Supplementary Material – 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

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 [3, 5, 6], 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) [12], 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 [15] 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

* Denotes equal contribution

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) [13] specifies a directed graphical model with latent variables z and output variables y. Conditional variational auto-encoder (CVAE) [19] 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 [9] 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 [23] and nuScenes [2]. 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 [3, 4, 14] 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 [23], 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 [22], 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 [14]. Following [4], 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 [4].

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 [23] 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 [7]. 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 [21] 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 [23], 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 [10] 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 [3], 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 1. 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$_\phi$, 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$_\theta$ 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 1 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 [11] 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 [8] 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 [3] 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 [3] was originally proposed in the joint perception and prediction setting and therefore does not require any adaptation. We adapt MTP [6] and MultiPath [5] 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:

$$y^s_{n,t} = \mu_{n,t} + A_{n,t} \cdot \varepsilon^s_n$$

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 [16,17,20] 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 [18] 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 [1], but adding white noise worked better.

For ESP [17], we extended our *R2P2-MA* implementation with the "whisker" indexing technique to get added context into the feature map at the location

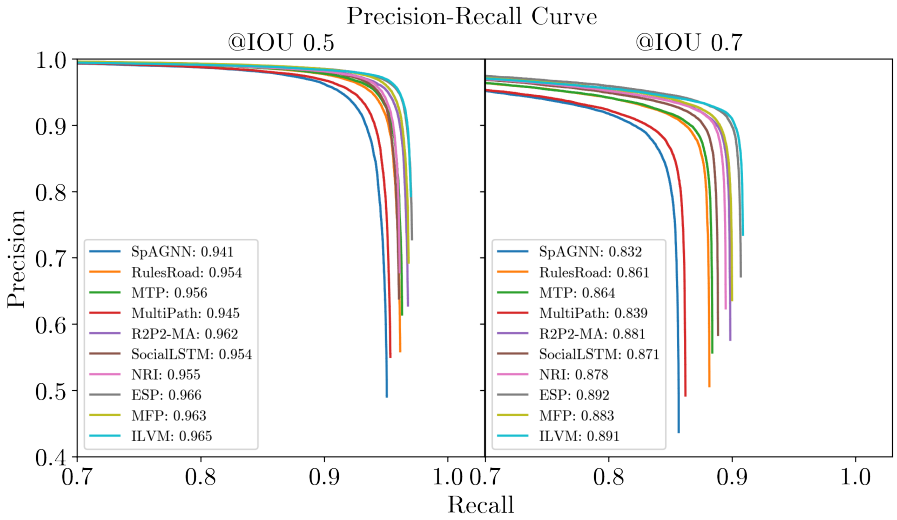

Fig. 1: Precision-Recall curve at IoU 0.5 and 0.7. Legend shows mAP (mean Average Precision) for each model. Note: horizontal axis starts at 0.7 recall.

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*.

D.1 Detection

Fig. 1 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).

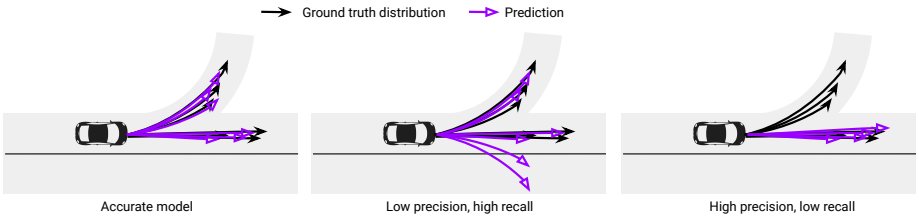

Fig. 2: Diversity vs. Precision for an individual actor

Diversity vs. Precision in multimodal prediction Fig. 2 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.

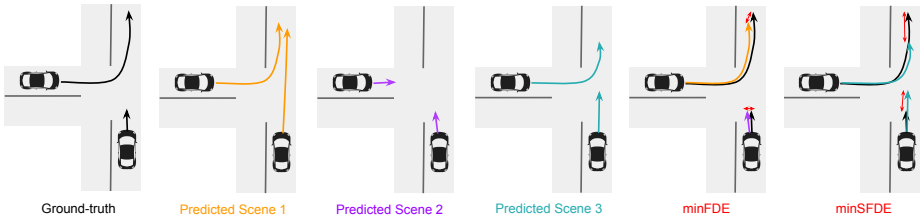

Fig. 3: Coverage metrics: actor-level (minFDE) vs. scene-level (minSFDE). The first column shows the groun-truth trajectories, the next 3 are possible futures predicted by the model, the last 2 show the trajectory samples selected by each metric (which are different), as well as their error (red arrows).

Actor-level vs. Scene-level metrics Fig. 3 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 4 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 [3] does is not enough to create scene consistent samples. Models that do joint sampling such as ILVM and ESP [17] 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.

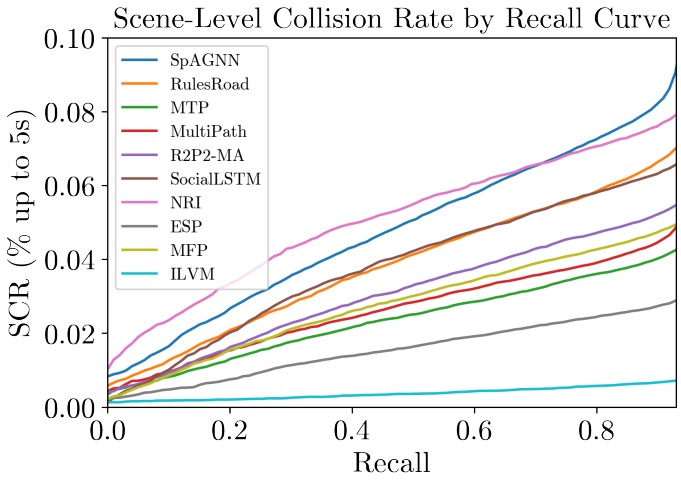

Fig. 4: ILVM models social interaction consistently well at different recall levels.

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. 5 shows that Our ILVM significantly outperforms all baselines in cumulative hit rate across across all time steps.

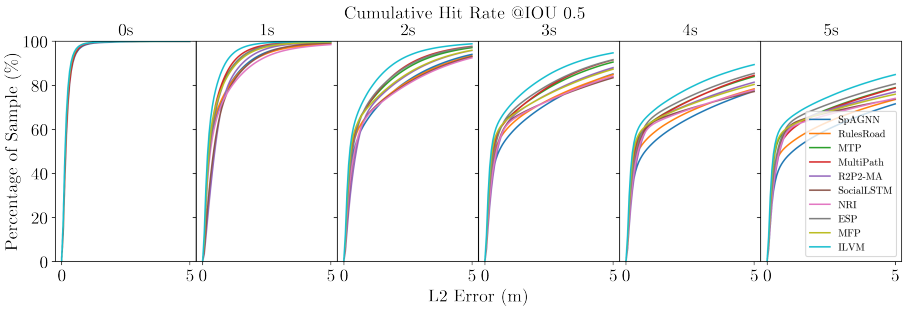

Fig. 5: ILVM obtains the best hit rate at all time-steps in the prediction horizon.

