Likelihood-Based Diverse Sampling for Trajectory Forecasting

Yecheng Jason Ma
University of Pennsylvania
jasonyma@seas.upenn.edu

Jeevana Priya Inala
MIT CSAIL
jinala@csail.mit.edu

Dinesh Jayaraman, Osbert Bastani
University of Pennsylvania
{dineshj, obastani}@seas.upenn.edu

Abstract

Forecasting complex vehicle and pedestrian multi-modal distributions requires powerful probabilistic approaches. Normalizing flows (NF) have recently emerged as an attractive tool to model such distributions. However, a key drawback is that independent samples drawn from a flow model often do not adequately capture all the modes in the underlying distribution. We propose ***L****ikelihood-Based* ***D****iverse* ***S****ampling (LDS), a method for improving the quality and the diversity of trajectory samples from a pre-trained flow model. Rather than producing individual samples, LDS produces a set of trajectories in one shot. Given a pre-trained forecasting flow model, we train LDS using gradients from the model, to optimize an objective function that rewards high likelihood for individual trajectories in the predicted set, together with high spatial separation among trajectories. LDS outperforms state-of-art post-hoc neural diverse forecasting methods for various pre-trained flow models as well as conditional variational autoencoder (CVAE) models. Crucially, it can also be used for transductive trajectory forecasting, where the diverse forecasts are trained on-the-fly on unlabeled test examples. LDS is easy to implement, and we show that it offers a simple plug-in improvement over baselines on two challenging benchmarks. Code is at:* https://github.com/JasonMa2016/LDS

1. Introduction

A key challenge facing self-driving cars is accurately forecasting the future trajectories of other vehicles. These future trajectories are often diverse and multi-modal, requiring a forecasting model to predict not a single ground truth future but the full range of plausible futures [25].

With the increasing abundance of driving data [4, 6], a promising approach is to learn a deep generative model from data to predict the probability distribution over future trajectories [24, 15, 18, 37, 33, 34, 36, 30]. However, due to natural biases, sampling i.i.d. from a deep generative model's prior distribution may fail to cover all modes in the trajectory distribution, especially given the uneven distribution of real-world traffic maneuvers. Consider the scenario in Figure 1(a) and a normalizing flow (NF) [32] forecasting model [35, 34]. The i.i.d. forecasts from the flow model in Figure 1(b) successfully capture the major mode corresponding to turning right; however, for driving safely at this intersection, we must also anticipate the minor mode corresponding to vehicles driving straight.

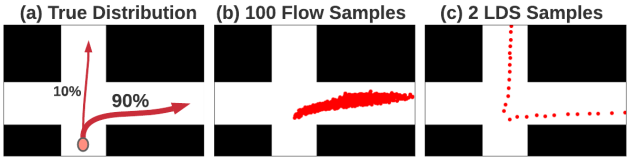

Figure 1: (a) At this intersection, 90% of cars turn right, and 10% drive straight in our training dataset. (b) A normalizing flow trajectory predictor trained on this data, sampled 100 times i.i.d., does not produce any straight trajectories. (c) With our LDS sampler plugged in, the same predictor generates both straight and right-turn trajectories with just 2 samples. Details are in Appendix B.

We propose a general, *post-hoc* approach, called **L**ikelihood-Based **D**iverse **S**ampling (LDS), for enhancing the quality and the diversity of samples from a pre-trained generative model. The key idea is that rather than drawing i.i.d. samples from the generative model, LDS learns a sampling distribution over an entire *set* of trajectories, which jointly maximizes two objectives: (i) the likelihood of the trajectories according to the model, and (ii) a robust goal-based diversity objective that encourages high final spatial separation among trajectories. Intuitively, these two objectives together encourage a set of forecasts to cover modes in the underlying trajectory distribution. The result of running LDS on our toy example is shown in Figure 1(c); it correctly discovers the minor mode of heading straight and distributes samples over both modes. Figure 2 provides an overview of the LDS objective and architecture. Because our technique leverages trajectory likelihood under the learned underlying generative model, it is naturally suited for NF-based

