Supplementary Materials: LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving

Alexander Cui [*,1,2], Sergio Casas [*,1,2], Abbas Sadat [*,1], Renjie Liao [3], Raquel Urtasun[1,2]
[†] Waabi[1], University of Toronto[2], Google Brain[3]
{acui, sergio, asadat, urtasun}@waabi.ai, rjliao@google.com

In this supplementary materials, we first describe additional implementation details, then we discuss additional evaluation details and results, and finally showcase additional qualitative results. The supplementary video contains a narrated overview of the method and longer duration rollouts of our model driving in closed-loop simulation.

1. Implementation Details

In this section, we cover implementation details about the submodules of our end-to-end driving model as well as training.

1.1. Joint Perception and Motion Forecasting Details

Here, we discuss the implementation details for our scene-consistent join perception and motion forecasting model from sensor data.

Data Input Parameterization: The preprocessing of our LiDAR point cloud input to the model follows in the same manner as [4]. Primarily, we use a Bird's Eye View (BEV) of a voxelized 3D LiDAR point cloud, with the height and time dimensions being raveled into the channel dimension. Our model does not use tracks as input, so motion information is accounted for by including past LiDAR sweeps into the input. We project past LiDAR sweeps into the coordinate frame of the SDV's current LiDAR sweep, and concatenate them in the channel dimension. Additionally, our high-definition maps are represented as a stack of rasterized images, as described in [4]. These maps encode elements of the road such as intersections, lanes and roads, with different elements encoded in different channels.

Shared Perception Backbone: To extract features for object detection and motion forecasting, we use a backbone network as described in [3], which was adapted from [14]. We separately process the LiDAR and HD maps in two separate sets of convolutional layers, concatenate those intermediate features channel-wise (as they use the same spatial resolution and coordinate system), and fuse them with a convolutional header to get a scene-level feature map. In particular, the LiDAR backbone is composed of 4 residual convolutional blocks with 2, 2, 3, and 6 layers respectively. These blocks use 32, 64, 128, and 256 filters and a stride of 1, 2, 2, and 2 respectively. The HD map backbone is also composed of 4 residual blocks with 2, 2, 3, and 3 layers respectively. The HD map backbone uses 16, 32, 64, and 128 filters and a stride of 1, 2, 2, and 2 respectively. For both backbones, the output of each residual block is concatenated to create a final multi-resolution feature map, as detailed in [14]. These features maps are down-sampled 4x relative to the input. Finally, we use a header network with 4 convolutional layers and 256 filters per layer to fuse the concatenated features. GroupNorm [13] is used because of our small batch size (number of frames) per GPU, which we adopt due to GPU memory constraints. The final feature map is used for the detection and motion forecasting networks. Note that the backbone network and object detector architecture is shared across all models including the baselines to make the comparison more fair and direct.

Object Detection Header: To detect the actors in the scene, we input the feature map from the backbone into one convolution layer to predict a confidence score and another convolution layer to predict a bounding box for each anchor location, following the parameterization described in [14]. Next, non-maximal suppression (NMS) with an IoU of 0.1 is used to filter

[*]Denotes equal contribution
[†]This work was done by all authors while at Uber ATG

overlapping detections and low probability detections are filtered by a threshold corresponding to the maximum F1 score across the Precision-Recall curve, to arrive at a final set of actor bounding boxes.

Actor Feature Extraction: Finally, in order to obtain local actor contexts x_n^{local}, we use rotated ROI Align [7] and extract a crop of the feature map of a fixed size around each detected actor, which is rotated to align with the actor's centroid and orientation. The cropped region spans 10m to the back, 70m to the front, and 40m to each side of the actor, and has dimension 40 x 40 x 256. We apply an *actor CNN* (a 4-layer convolutional network with heavy downsampling) to each feature map to get a 512-dimensional feature vector x_n^{local} for every actor. In order to incorporate global information about the actor's pose in the context of the whole scene, we add the BEV centroid and rotation relative to the SDV $x_n^{global} = \{c_{x,n}, c_{y,n}, a_n\}$ as features, and concatenate these two feature vectors channel wise to get the final actor context $x_n = [x_n^{local}, x_n^{global}] \in \mathbb{R}^D$