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. 6 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.

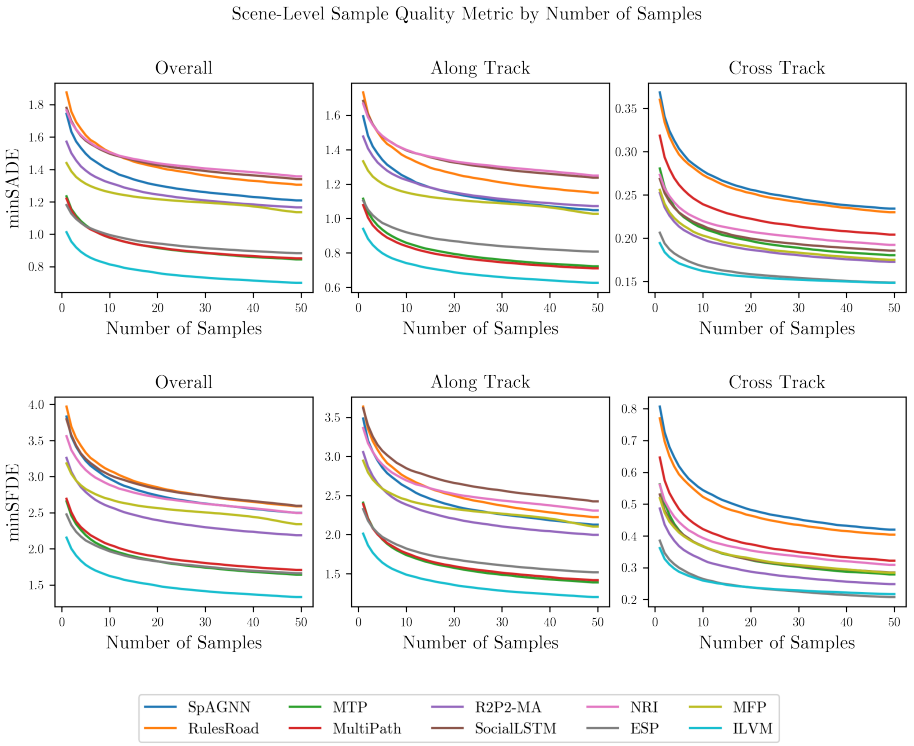

Fig. 6: Our ILVM outperforms more significantly in along track error than cross track error for both scene-level minADE and minFDE metrics.

Sample Quality – Precision Diversity Tradeoff In Fig. 7, 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.

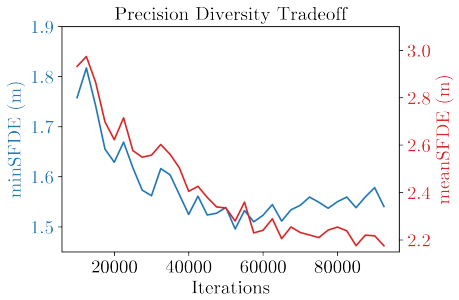

Fig. 7: Tradeoff between minSFDE and meanSFDE as model training progresses. Note that the *minSFDE curve* follows the y-axis on the left, and the *meanSFDE curve* follows the y-axis on the right.

Ablation – KL Term We include an analysis of how the *beta* weight on the KL term trades off diversity and precision in Fig. 1.

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 1: **[ATG4D] KL loss ablation study**

We observe that:

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 it's not trained on.

E Additional Visualizations

Scene Consistency: In Figure 8, 9, 10, 11 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. 12 and 13, 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. 14, 15, 16, 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. 18, 17, 19, 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.

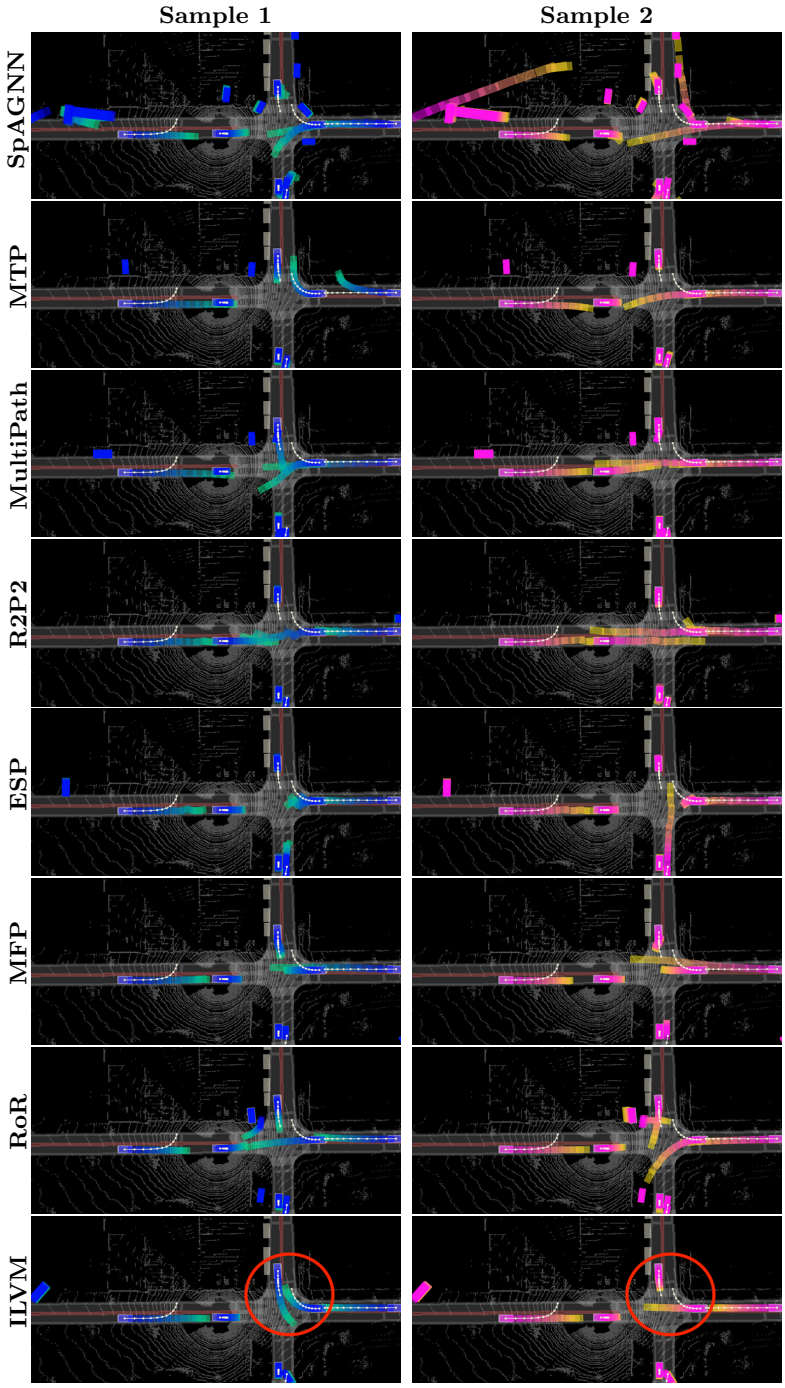

Fig. 8: **Scene-level samples**. Our latent variable model captures complex interactions at intersections. In this example, the car facing south will yield/go if the car facing west goes straight/turns right, respectively. The baselines do not capture this complex interaction, and most show inconsistent (colliding) samples for the 2 highlighted actors.

