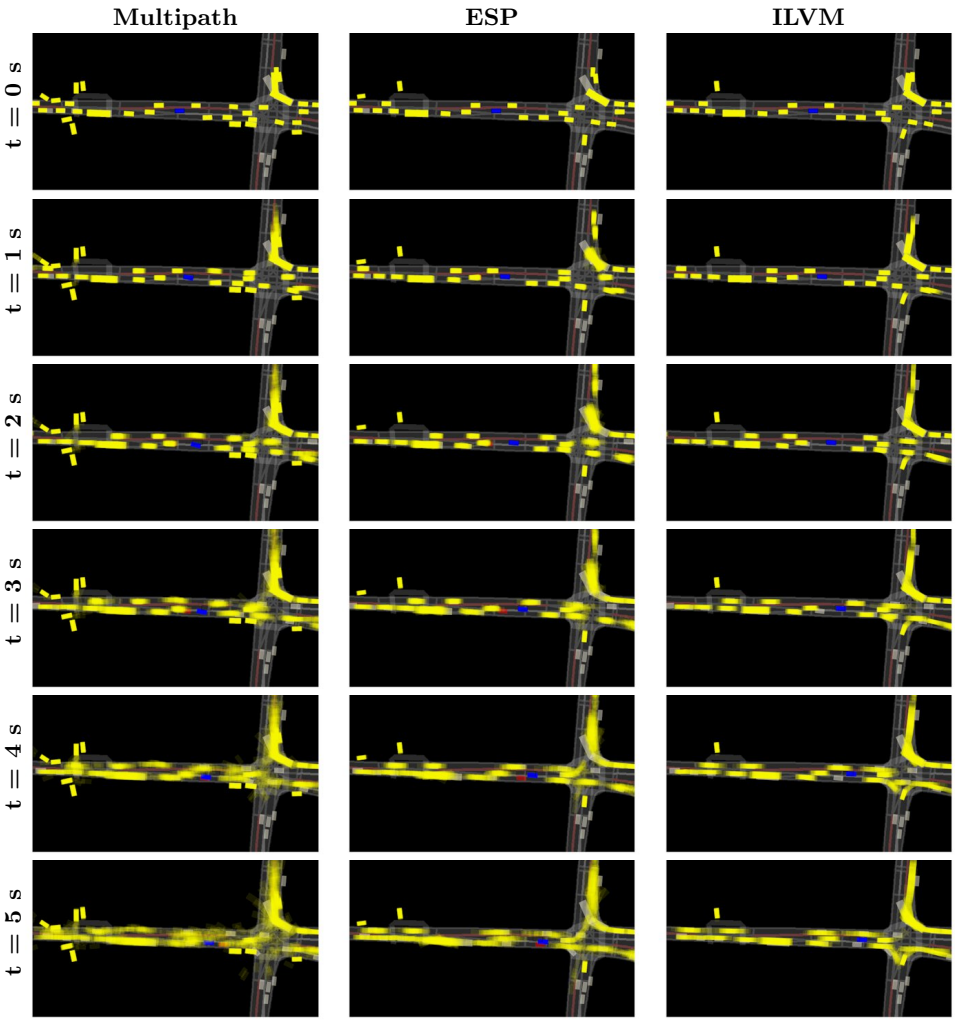


Fig. 23: In this scenario, both MultiPath and ESP generate motion forecasts that get into the SDV lane, forcing it to lane change to its right, where it collides with an actual vehicle that is lane changing from behind the SDV and is not well captured by the prediction models, including ours.