Scene Interaction Module: For the basis of our motion forecasting model, we use a "scene interaction module" (SIM) graph neural network as described in [3] and inspired by [2]. SIM is used in the Prior, Encoder, Decoder, Diverse Sampling networks as well as the Scenario scorer. Given a graph of actor nodes with embeddings, the SIM will pass in the hidden states of the two actor nodes in a given edge, in addition to the projected distance between their two bounding boxes, to a 3-layer MLP. This computes an activation for each edge in the graph, that goes through feature-wise max-pooling and a GRU cell to compute a new hidden state for each node. Finally, a 2-layer MLP is applied to each node to get their final outputs. All our SIMs use a hidden state size of 64. We run two SIMs in parallel for each of our Prior and Encoder networks to compute a latent μ and σ vector of length 64 for each actor. These are sampled in a gaussian parameterization to get z_n vectors of length 64 for each agent. These z_n vectors and the x_n feature vector is then concatenated for each actor for a total length of 576. This set of actor vectors is then fed into a decoder SIM that computes a length 20 output ((x, y) waypoints over 10 time steps).

1.2. Planning-Centric Diverse Sampler Details

Here, we will describe the implementation details of our diverse sampler network.

Network Architecture: To train the diverse sampler network $\mathcal{M}_\eta$, we first train the scene-consistent joint perception and motion forecasting model described above, and freeze the detection backbone, actor feature extractors and decoder network. We replace the Encoder and Prior networks with our diverse sampler. This model consists of two SIMs, one for the A vector, and another for the B vector. These SIMs have identical architecture - they take as input the graph of actor feature vectors of length 512, and use hidden states of length 64. Each model jointly outputs $K = 15$ samples of 64-dimensional vectors for each agent, and does this by outputting a $K * 64$ length vector for each agent. As described in section 3.2, these vectors parameterize the latent mapping that is used to sample $\mathbf{Z}$, a set of K latent vectors. These are then decoded into K scene predictions, which is done in parallel by batching the K latent samples as input to the decoder.

As mentioned in the description of the DLow baseline, we stray from the implementation described in [16] by predicting only the diagonals of the A matrix instead of a full matrix, because the full matrix would not fit in memory given our high dimensionality. We have higher dimensionality because our scene latent vectors represent a sample of the joint distribution of all actors, as opposed to the marginal distribution of a single actor.

Hyperparameters: For learning, we weight the energies in our model as follows: E_r has a coefficient of 0.02, E_p has a coefficient of 0.01, E_d has a coefficient of 10, and σ_d is equal to 10000.

1.3. Scenario Probability Estimation Details

The scenario probability estimation network is parameterized as another SIM with a hidden dimension of 128. This SIM takes as input the K predicted scenarios $\mathbf{Y}$. To do so, the n-th node state in the graph is initialized to the K trajectories of actor n, $\mathbf{Y}_n \in \mathbb{R}^{2TK}$. After 1 round of message passing in the SIM, we take all the updated node states and average pool them over the node (or actor) dimension, thus obtaining a single feature vector of the hidden dimension (128). Then, a MLP maps these features into the K scores, one for each future.

1.4. Contingency Planner Details

In this section we provide details of the action and trajectory sampling, followed by the description of the planner cost functions.

1.4.1 Action and trajectory sampling

Since the motion-paths representing the lane centers are strong priors for potential SDV paths, we perform the action and trajectory sampling in Frenet Frame of the goal motion-path, given by the input route. The action and its corresponding set of long-term trajectories are represented by lateral and longitudinal trajectories relative to the goal motion-path [12]. The sampling is achieved by first generating a lateral profile. We use two quintic polynomials that are generated by the initial SDV state in Frenet frame, and sampled lateral offsets for mid/end-conditions as in [10]. Next, to generate the actions, we sample longitudinal profiles in form of quartic polynomials which, combined with the generated lateral trajectory, yields a bicycle model trajectory representation. Similarly, in order to sample contingent plans, we generate long-term longitudinal trajectories in form of two quartic polynomials. These polynomials are conditioned on the end longitudinal-state of the corresponding action, and sampled mid/end-conditions [10]. Note that we use 1 second horizon for the actions and 4 seconds for the trajectories.

1.4.2 Costing

The planner cost function includes subcosts that encode different aspects of the sample actions and trajectories, including safety, traffic-rules, and comfort.

Safety: Given the predicted trajectories of the actors, the collision subcost penalizes a sample SDV trajectory if the SDV polygon is overlapping with the polygon of the other actors. This collision cost is computed separately for each class of actors. Furthermore, a trajectory is penalized if it has high velocity close to other actors. Another subcost related to safety is the headway subcost, in which the SDV trajectory is penalized if it is violating a safety distance to the leading vehicle. This safety distance is determined by the velocity of both SDV and the lead vehicle such that the SDV can stop with a comfortable acceleration profile, in case the lead vehicle suddenly stops with hard breaking.