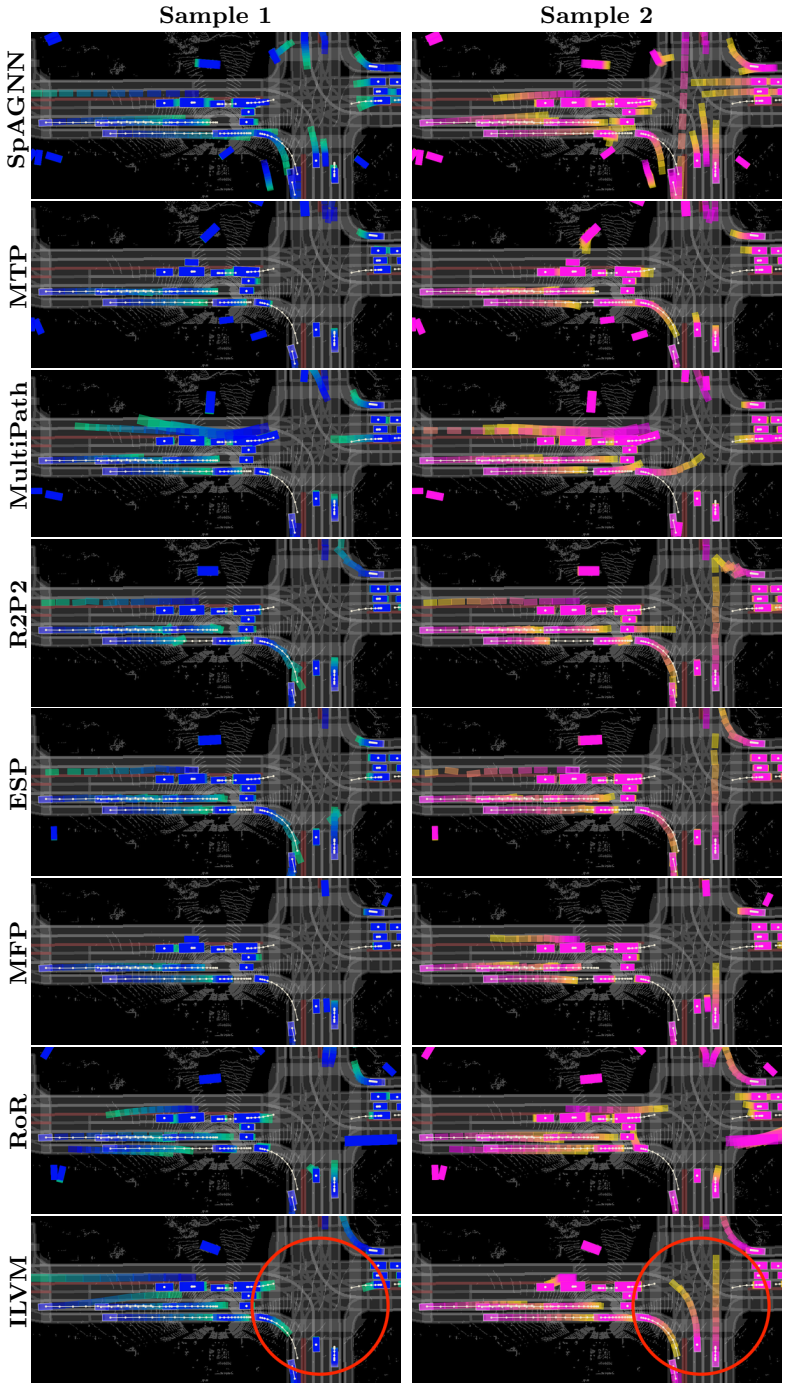

Fig. 9: **Scene-level samples**. Our latent variable model captures the different scene outcomes for possible states of a given traffic light intersection (vertical vs. horizontal traffic).