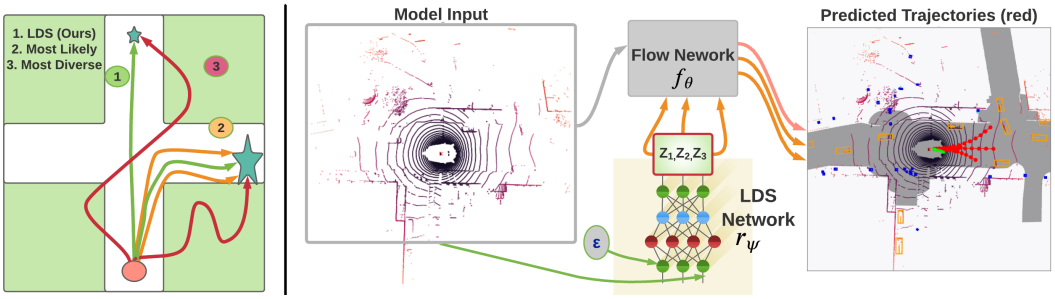

Figure 2: **Left**: Fitted on data at this intersection where most vehicles turn right and a small fraction head straight, **(1)**: LDS predicts a set of paths that cover both modes and are realistic, **(2)**: Optimizing for only model likelihood generates samples that are realistic but miss the minor mode (small star) at the top, and **(3)**: Optimizing for only diversity generates samples that may cover both modes but are not realistic. **Right**: LDS architecture overview. LDS replaces standard i.i.d. sampling in the flow model with a learned joint distribution over a set of samples in the latent space. These samples allow the pre-trained flow model to output diverse and realistic trajectories.

models, which compute the exact likelihood of their generated samples. To the best of our knowledge, our method is the first diverse sampling technique tailored for NF models. We note, however, that our technique can also be applied to other likelihood-based models—e.g., we can handle VAEs by using the evidence lower-bound for the likelihood.

A key advantage of LDS is that it can be leveraged in the setting of *transductive learning* [38]. In particular, since LDS does not require paired trajectories (history and future) for training, it can directly tailor the sampler to improve predictions for a given novel test instance, even without any prior training. By specializing to this test instance, transductive learning can outperform supervised learning which cannot perform test-time adaptation. To the best of our knowledge, the tranductive setting has not been previously considered for trajectory forecasting, but we believe it most closely mirrors the forecasting task autonomous vehicles face in the real world.

LDS is simple to implement, requiring fewer than 30 lines of code. We evaluate LDS on nuScenes [4] and Forking Paths [25], two challenging single-future and multi-future [25] benchmarks. Our experiments demonstrate that LDS provides a reliable performance boost when plugged into various existing NF/CVAE forecasting models and outperforms competing approaches. These results further improve when LDS is applied for transductive forecasting.

2. Related Work

Multi-Modal Forecasting. There are various approaches to the multi-modal forecasting problem. One approach is to pre-define trajectory primitives that serve as candidate outputs and formulate the forecasting problem as a mix of classification and regression [10, 5, 9, 31]. However, these approaches require extra labeled information; furthermore, they often only output a deterministic set of predictions for a given input. We instead build on deep generative models [15, 24, 37, 18, 36, 35, 30] that directly model multi-modal densities. In particular, normalizing flows [32, 29] have become a popular choice due to their comparative ease of optimization [33, 30, 35, 13], as well as their flexibility for downstream tasks such as goal-conditioned planning [34]. Our method is designed as a plug-in improvement for sampling better and more diverse predictions from any model within this family. Our method is also compatible with VAE [21]-based forecasting models [24, 40, 36, 27]. Our diversity loss is formulated using predicted trajectories' endpoints. Similar ideas have been explored recently [27, 42, 7], but our method is a post-hoc approach and infers plausible endpoints in an unsupervised manner while previous works are end-to-end architectures and require ground truth endpoints to guide model training.