Traffic rules: The SDV is required to stay on its lane and close to the centerline. Therefore, trajectories that are far from the lane-motion paths are penalized proportional to the offset. Similarly, if the SDV polygon goes off of the lane, the trajectory is penalized. In order to prevent the SDV from violating red-lights, trajectories that enter red-light intersections are penalized proportional to the violation distance. We use similar costing for junctions with stop-signs. Furthermore, trajectories that go above the speed-limit of the road are penalized proportional to the violation margin.

Progress and comfort: In order to promote trajectory samples that progress in the route, we use the traveled longitudinal distance as a reward (negative cost). Additionally, trajectories that violate the kinematic and dynamic constraints of the vehicle are penalized, including curvature, acceleration, deceleration, and lateral acceleration. Additionally, high jerk and acceleration and decelerations are penalized by cost functions to promote comfortable trajectories.

1.5. Optimization Details

When training the scene-consistent motion forecaster, we used the same optimization settings as in [3], where we use the Adam optimizer [8] with a learning rate of 1.25e-5, with a cyclical annealing schedule for one of our coefficients. This training ran for 50,000 iterations of batch size 4 on 16 Nvidia RTX 5000 GPUs. When training the diverse sampler, we use the same learning rate, without cyclical annealing schedule, training this model for 40,000 iterations of batch size 1 (due to memory constraints) on 8 Nvidia RTX 5000 GPUs.

2. Additional Evaluation Details

2.1. Operating point for evaluation

We evaluate motion forecasting on true positive detections. To find a fair operating point of the object detectors for all models in the motion forecasting task, we follow [2, 3] and find the detection threshold corresponding to a common recall point. In particular, we evaluate motion forecasting at 90% recall for vehicles, 60% for bicyclists and 70% for pedestrians.

For the downstream task evaluation of motion planning, we find the maximum F1 score point in the Precision-Recall curve for each baseline, and operate the detector at that point to minimize false positive and false negatives.

2.2. Formal sub-system level metrics definitions

The metrics used at the sub-system level in our open-loop evaluation are defined below. The *minimum scene average displacement error (minSADE)* measures how well we recall the ground-truth trajectory by measuring the distance between the ground-truth scene and the closest predicted scene. The *mean scene average displacement error (meanSADE)* measures the precision of the predicted distribution at the scene-level as proposed in [3] by measuring how different the predicted scenes are on average with the ground-truth. In our formulation, meanSADE takes into account the scenario probabilities in computing the weighted sum of SADEs.

$$\text{minSADE} = \min_{k \in 1 \dots K} \frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} ||y_{n,GT}^{t} - y_{n,k}^{t}||_2, \tag{1}$$

$$\text{meanSADE} = \frac{1}{NT} \sum_{k=1}^{K} p_\psi(Y_k|X) \sum_{n=1}^{N} \sum_{t=1}^{T} ||y_{n,GT}^{t} - y_{n,k}^{t}||_2, \tag{2}$$

where N is the number of actors, T is the number of timesteps, K is the number of scene samples for a given scenario, $p_\psi(Y_k|X)$ is the scenario probability score for scene sample k, $y_{n,k}$ is the predicted trajectory for actor n in the scene sample k, $y_{n,GT}$ is the ground truth trajectory for actor n.

We focused on measuring motion forecasting diversity that impacts the safety of the SDV, by evaluating how the diversity of these predictions impact the subsequent contingent plans. To do this, we measure the pairwise plan average self-distance (*meanPlanASD*), i.e., the average distance between the contingent plans for 2 distinct futures. Additionally, we also compute the scene average self-distance (*meanSASD*), which computes the average pairwise distance among scene samples, and the minimum scene self-distance (*minSASD*), which computes the minimum pairwise distance for each scene sample, as ways to measure general diversity as proposed by [15, 16]. In our implementation, meanSASD does not take into account scenario probabilities of each predicted future, as it aims to measure their diversity.

$$\text{meanPlanASD} = \frac{1}{K} \sum_{i=1}^{K} \sum_{j \neq i}^{K} \ell_2(\tau_i, \tau_j) \tag{3}$$

where $\tau_i = \tau(Y_i)$ is the corresponding planned contingent trajectory to the scene-level future Y_i.

$$\text{meanSASD} = \frac{1}{K} \sum_{i=1}^{K} \sum_{j \neq i}^{K} \ell_2(Y_i, Y_j) \tag{4}$$

$$\text{minSASD} = \sum_{i=1}^{K} \min_{k \in 1 \dots K} \ell_2(Y_i, Y_k) \tag{5}$$

where Y_i is the tensor of coordinates of the future trajectories in scene i, with dimensions $(N, K, T, 2)$.