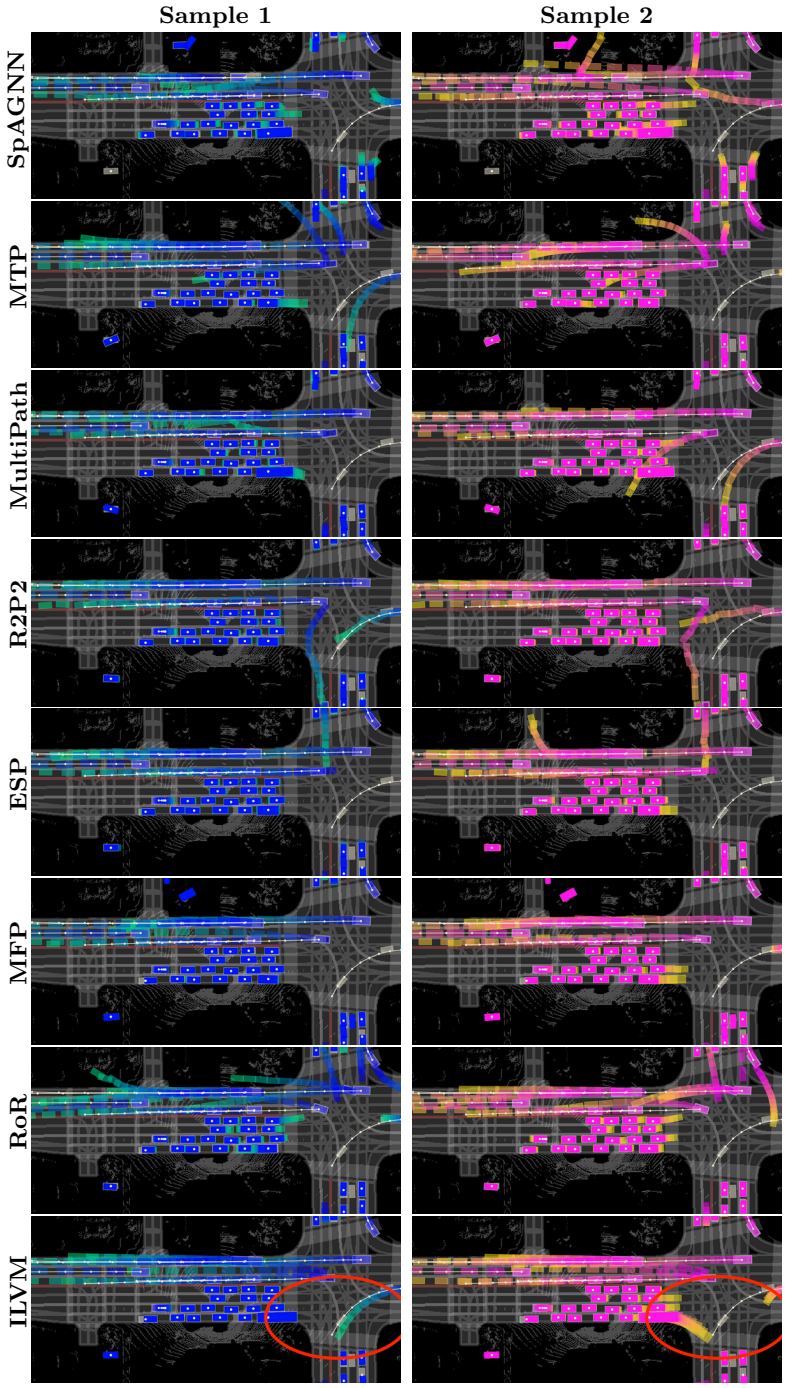

Fig. 10: **Scene-level samples**. Our latent variable model captures whether the bus will proceed with the right turn, or the left-turning vehicle will.

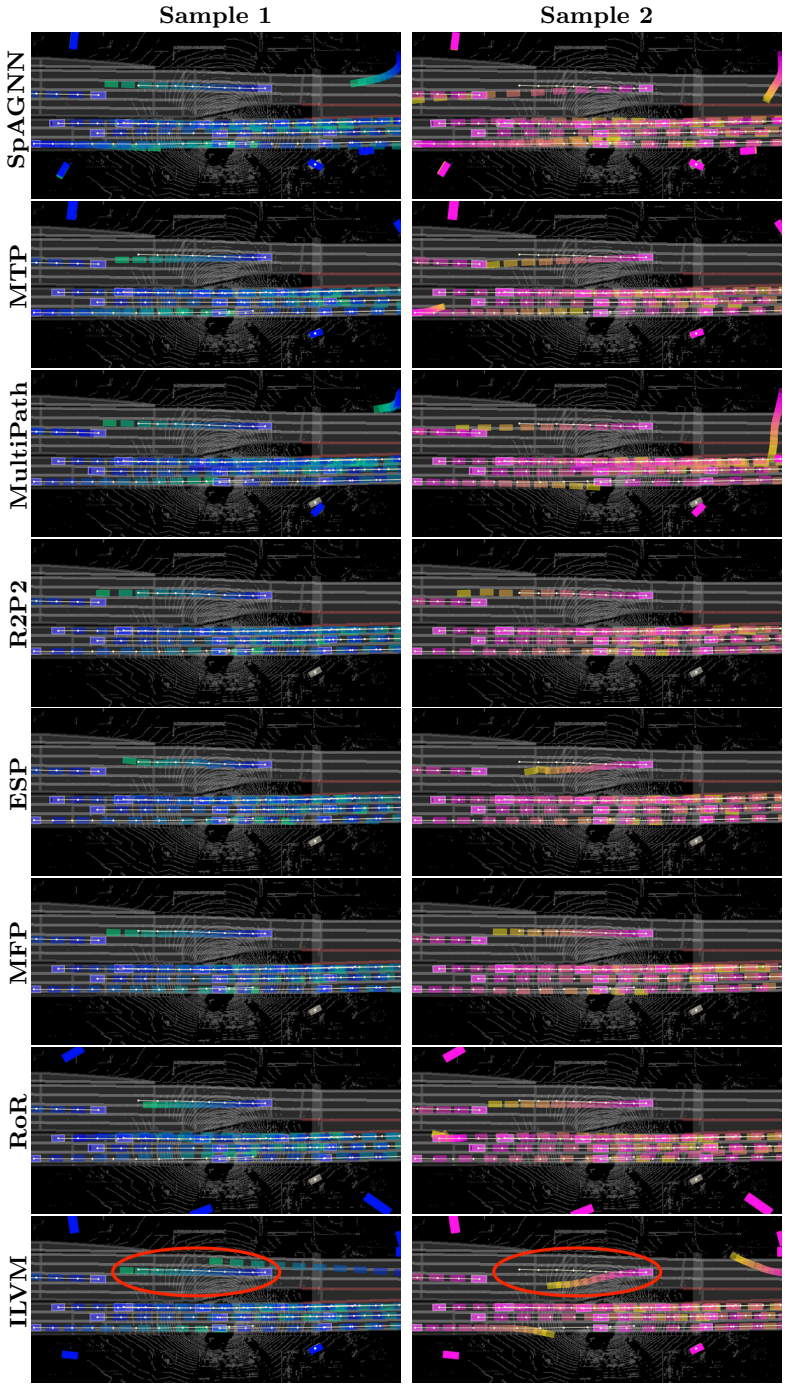

Fig. 11: **Scene-level samples**. Our latent variable model captures multiple realistic futures (including lane changes) that respect the map geometries and are dynamically feasible.

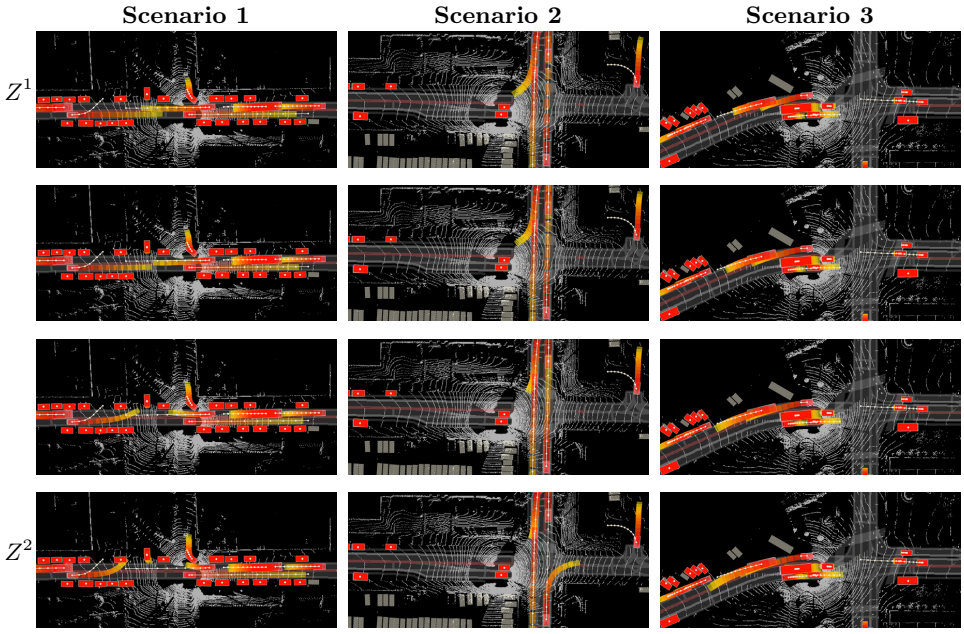

Fig. 12: **Latent space interpolation:** *Scenario 1* showcases a complex interaction between 3 vehicles: when the 2 vehicles in the road predict turning or slow moving trajectories, the third vehicle pulls out of the driveway, and when the 2 vehicles in the road keep constant velocity to go straight, 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.