Post-Hoc Neural Diversity Sampling. Several prior works learn to sample from pre-trained generative models [3, 14, 12, 27]. Our approach is most closely related to two recent neural post-hoc diverse sampling methods for forecasting, DSF [40] and DLow [41]. DSF uses a neural network to parameterize a deterministic point process (DPP) [22] from which samples are drawn. The DPP kernel chooses samples using a threshold function based on the Euclidean distance of the latent samples from its prior distribution mean. DSF inherits the disadvantages of slow training and inference from its DPP. Computational issues aside, DSF fails to scale to high-dimensional latent spaces where Euclidean distance is not an informative metric of closeness. In comparison, DLow uses a modified ground truth trajectory reconstruction loss and KL divergence to shape the latent samples and an exponential kernel function to induce sample diversity. However, it restricts the architecture of its sampling network to be a single-layer linear network with respect to the latent samples to admit tractable KL-constraint computa-

tion in its objective. This limits the expressiveness of the learned sampling distribution. In addition, because its objective requires ground-truth trajectory futures for training, it cannot be used in the transductive setup we introduce in our experiments. Both DSF and DLow also introduce additional kernel-related hyperparameters that are difficult to optimize. Compared to DSF and DLow, LDS permits multi-layered sampling architectures and high-dimensional latent spaces, exploits trajectory likelihood under the generative model to permit flexible application including transductive forecasting, introduces few hyperparameters, and performs consistently better across our experiments.

3. Problem Setup

Consider the problem of predicting the trajectory of an agent whose 2D position at time t is denoted as $\mathbf{S}_t = (x_t, y_t)$. We denote the current time step as $t = 0$, and the future aggregated state as $\mathbf{S} \coloneqq \mathbf{S}_{1:T} \in \mathbb{R}^{T\times 2}$. At time $t = 0$, the agent has access to observation $\mathbf{o}$, which may include contextual features such as Lidar scans, physical attributes of the vehicle/pedestrian agent (e.g. velocity, yaw), and the state histories of all agents in the scene. The goal of trajectory forecasting is to predict $\mathbf{S}$ given $\mathbf{o}$, $p(\mathbf{S}|\mathbf{o})$. We denote the training dataset as $\mathcal{D} = \{(\mathbf{o}, \mathbf{S})\}$.

Our approach assumes as given, a flow model f_θ that has been pre-trained to learn the distribution $p_\theta(\mathbf{S}|\mathbf{o}; \mathcal{D})$. At a high level, assuming a multivariate Gaussian base sampling distribution $\mathbf{Z} \sim P_\mathbf{Z} \equiv \mathcal{N}(0, \mathbf{I})$, f_θ is a bijective mapping between $\mathbf{Z}$ and $\mathbf{S}$, captured by the following forward and inverse computations of f_θ:

$$\mathbf{S} = f_\theta(\mathbf{Z}; \mathbf{o}) \sim p_\theta(\mathbf{S} \mid \mathbf{o}), \quad \mathbf{Z} = f_\theta^{-1}(\mathbf{S}; \mathbf{o}) \sim P_\mathbf{Z} \tag{1}$$

To draw one trajectory sample $\mathbf{S}$, we sample $\mathbf{Z} \sim P_\mathbf{Z}$ and compute $\mathbf{S} = f_\theta(\mathbf{Z}; \mathbf{o})$. Furthermore, the exact likelihood of a trajectory $\mathbf{S}$ is given by the change of variables rule:

$$\log p_\theta(\mathbf{S}|\mathbf{o}) = \log \left(p(\mathbf{Z}) \cdot \left| \det \frac{\mathrm{d} f_\theta}{\mathrm{d}\mathbf{Z}} \right|_{\mathbf{Z}=f_\theta^{-1}(\mathbf{S};\mathbf{o})} \right|^{-1} \Bigg), \tag{2}$$

where the bijective property and standard architectural choices for f_θ permit easy computation of the determinant. We refer readers to Appendix A for a more detailed introduction to flow-based trajectory forecasting.

4. Diversity Sampling for Flow

In stochastic settings, it is often necessary to use $K > 1$ trajectory predictions rather than just one, to ensure that the samples cover the full range of possible stochastic futures; we assume the number of predictions K is a given hyperparameter. However, as Figure 1 shows, simply drawing K i.i.d. samples from the flow model f_θ may undersample from minor modes and fail to capture all potential outcomes. We propose an alternative strategy, which we call **L**ikelihood-Based **D**iverse **S**ampling (LDS), that learns a joint distribution over K samples $\{\mathbf{Z}_1, ..., \mathbf{Z}_K\}$ in the latent space of f_θ. In doing so, it aims to improve the diversity of the trajectories $f_\theta(\mathbf{Z}_1), ..., f_\theta(\mathbf{Z}_K)$ while maintaining their plausibility according to the flow model[1].