2.3. Open-sourced datasets and simulators

We chose not to run our metrics on nuScenes [1] and Argoverse [6] as they do not have a closed-loop simulator, and open-loop metrics in general do not reflect the quality of real-world (closed-loop) driving due to compounding errors and causal confusion. Moreover, the open-loop planning metrics used in previous works cannot be directly applied to our multiple contingency plans.

2.4. Baselines

All of these baselines share the shared perception backbone, object detection, and actor feature extraction models as described above. These baselines are divided into actor-independent models that either output explicit marginal likelihoods (i.e. MultiPath [5]) or output sampled trajectories such as our CVAE[11], DPP [15] and DLow [16] models, and scene-consistent models that are autoregressive (ESP [9]) or implicit variable models (ILVM [3]).

As detailed in [3], for MultiPath, we use their mixture of trajectories parameterization instead of our encoder-decoder architecture to predict a gaussian for each waypoint.

For the CVAE model, we replace the Encoder, Prior and Decoder SIMs with MLPs of similar dimension, but use the same variational inference parameterization.

Prediction	Planning	CR(%)	$\frac{\text{Progress}}{collision}(m)$	Progress(m)	Jerk$(\frac{m}{s^3})$	Lat.Acc.$(\frac{m}{s^2})$	Acc$(\frac{m}{s^2})$	Decel$(\frac{m}{s^2})$
CVAE-DPP	PLT	17.07	123.97	21.17	11.99	**0.06**	1.11	0.80
	Contingency	12.80	235.21	30.12	8.83	0.16	1.03	0.54
CVAE-DLow	PLT	14.63	377.07	55.18	5.22	0.15	0.84	0.54
	Contingency	9.76	628.67	61.33	4.44	0.36	0.79	0.36
MultiPath	PLT	12.20	394.37	48.09	12.92	0.13	1.24	0.80
	Contingency	10.37	548.72	56.88	7.62	0.33	1.08	0.57
ESP	PLT	11.59	464.44	53.81	6.52	0.15	0.89	0.57
	Contingency	10.98	549.89	60.35	5.20	0.35	0.82	0.53
ILVM	PLT	10.98	553.96	60.80	5.50	0.16	0.86	0.56
	Contingency	9.15	709.60	**64.90**	**4.40**	0.38	**0.77**	**0.52**
CVAE	PLT	8.54	655.22	55.93	7.22	0.15	0.96	0.62
	Contingency	9.76	630.50	61.51	5.07	0.36	0.82	0.53
LookOut		**7.93**	**790.37**	62.65	4.69	0.37	0.79	0.53

Table 1. **Closed loop motion planning comparison against the baselines with Contingency Planner.**

For the DPP model, we use the frozen Decoder from the CVAE model. Similarly to [15], instead of predicting a μ and σ vectors, we predict K scenes samples of latent vectors using an MLP, where each scene sample consists of a 64-dimensional latent vector for each actor. We do this by predicting a $K * 64$-dimensional vector for each actor, and reshaping that into K vectors of length 64 per actor. Then, we decode each scene sample of latent vectors to get K separate scene motion forecasts Y. We then apply the determinantal point process loss as described in [15]. To get the x_i vectors, as referred to their work (not to be confused with the LOOKOUT definition), we concatenate the trajectories in one scene over all the actors, their (x, y) coordinates over time for each scene to get a $N * T * 2$-dimensional vector x_i for each scene i, where N is the number of actors and T is the number of future timesteps predicted. To get the z_i vectors, as referred to their work (not to be confused with the LOOKOUT definition), we concatenate the per-actor vectors in each scene to get $N * 64$-dimensional vectors z_i for each scene i. We use these vectors to compute the Diversity Loss as described in their paper.

For the DLow model, we use the frozen Decoder from the CVAE model. Similarly to [16], we predict K scene samples of an A and B vector for each agent using an MLP. Unlike their paper, we output an A vector representing the diagonal of the matrix, because the full matrix would not fit in memory given our high dimensionality. These A and B are essentially used in the same way as described in section 3.2 of our paper, except planning-based diversity and scenario probability scoring are not used.

For our ESP model, we adapt it to our feature contexts and memory constraints as described in [3].

For our ILVM baseline, we use exactly the formulation described in [3] (where SIMs are used for the Encoder, Prior and Decoder), except we use a fixed standard gaussian distribution as our target encoder distribution instead leaving it unconstrained. The same is true for our CVAE model, and LOOKOUT. This is because it allows our DPP, DLow and LOOKOUT models to more easily learn to fit and map to the distribution of the latent space, making training much more tractable.