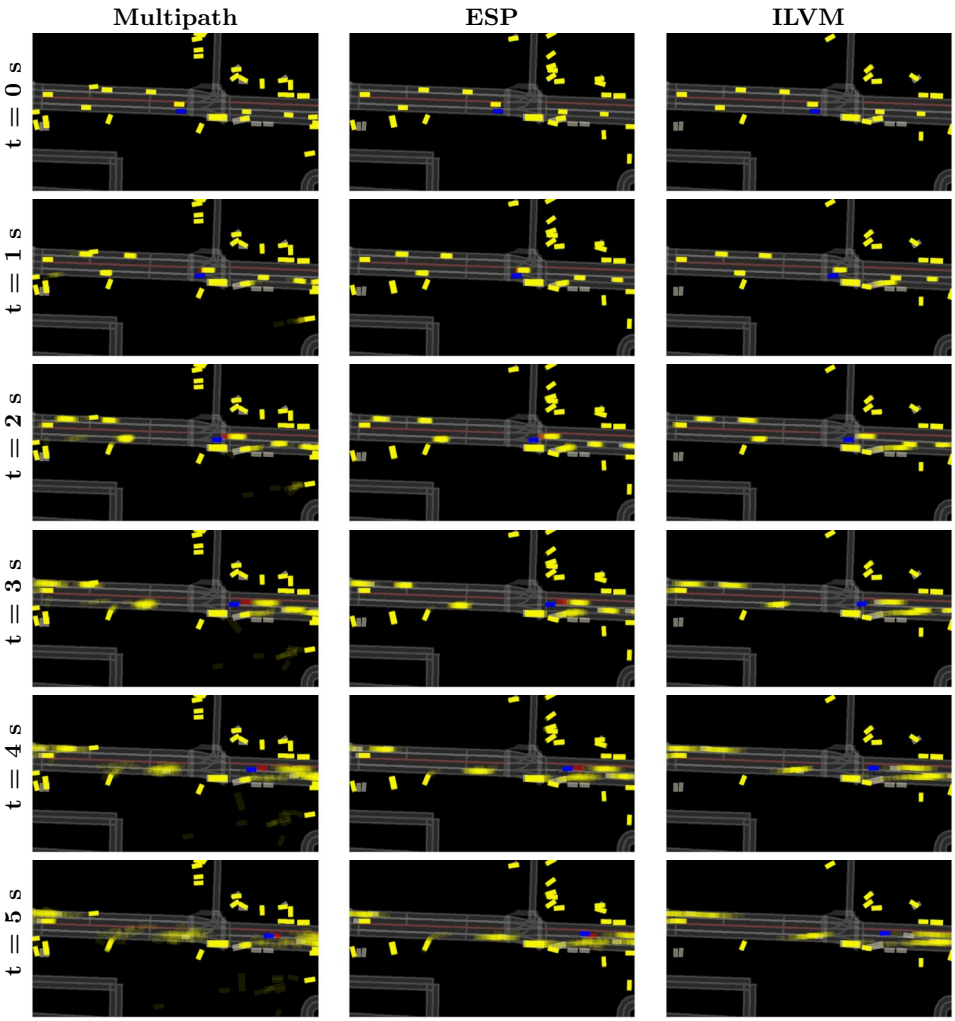


Fig. 24: In this scenario, the 3 models generate pulling out trajectories for a big vehicle, forcing the SDV to maneuver to an unoccupied region. However, ILVM captures well the distribution of the rest of the actors and the SDV performs a safe left lane change. However, in ESP and MultiPath the trajectory of the vehicle to the left is not well captured and the SDV proceeds too aggressively, resulting in a collision.

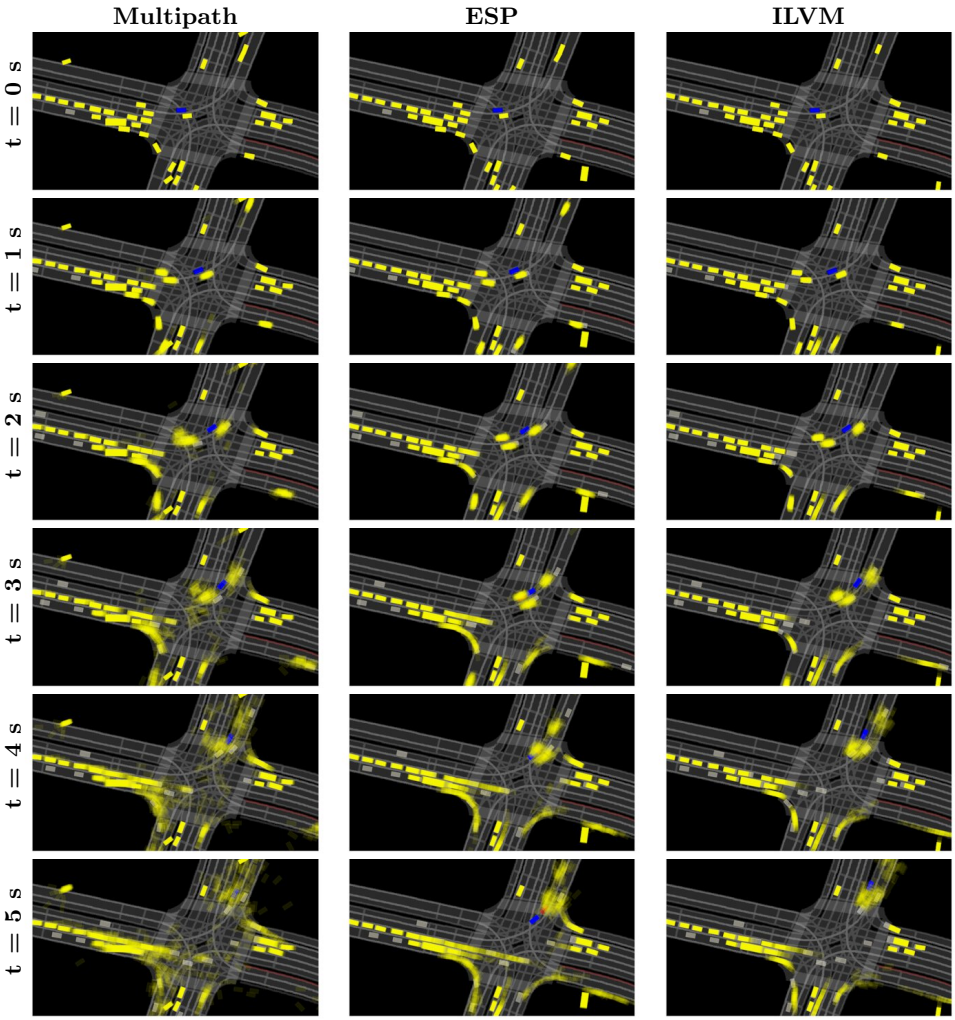


Fig. 25: ESP predicts that the vehicle that starts at the right of the SDV is going to cut-off the SDV by lane changing left, causing the SDV to hard break and causing a collision with the vehicle behind. MultiPath and ILVM successfully drive through the scenario, even though we can see how MultiPath's prediction go even into opposite traffic, but luckily do not interfere the SDV.