In particular, LDS trains a neural network r_ψ to transform a Gaussian distribution $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ into a distribution over a set $\mathcal{Z} \coloneqq \{\mathbf{Z}_1, ..., \mathbf{Z}_K\} = r_\psi(\epsilon; \mathbf{o})$ of latent vectors given an observation $\mathbf{o}$. This set in turn induces a distribution over trajectories $\mathcal{S} \coloneqq \{\mathbf{S}_1, ..., \mathbf{S}_K\}$, where $\mathbf{S}_k = f_\theta(\mathbf{Z}_k; \mathbf{o})$ for each k. Since the distribution is defined over multisets of samples, the individual samples $\mathbf{S}_k$ are no longer independent. Informally, they should be anti-correlated to ensure they cover different modes. We train r_ψ to minimize the following loss function:

$$\mathrm{L}_{\mathrm{LDS}}(\psi) \coloneqq \mathrm{NLL}(\psi) - \lambda_d \mathrm{L}_\mathrm{d}(\psi), \tag{3}$$

which combines the negative log likelihood (NLL) loss from the flow model and a goal diversity loss L_d. Figure 2 (Left) provides intuition for these two terms, and we explain them in detail below.

Likelihood Objective. The NLL term is defined as:

$$\mathrm{NLL}(\psi) \coloneqq -\sum_{k=1}^{K} \log p_\theta(\mathbf{S}_k|\mathbf{o}_k), \tag{4}$$

where $\{\mathbf{S}_1, ..., \mathbf{S}_k\} = f_\theta(r_\psi(\epsilon; \mathbf{o}))$ and $\log p_\theta(\mathbf{S}_k|\mathbf{o})$ is computed as in Equation (2). This NLL term incentivizes LDS to output a set of forecasts that all have high likelihood according to the flow model f_θ. This term incentivizes f_θ to maximize the likelihood of the training trajectories $\mathcal{D}$ by selecting trajectories that are plausible and likely to occur. By itself, it does not incentivize diversity among forecasts; they may easily concentrate around the major mode as in the "most likely" trajectories in Figure 2 (Left).

Diversity Objective. To combat this tendency, we introduce the *minimum* pairwise squared L_2 distance between the predicted trajectory endpoints:

$$\mathrm{L}_\mathrm{d}(\psi) \coloneqq \min_{i\neq j\in K} \|f_\theta(\mathbf{Z}_i)_T - f_\theta(\mathbf{Z}_j)_T\|_2^2. \tag{5}$$

The minimum formulation strongly incentivizes LDS to distribute its samples among different modes in the distribution, since any two predictions that end up too close to each other would significantly decrease L_d. In our experiments, we have observed *mean*-based diversity formulation suffers from network "cheating" behavior, in which all but one prediction collapse to a single trajectory and the left-out trajectory is distant, resulting in comparatively high *average*

[1] Though our technical discussion focuses on NF models, we emphasize that LDS can also be applied to CVAE models by replacing NLL with ELBO; we provide details on LDS-CVAE in Appendix C.

Algorithm 1 Batch LDS Training

Input: Flow f_θ, Observation Batch $\{\mathbf{o}\}$

Initialize LDS model r_ψ
for $\mathbf{o}_i \in \{\mathbf{o}\}$ **do**
  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  Compute $\mathbf{Z}_1, ..., \mathbf{Z}_K = r_\psi(\epsilon; \mathbf{o}_i)$
  Generate predictions $f_\theta(\mathbf{Z}_1), .., f_\theta(\mathbf{Z}_K)$
  Compute losses using Equations 4 & 5
end for
Perform stochastic gradient descent on ψ to minimize $\mathrm{L}_{\mathrm{LDS}}$ (Equation 3)

Output: Trained LDS model r_ψ

diversity. Our formulation is robust to this degeneracy as only the pairwise minimum will be considered. While many other notions of distances between trajectories are compatible with our framework, we measure the distances at the last time step alone, since spatial separation between trajectory endpoints is a good measure of trajectory diversity in our applications [27, 42, 7]. Finally, to train r_ψ, LDS minimizes $\mathrm{L}_{\mathrm{LDS}}$ using stochastic gradient descent; see Algorithm 1.