3. Additional Evaluation Results

Closed-loop experiments with baselines with Contingency Planner: We see in Table 1 that the contingency planner increases the progress and decreases the acceleration and deceleration of the motion planning for all baselines. Additionally, it increases the lateral acceleration for all baselines, possibly in order to make more active maneuvers to nudge around obstacles. We see that the LOOKOUT still maintains the best safety and progress per collision even when the baselines are paired with our contingency planner, and has similar values in other metrics to the other most competitive baseline, ILVM + Contingency Planner. This results demonstrates the necessicity of using both scene-consistent diverse predictions, and the contingency planner.

Multi-class motion forecasting comparison in ATG4D: We can see in Table 2 that LOOKOUT's prediction module most accurately models the ground truth future trajectories for vehicles, with the lowest minimum and mean scene average

Category	Model	minSADE (m)	meanSADE (m)	minSASD (m)	meanSASD (m)
Vehicles	MultiPath	0.929	1.314	1.175	4.628
	CVAE	0.804	1.083	0.488	2.693
	CVAE-DPP	1.143	4.267	3.551	19.849
	CVAE-DLow	0.839	1.152	0.559	3.277
	ESP	1.090	1.441	1.120	3.991
	ILVM	0.770	1.061	0.455	2.534
	LOOKOUT	**0.765**	**1.022**	0.704	4.078
Pedestrians	MultiPath	0.531	0.691	0.619	1.960
	CVAE	**0.527**	**0.550**	0.075	0.284
	CVAE-DPP	0.668	3.748	4.477	17.511
	CVAE-DLow	0.552	0.556	0.018	0.102
	ESP	0.547	0.637	0.750	1.424
	ILVM	0.562	0.576	0.069	0.239
	LOOKOUT	0.583	0.582	0.085	0.811
Bicyclists	MultiPath	0.480	0.708	0.675	2.244
	CVAE	0.514	0.636	0.249	0.963
	CVAE-DPP	0.553	3.890	4.476	17.797
	CVAE-DLow	0.510	0.629	0.103	0.574
	ESP	0.601	0.925	0.770	2.239
	ILVM	**0.465**	0.633	0.178	0.807
	LOOKOUT	0.510	**0.627**	0.164	0.770

Table 2. **Multi-class motion forecasting results** in ATG4D ($K = 15$ samples).

Model	mAP (%) Vehicles		mAP (%) Pedestrians		mAP (%) Bicyclists	
	IoU 0.5	IoU 0.7	IoU 0.1	IoU 0.3	IoU 0.1	IoU 0.3
MultiPath	93.95	82.77	78.47	75.48	62.72	56.19
ESP	95.10	84.95	80.97	77.57	70.21	63.29
CVAE	95.76	87.98	**81.48**	**78.67**	74.31	68.76
LOOKOUT	**95.80**	**87.99**	80.66	78.32	**75.18**	**69.48**

Table 3. **Multi-class detection results** in ATG4D.

displacement error. This is important because vehicles make up the vast majority of actors on the road. Compared to the baseline with the next lowest error, ILVM, we have much greater prediction diversity (61% greater meanSASD).

We can see that for pedestrians and bicyclists, our model maintains a competitive accuracy/diversity tradeoff. We leave the problem of improving accuracy while maintaining the diversity for pedestrians and bicyclists while not regressing in vehicles for future work.

Detection comparison in ATG4D: We see in Table 3 that our detector's mAP on vehicles is 0.958 and is the highest among competitors for vehicles and bicyclists, and is comparable to the best performing baseline on pedestrians. The detection performance amongst most models is similar since they all share the perception backbone network, as explained in Section 2.3. The difference stems from the differences in prediction loss in the joint perception and prediction training (stage 1 in LOOKOUT).

Planning vs. diversity tradeoffs when using contingency planner: We see in Fig 1 that LOOKOUT offers the safest plans in closed-loop experiments, even when we pair the baselines with the contingency planner. We achieve a strong tradeoff between safety and meanPlanASD, with a 3% absolute improvement in safe-rate (or equivalently, 24% lower rate of collisions) than the only baseline that has greater planning diversity, MultiPath. This shows that our model is generating diverse predictions that are relevant to the SDV and accurately anticipate possible safety-critical scenarios the SDV should prepare for.

In addition, we see that our SDV demonstrates a competitive tradeoff between progress and planning diversity in Fig 1.