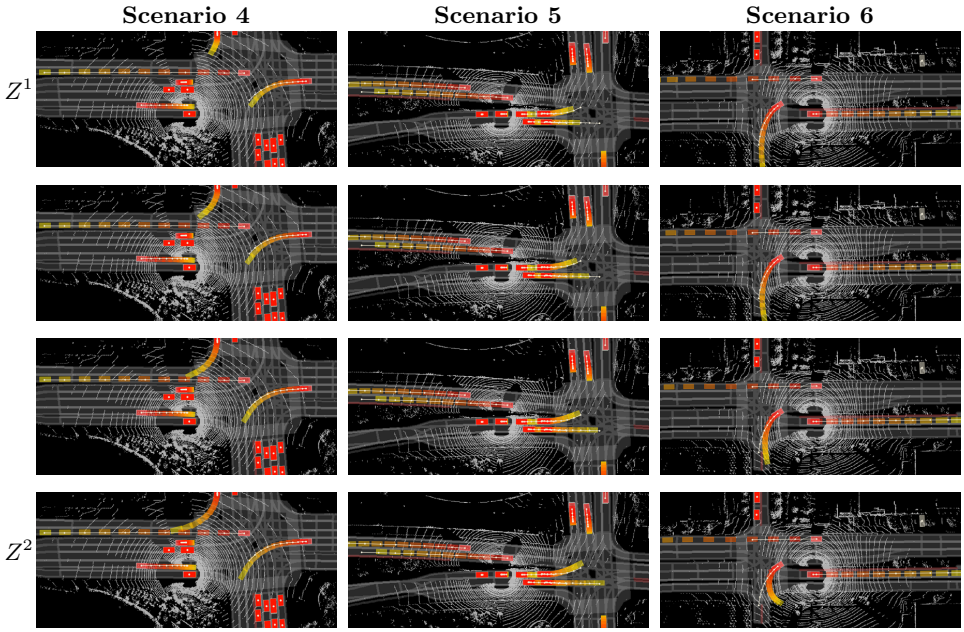

Fig. 13: **Latent space interpolation:** *Scenario 4* and *Scenario 5* showcase smooth transitions between different speed profiles when turning and going straight at an intersection. *Scenario 6* we can see all the range of possibilities from a left-turn to a u-turn, which is a pretty rare event.

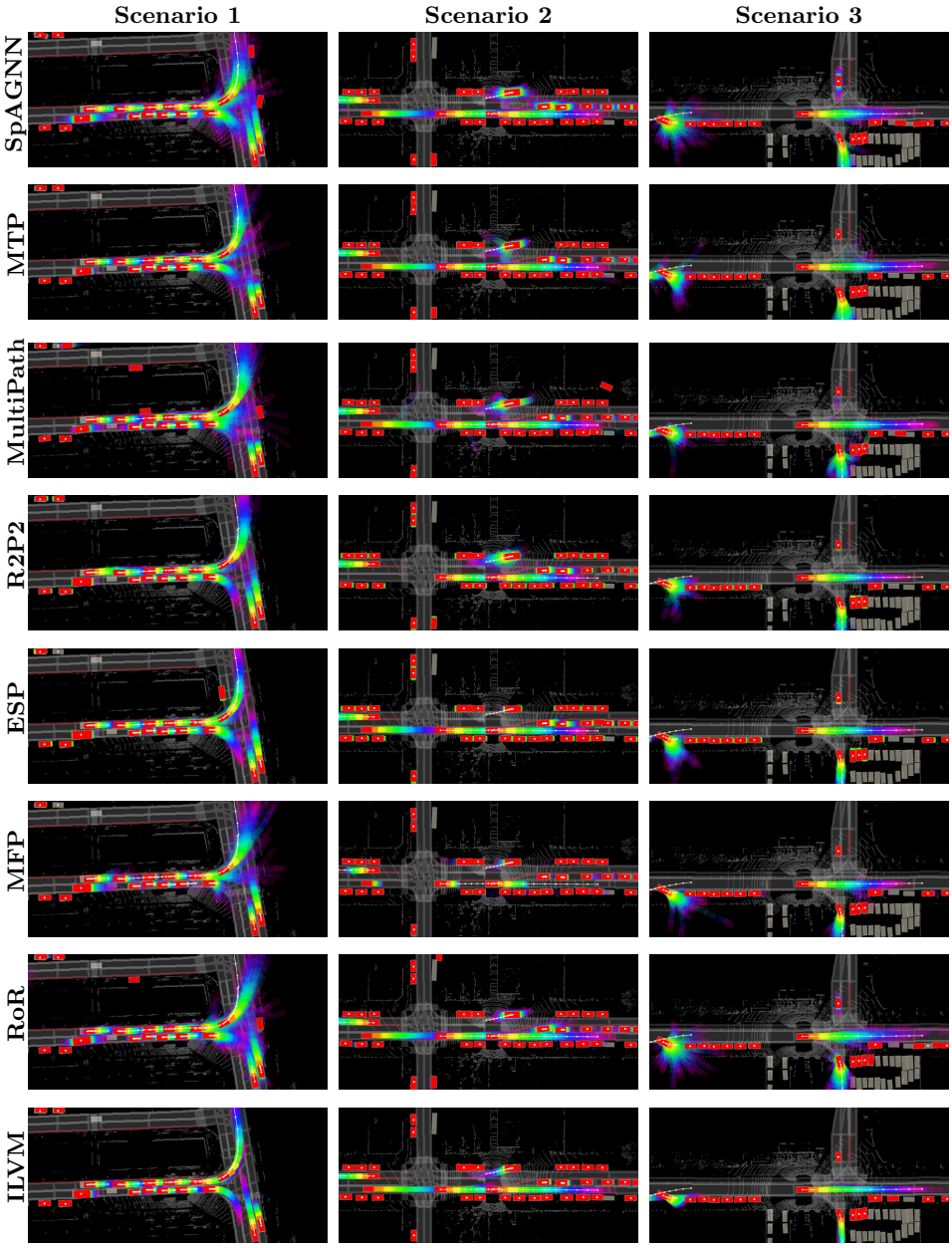

Fig. 14: **Overall Sample Quality:** *Scenario 1* showcases a T-intersection with fast-moving turns. *Scenario 2* is interesting because there is a vehicle coming out of a parking spot, which is not very frequent in driving logs. *Scenario 3* captures a vehicle maneuvering into a parking spot, also an unusual event.