Implementation Details. LDS r_ψ is implemented as a 3-layer feed-forward neural network with K heads, each of which corresponds to the latent z_k for a single output in the predicted set. We assume access to the input embedding layers of the pre-trained flow model, which embeds high-dimensional (visual) input $\mathbf{o}$ into lower dimensional feature vectors. These feature vectors are taken as input to LDS. Additionally, we fix the dimensions of the input Gaussian noise ϵ to be the same as the trajectory output $\mathbf{S}$ from f_θ (i.e. $T \times 2$). Furthermore, to prevent the diversity loss from diverging, we clip it to a positive value. Additional details are in Appendix F.5 & G.3.

5. Transductive Trajectory Forecasting

Transductive learning refers to "test-time training" on unlabeled test instances [38, 19]. For diversity sampling, it amounts to the following: *given a new observation* $\mathbf{o}$*, compute a set of diverse trajectories that best captures the possible stochastic futures*. That is, whereas supervised learning focuses on *average* accuracy across all test trajectories, transductive learning focuses on the vehicle's current situation. This setting closely captures the forecasting problem autonomous vehicles face in practice; however, existing end-to-end forecasting models typically lack the capability for the transductive setting because their training procedures require data with ground truth labels.

In contrast, LDS's objective function does not depend on access to any ground truth future trajectories, and relies only on the pre-trained generative model and unlabeled inputs $\mathbf{o}$. In this sense, the LDS sampler is trained without supervision. It can therefore be used transductively and adapt on-the-fly to a given new observation (e.g. a new traffic scene the vehicle enters). Formally, given an unlabeled input $\mathbf{o}$, we can train a LDS model r_ψ tailored to $\mathbf{o}$. We call this variant **LDS-TD-NN**. Compared to vanilla "batch" LDS, LDS-TD-NN does not need to be trained offline with a large dataset, making it also suitable for settings where little or no training data is available —e.g., the training set for the pre-trained flow model is unavailable.

In the transductive setting, LDS could eschew the neural network and directly optimize the latent space samples $\mathbf{Z}_1, ..., \mathbf{Z}_K$. We call this *particle* variant **LDS-TD-P**. Note that, unlike the neural variant, LDS-TD-P cannot take the observation $\mathbf{o}$ as input. LDS-TD-NN and LDS-TD-P are summarized in Algorithm 2 and 3 in Appendix D.

6. Experiments

Our experiments aim to address the following questions: **(1)** Does LDS boost performance across different pre-trained flow models? **(2)** Can LDS be applied to pre-trained VAE models as well?, **(3)** How does LDS compare against (a) other learning-based diverse sampling methods and (b) existing end-to-end multi-modal forecasting models?, **(4)** Is LDS effective in the transductive learning setting? **(5)** Which components of LDS are most important for performance? We address questions 1-4 via experiments on two qualitatively different datasets, nuScenes [4] and Forking Paths [25] (Section 6.3 and 6.4). For each one, we use a distinct pre-trained flow model with architecture and input representation tailored to that dataset. We address question 5 via an ablation study in Section 6.5.

6.1. Datasets and Models

We begin by describing our datasets, models, and evaluation metrics; see Appendix F & G for details on model architectures, hyperparameters, and training procedures.

NuScenes. NuScenes is a multi-purpose autonomous vehicle dataset with a large trajectory forecasting dataset [4]. Following prior work [9, 31, 13], the predictor takes as input the current observation (e.g., Lidar scan) and attributes (e.g., velocity) of a vehicle, and forecasts this vehicle's trajectory over the next 6 seconds (i.e., 12 frames).

LDS Models. We train an autoregressive affine flow model (**AF**) proposed by [35, 34] as our underlying flow model for trajectory prediction. On top of **AF**, we train three variants of LDS. The first is **LDS-AF**, the batch version in Algorithm 1. The latter two are the transductive neural and particle-based variants **LDS-AF-TD-NN** and **LDS-AF-TD-P** discussed in Section 5. To further illustrate the generality of LDS, we also consider **LDS-CVAE**, a LDS applied to a CVAE model. Here, we replace the exact likelihood in Equation 4 with ELBO.