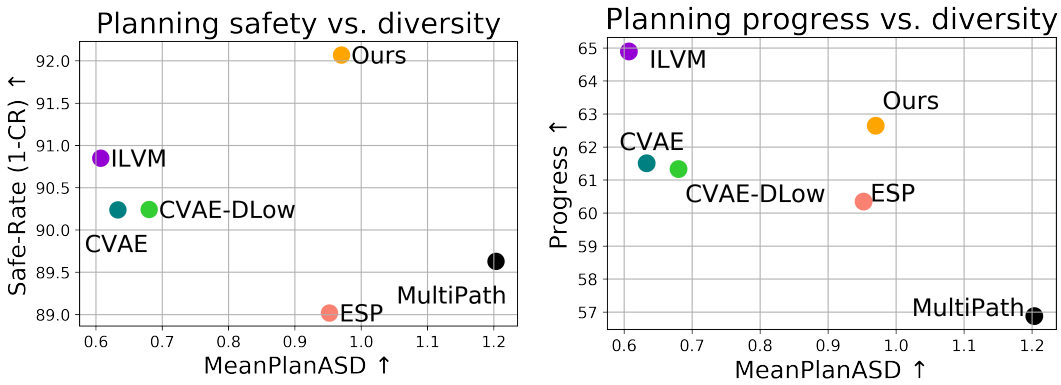

Figure 1. **Planning quality as a function of diversity** when we pair the baseline prediction models with **our contingency planner**.

Qualitatively, we see our motion planning not make as much progress compared to the ILVM baseline, in order to slow down in situations of uncertainty to avoid collisions. On the other hand, having too many unrealistic, varied trajectories and a lack of realistic trajectories can result in a low rate of progress (such as in MultiPath) because the SDV struggles to avoid all the predicted trajectories of other actors.

4. Additional Visualizations

Contingency planner visualizations: Fig 2 contains visualizations for interactive situations that showcase why we need contingency planning.

In Scenario 1, the SDV must decide to go before or after the left turning vehicle at the intersection, which plans to merge into the SDV lane. Since the intersection is crowded and we do not know the exact moment the other vehicle is going to go through, our predictions are diverse in terms of the other actor's speed, planning corresponding safe plans for the distinct futures while keeping a comfortable velocity.

Scenarios 2 and 6 showcase scenarios where the vehicle at the other side of the junction is planning to perform an unprotected left-turn, and the model are not sure whether it will yield to the SDV or not. The SDV plans a cautious immediate action that will allow it to decide to either go or brake when the action from the other actor is more clear.

Scenario 3 shows that when we are following an actor in the right-most lane, we plan slower trajectories in case we need to brake if the actor decides to turn right, and thus slow down.

In Scenario 4, the SDV is taking an unprotected left turn in high moving traffic and considers multiple speed profiles to account for the multiple futures of the incoming traffic.

Finally, Scenario 5 showcases a narrow passage due to parked vehicles, where another actor might either aggressively nudge the parked cars or gently progress. The SDV does not immediately hard-brake because it can plan a safe immediate action that is comfortable and allows it to postpone the decision to whenever more evidence is available or it becomes unsafe to progress.

Motion forecasting Sample Diversity and Quality: From Scenario 1 in Fig 3, we can see that LOOKOUT shows a diverse range of modalities, and most strongly predicts the forwards acceleration of the SDV (located in the center of the image, next to the curb).

From Scenario 2 in Fig 3, we see that the left turning vehicle in the center and right turning vehicle about it has a wide range of expected turn modalities. MultiPath demonstrates a wide spread of expected behaviours - however, many in the top left enter the curb, and the left turn on the agent driving from the right is not represented. ESP does the best job in fitting the SDV trajectory, at the cost of diversity.

From Scenario 1 in Fig 4, we can see LOOKOUT and MultiPath predict multiple modalities at both intersections, whereas ESP and ILVM trajectories fit one mode predominantly at each intersection.

From Scenario 2 in Fig 4, we mainly see that MultiPath struggles here in making realistic predictions, with many entering the curb. On the other hand, LOOKOUT demonstrates diversity mostly in speed and the trajectory of the bus, which is in the path of the SDV and thus is important to model diversely.