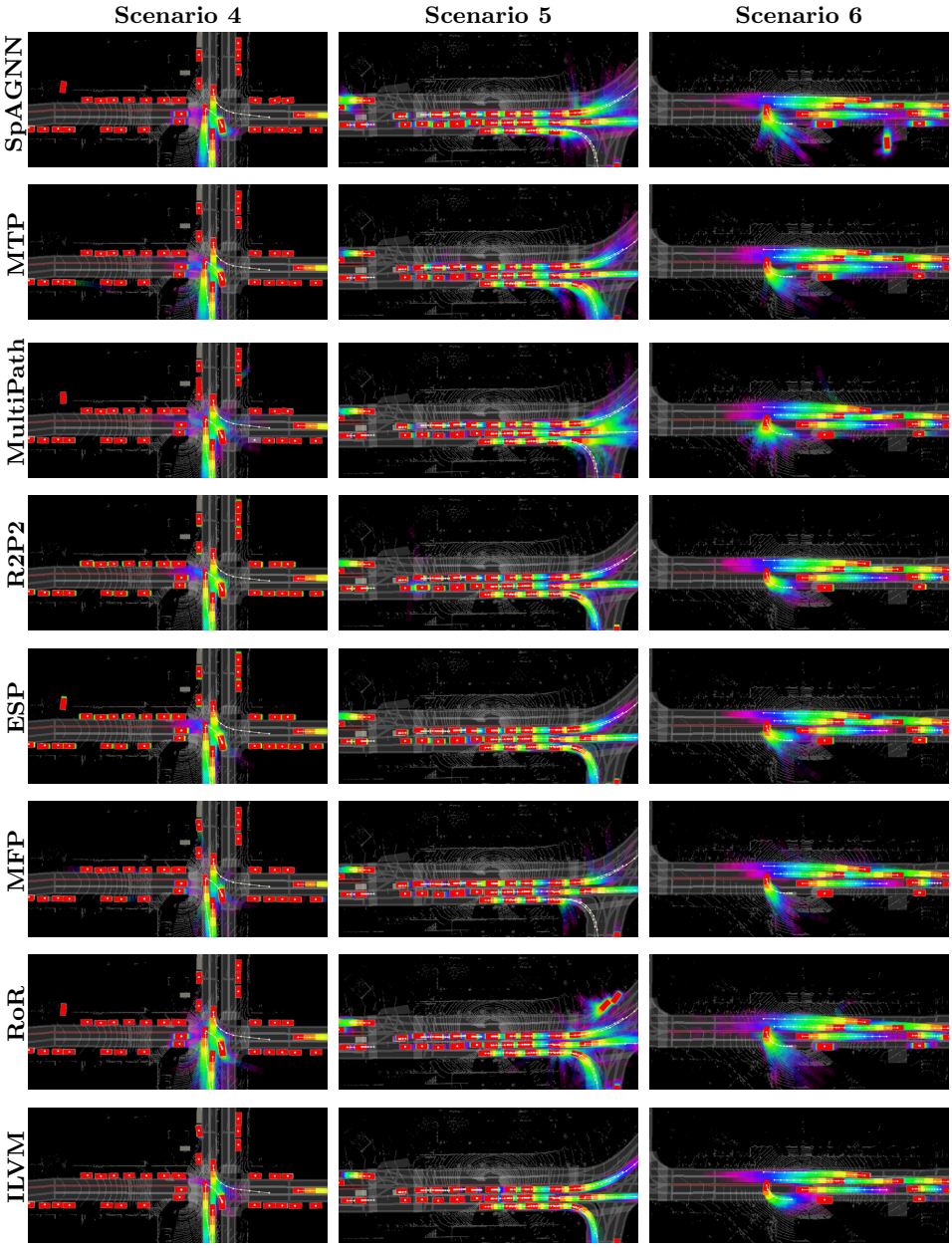

Fig. 15: **Overall Sample Quality:** *Scenario 4* showcases a complex interaction between 3 vehicles at a 4-way intersection, where our model identifies sharp "modes". However, like all other baselines, it misses (or predicts with very low probability) the true mode of the vehicle facing south and left-turning. *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.