Baselines. For neural diverse sampling approaches, we consider DLow and DSF and apply them to both CVAE and AF to form DLow-{AF, CVAE} and DSF-{AF, CVAE}. In addition to neural diverse sampling baselines, we include three end-to-end multi-modal forecasting models: **CoverNet** [31], **MultiPath** [5], and **MTP** [9]. For the first two, we directly report results published in [31]; for **MTP**, we re-train the model using nuScenes' official implementation to provide an end-to-end model for evaluation metrics not included in [31]. We choose these end-to-end models for comparison since they use the same inputs as our AF and CVAE backbones, and we aim to test whether post-hoc sampling methods applied to simple backbone models are competitive with specialized end-to-end models. In Appendix F.7, we compare to Trajectron++ [36], which uses an experimental setup different from other prior approaches.

Forking Paths. One limitation of most trajectory forecasting datasets such as nuScenes is that there is only a single ground-truth future trajectory for each training sample. To evaluate each forecasting model's ability to predict diverse, plausible future trajectories, it is critical to evaluate the model against multiple ground-truth trajectories in a multi-future dataset. This approach directly evaluates whether a model captures the intrinsic stochasticity in future trajectories. Therefore, we additionally evaluate LDS on the recent Forking Paths (FP) dataset [25]. FP recreates scenes from real-world pedestrian trajectory datasets [28, 2] in the CARLA simulator [11], and asks multiple human annotators to annotate future trajectories in the simulator, thereby creating multiple ground truth future pedestrian trajectories for each scene. The flow model takes as input the trajectory of a pedestrian over the past 3 seconds (i.e., 12 frames), and its goal is to predict their trajectory over the next 5 seconds (i.e., 20 frames).

LDS Models. For this dataset, we focus on flow models and use the recently introduced Cross-Agent Attention Model Normalizing Flow (**CAM-NF**) [30] as our underlying flow model for trajectory prediction; see Appendix G for CAM-NF details. Compared to AF used for nuScenes experiments, CAM-NF is an already performant generative model, and an additional goal of this experiment is to investigate whether LDS can be useful for already performant generative models. As before, we train **LDS** and **LDS-TD-NN** on top of CAM-NF.

Baselines. We compare to **DSF** and **DLow** applied to CAM-NF. All other baseline results are taken directly from [25] (rows with $*$ in Table 2 (Left)), including **Social-LSTM** [1], **Social-GAN** [15], **Next** [26], and **Multiverse** [25], as well as simple **Linear** and **LSTM** networks.

Training and Evaluation. We follow the procedure in [25]. We first train CAM-NF using VIRAT/ActEV [28, 2], the real-world datasets from which FP extracts simulated pedestrian scenes. Then, we train {DLow, DSF, LDS} on top of the pre-trained CAM-NF model using the training dataset (VIRAT/ActEV). Finally, we evaluate all models on the test set FP using $K = 20$ samples against the multiple ground-truth futures. For LDS-TD-NN, we directly train and evaluate r_ψ on FP using small minibatches as described in Algorithm 2 in Appendix D.

An important challenge is that trajectories in FP have different (typically longer) lengths compared to trajectories in VIRAT/ActEV, since the human annotators provided trajectories of varying durations; this complicates the forecasting problem on the FP test set.

6.2. Evaluation Metrics

We report minimum average displacement error **minADE**$_\text{K}$ and final displacement error **minFDE**$_\text{K}$ of K prediction samples $\hat{\mathbf{S}}_k$ compared to the ground truth trajectories $\mathbf{S}_1, ..., \mathbf{S}_J$ [37, 5, 25]:

$$\text{minADE}_K(\hat{\mathbf{S}}, \mathbf{S}) = \frac{\sum_{j=1}^{J} \min_{i \in K} \sum_{t=1}^{T} \|\hat{\mathbf{S}}_{i,t} - \mathbf{S}_t\|^2}{T \times J},$$

$$\text{minFDE}_K(\hat{\mathbf{S}}, \mathbf{S}) = \frac{\sum_{j=1}^{J} \min_{i \in K} \|\hat{\mathbf{S}}_{i,T} - \mathbf{S}_T\|^2}{J}$$