Figure 2. **Contingency planner qualitative results** in closed-loop simulation. The green bounding box is the SDV. The immediate action (1s) is shown in black starting from the rear axle of the SDV. The contingent trajectories planned for each possible future scenario are shown in distinct colors.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 4

[2] S. Casas, C. Gulino, R. Liao, and R. Urtasun. Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 2, 3

[3] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. *ECCV 2020*, 2020. 1, 2, 3, 4, 5

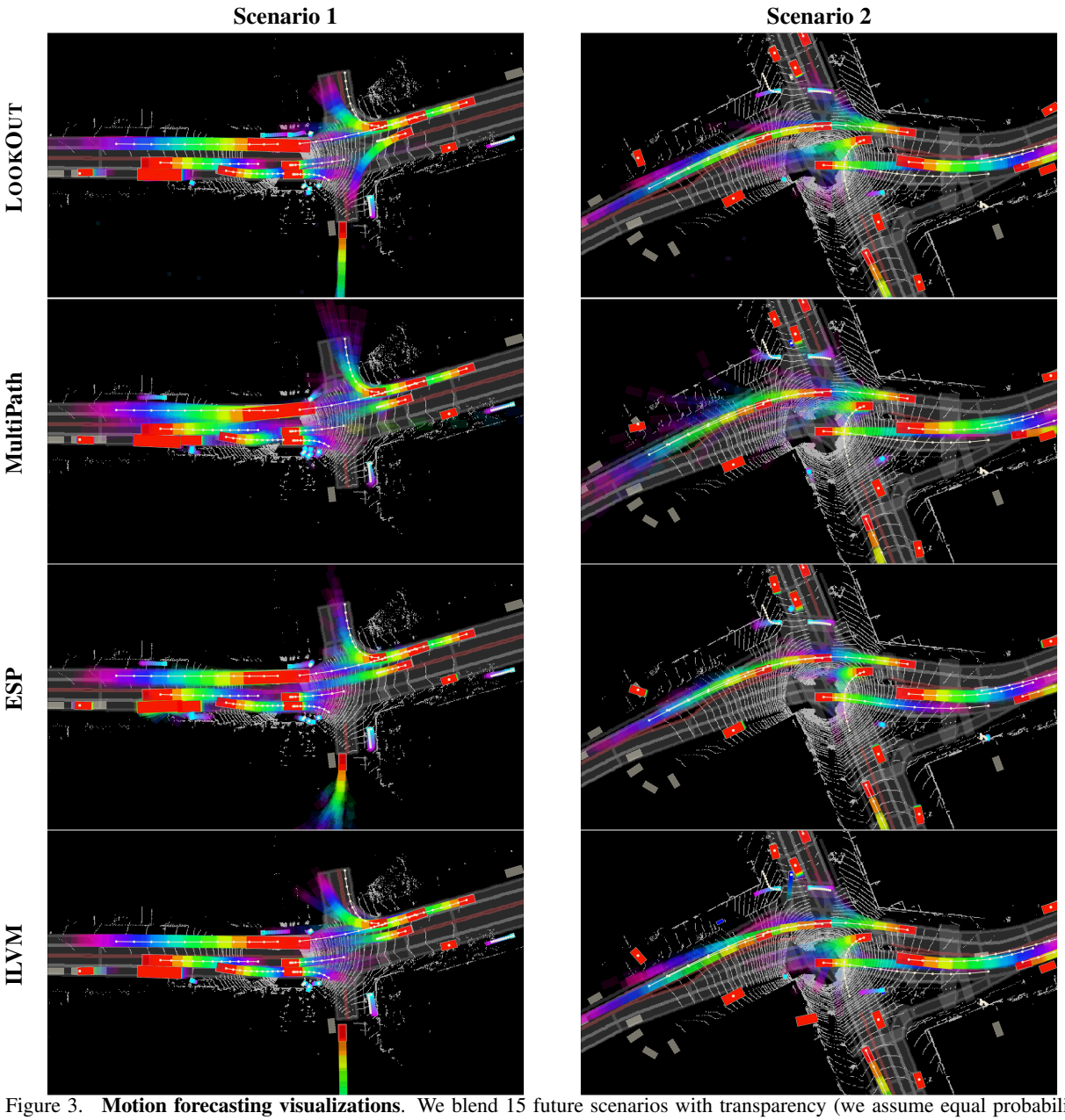

Figure 3. **Motion forecasting visualizations**. We blend 15 future scenarios with transparency (we assume equal probability for visualization purposes). Time is encoded in the rainbow color map ranging from red (0s) to pink (5s). This can be seen as a sample-based characterization of the per-actor marginal distributions.