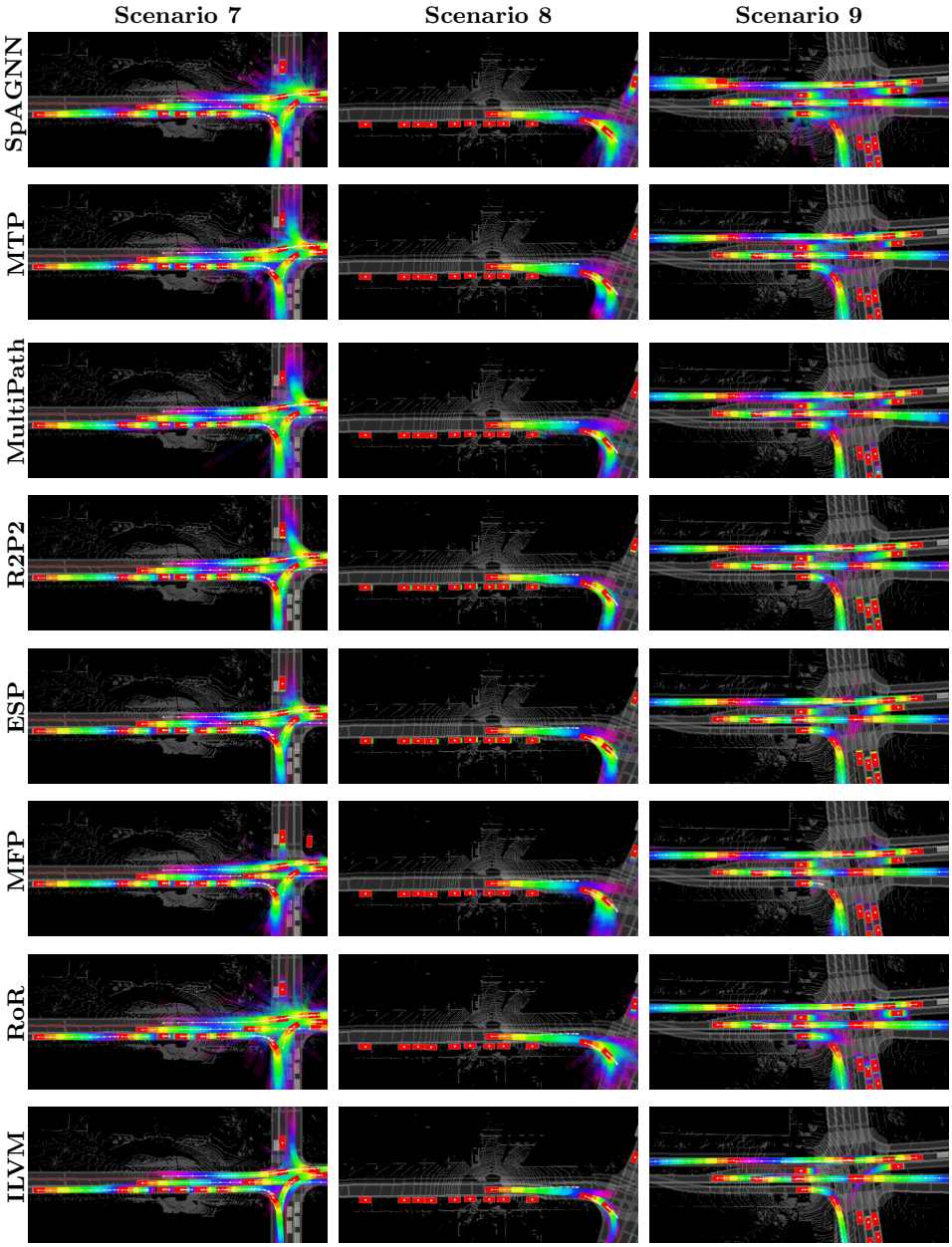

Fig. 16: **Overall Sample Quality:** We highlighted the accuracy and sharpness of our predictions in *Scenarios 7, 8 and 9*.

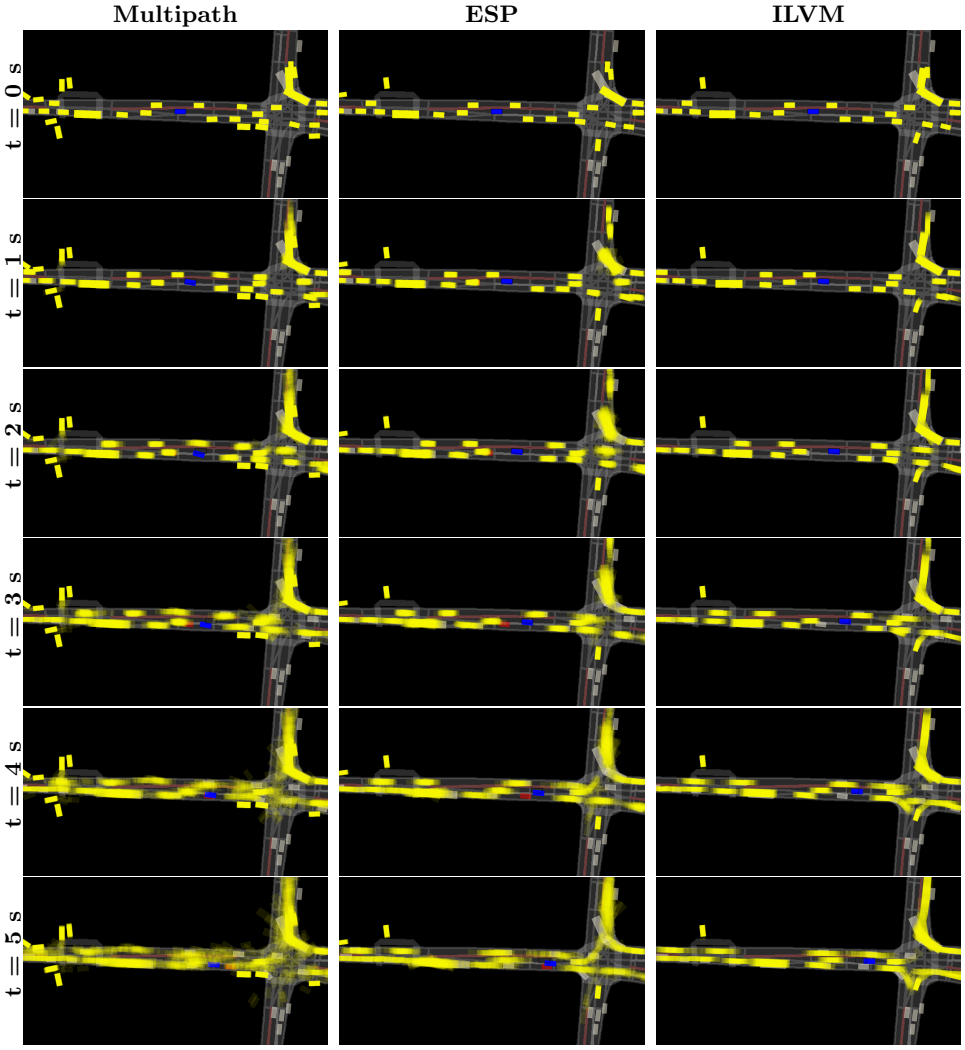

Fig. 17: In this scenario, both MultiPath and ESP generate motion forecasts that get into the SDV lane, forcing it to lane change to its right, where it collides with an actual vehicle that is lane changing from behind the SDV and is not well captured by the prediction models, including ours.