[4] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, 2018. 1

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 4

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. *CoRR*, abs/1911.02620, 2019. 4

[7] Jing Huang, Viswanath Sivakumar, Mher Mnatsakanyan, and Guan Pang. Improving rotated text detection with rotation region proposal networks, 2018. 2

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[9] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 4

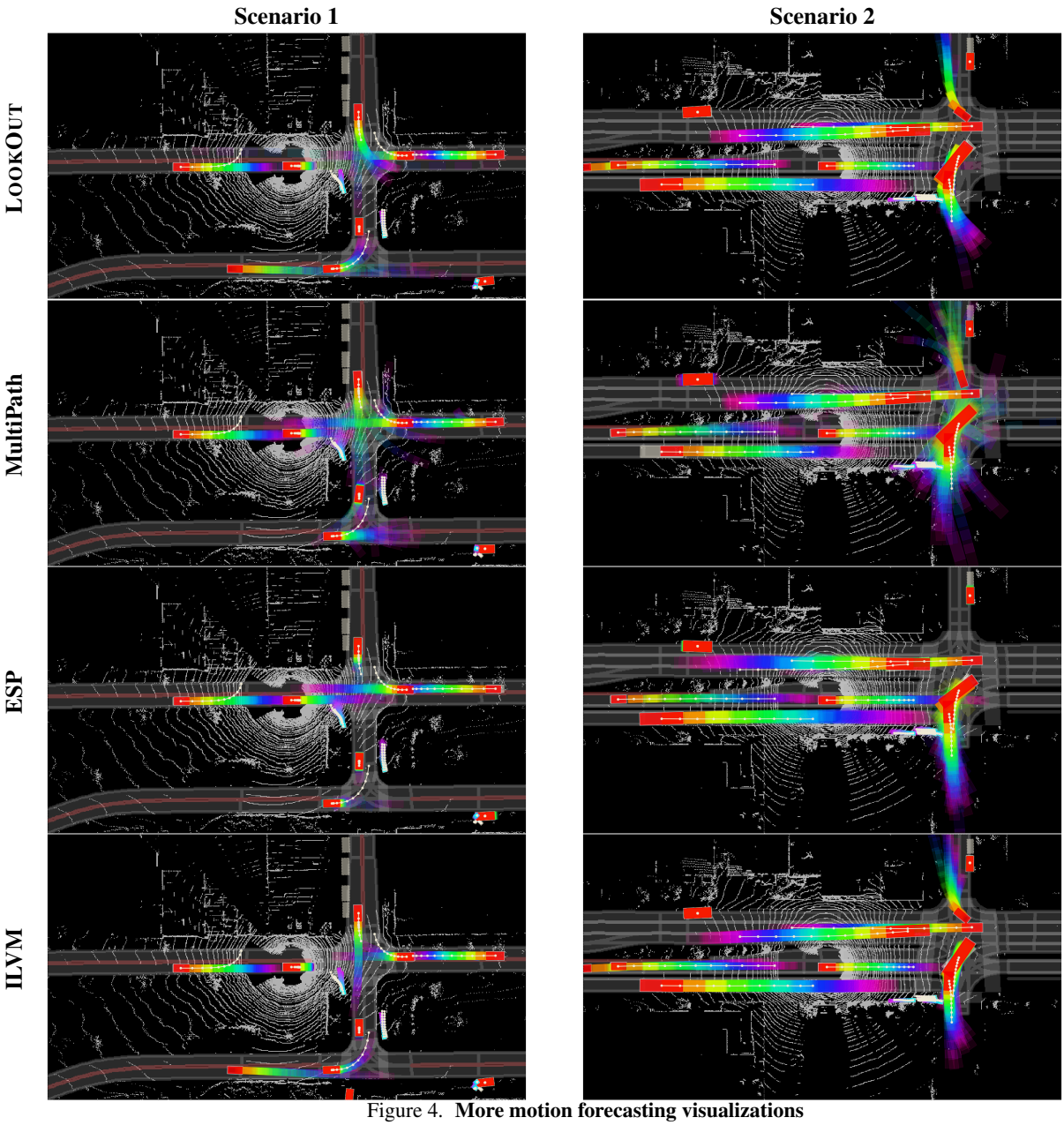

Figure 4. **More motion forecasting visualizations**

[10] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. *IROS 2019*, 2019. 3

[11] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015. 4

[12] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010. 3

[13] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the ECCV (ECCV)*, 2018. 1

[14] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE CVPR*, 2018. 1

[15] Ye Yuan and Kris Kitani. Diverse trajectory forecasting with determinantal point processes. *arXiv preprint arXiv:1907.04967*, 2019. 4, 5

[16] Ye Yuan and Kris Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. *arXiv preprint arXiv:2003.08386*, 2020. 2, 4, 5