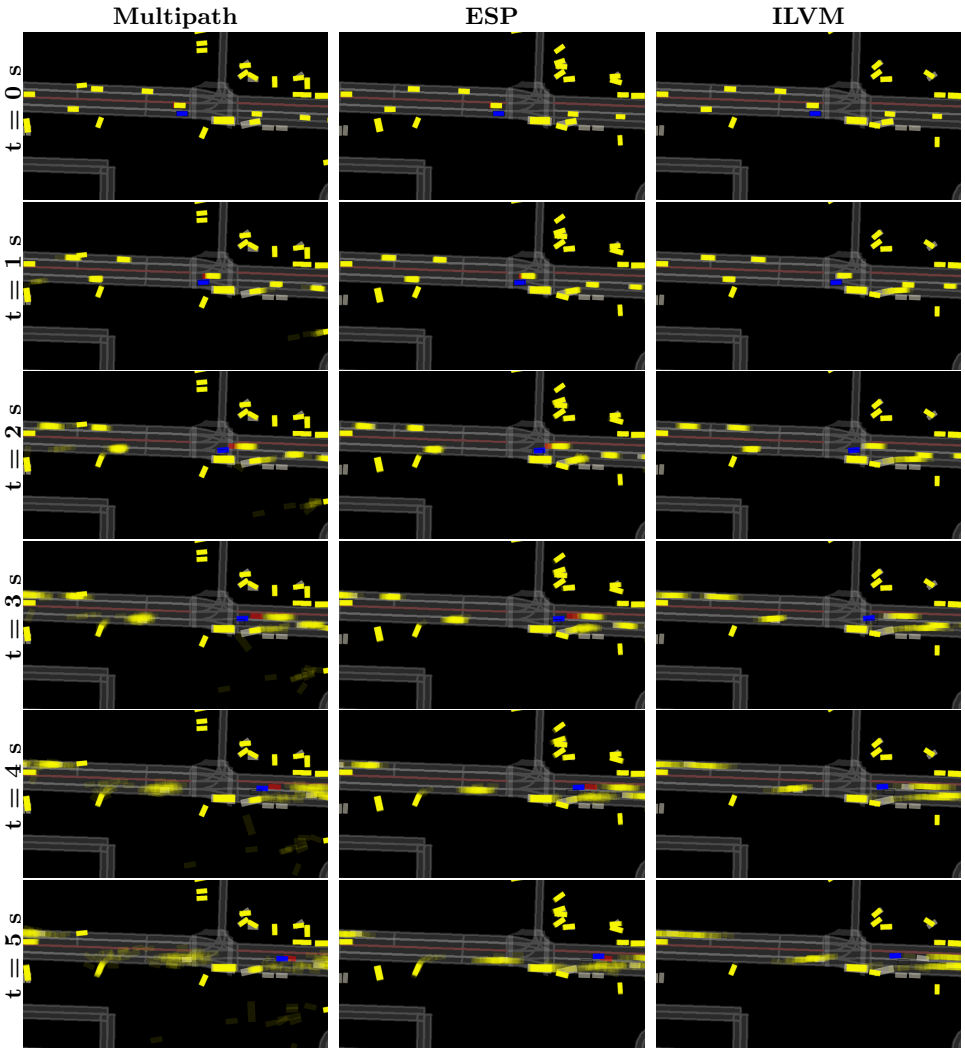

Fig. 18: In this scenario, the 3 models generate pulling out trajectories for a big vehicle, forcing the SDV to maneuver to an unoccupied region. However, ILVM captures well the distribution of the rest of the actors and the SDV performs a safe left lane change. However, in ESP and MultiPath the trajectory of the vehicle to the left is not well captured and the SDV proceeds too aggressively, resulting in a collision.

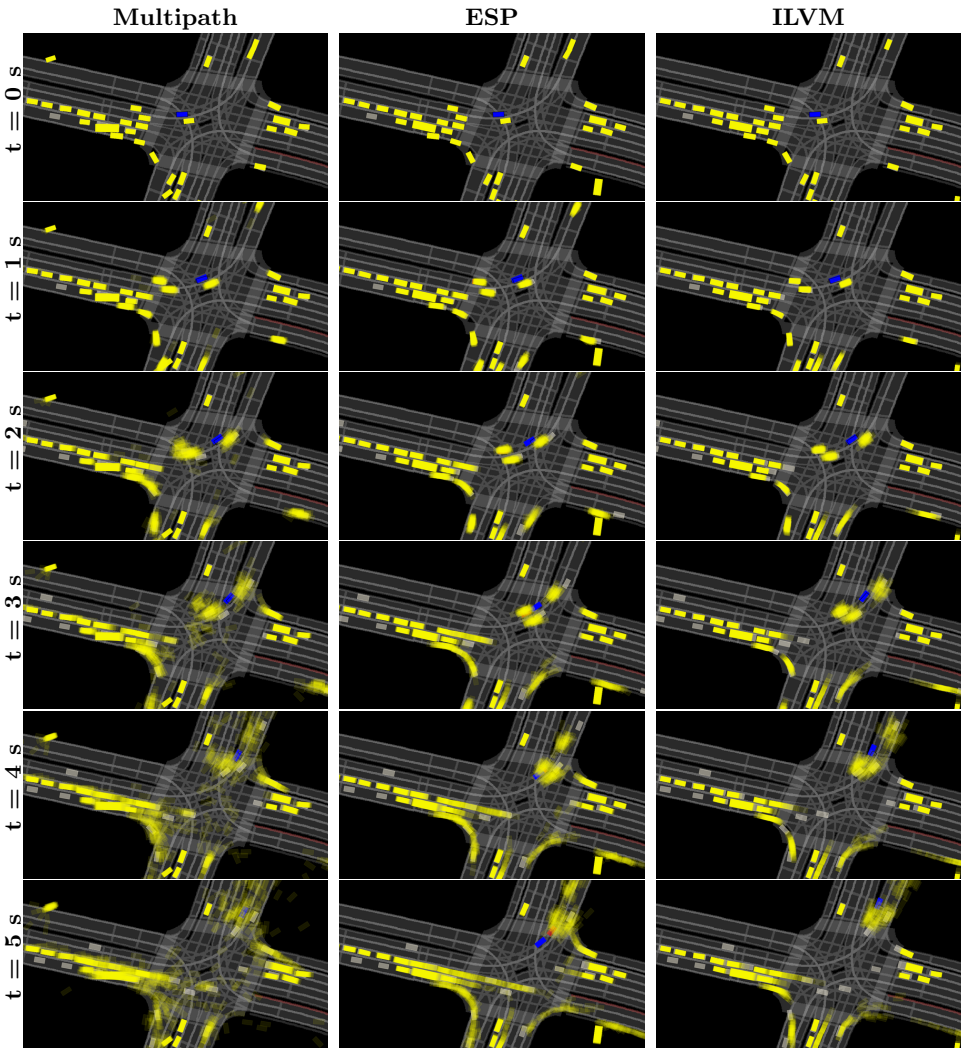

Fig. 19: ESP predicts that the vehicle that starts at the right of the SDV is going to cut-off the SDV by lane changing left, causing the SDV to hard break and causing a collision with the vehicle behind. MultiPath and ILVM successfully drive through the scenario, even though we can see how MultiPath's prediction go even into opposite traffic, but luckily do not interfere the SDV.

References

1. 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)
2. 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)
3. 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)
4. Casas, S., Luo, W., Urtasun, R.: Intentnet: Learning to predict intention from raw sensor data. In: Conference on Robot Learning (2018)
5. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449 (2019)
6. 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)
7. 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)
8. 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
9. 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.
10. Huang, J., Sivakumar, V., Mnatsakanyan, M., Pang, G.: Improving rotated text detection with rotation region proposal networks (2018)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
13. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2013)
14. 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)
15. Mohamed, S., Lakshminarayanan, B.: Learning in implicit generative models. arXiv preprint arXiv:1610.03483 (2016)
16. 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)
17. 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)
18. 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)

19. 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)
20. Tang, C., Salakhutdinov, R.R.: Multiple futures prediction. In: Advances in Neural Information Processing Systems. pp. 15398–15408 (2019)
21. Wu, Y., He, K.: Group normalization. In: Proceedings of the ECCV (ECCV) (2018)
22. Yang, B., Liang, M., Urtasun, R.: Hdnet: Exploiting hd maps for 3d object detection. In: Conference on Robot Learning. pp. 146–155 (2018)
23. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: Proceedings of the IEEE CVPR (2018)