These metrics are widely used in stochastic prediction tasks [37, 15] and tend to reward predicted sets of trajectories that are both diverse and realistic. In multi-future datasets ($J > 1$) such as Forking Paths, these metrics are standalone sufficient to evaluate both the diversity and the plausibility of model predictions, because a set of predictions that does not adequately cover all futures will naturally incur high errors. In single-future datasets ($J = 1$) such as nuScenes, however, they do not explicitly penalize a predicted set of trajectories that simply repeats trajectories close to the single ground truth. To explicitly measure prediction diversity on nuScenes, we also report the minimum average self-distance **minASD**$_\text{K}$ and minimum final self-distance **minFSD**$_\text{K}$ between pairs of predictions samples:

$$\text{minASD}_\text{K}(\hat{\mathbf{S}}) = \min_{i \neq j \in K} \frac{1}{T} \sum_{t=1}^{T} \|\hat{\mathbf{S}}_{i,t} - \hat{\mathbf{S}}_{j,t}\|^2$$

$$\text{minFSD}_\text{K}(\hat{\mathbf{S}}) = \min_{i \neq j \in K} \|\hat{\mathbf{S}}_{i,T} - \hat{\mathbf{S}}_{j,T}\|^2$$

These metrics evaluate the lower bound diversity among a predicted set of trajectories, and they tend to decrease as K increases since the predictions become more "crowded" around the modes already covered. Note that minFSD is identical to the diversity term in the LDS objective (Equation (5)). Several prior works have reported the *average* ASD (meanASD) and FSD (meanFSD) instead [40, 41]; however, we observe that minASD is a superior metric since it is more robust to outliers among the predictions (see Appendix F.8 for an illustrative example). For completeness,

we also report meanASD and meanFSD in Appendix F.7; our findings are consistent with those presented here. Finally, since minFDE_K, minASD_K, and minFSD_K were not reported in previous work, we only report them for the models we implement.

All compared models, except vanilla CVAE and NF, take the number K of modes/number of samples in the prediction set as a hyperparameter. We report results for each metric using the corresponding model configurations—e.g., when measuring minASD_5, we use $K = 5$ for all models.

6.3. Quantitative Results

NuScenes. In Table 1 (Left), we compare the prediction accuracy of LDS-AF, LDS-AF-TD-{NN, P}, and the baselines described above. The best method within each subcategory is bolded. **LDS-AF** and **LDS-AF-TD-NN** achieve the best overall performance.

Comparing AF and CVAE to prior multi-modal models, we see that although the two "vanilla" generative models achieve better one-sample prediction (i.e., minADE_1), they perform significantly worse when more predictions are made. This confirms our hypothesis that i.i.d. samples from a generative model do not adequately capture diverse modes, causing it to fail to cover the ground truth with good accuracy. Consequently, post-hoc diverse sampling significantly improves the performance of these models. Out of all the post-hoc neural diverse sampling methods, LDS provides the most significant improvement for both AF and CVAE. In particular, the most performant model LDS-AF achieves the best results out of all models in the batch setting, even outperforming the strongest multi-modal model CoverNet. This suggests there is much to gain from a (simple) pre-trained model by applying appropriate post-hoc diverse sampling. We highlight that despite not being designed for CVAE, LDS still outperforms DSF and DLow, both of which are originally intended for CVAE, demonstrating the general merit of our approach. Next, in the transductive setting, LDS-AF-TD-NN is indeed able to tailor its predictions towards each small batch of test instances and achieves the overall best results among all models. Finally, LDS-AF-TD-NN also significantly outperforms the particle variant LDS-AF-TD-P, likely due to the neural variant having the advantage of explicitly conditioning the samples on the observation input $\mathbf{o}$.

Next, we compare the models in terms of their prediction diversity in Table 1 (Right). LDS models consistently outperform the baseline models by a large margin. In particular, they are the only models whose diversity does not collapse when the number of modes increases from 5 to 10. This shows that LDS is more "efficient" with its samples, since it does not repeat any trajectories. In contrast, all other methods produce pairs of very similar predictions when $K = 10$. Given that LDS also produces accurate predictions, these results provide strong evidence that LDS is able to simultaneously optimize accuracy and diversity. Furthermore, LDS also achieves the highest diversity under the mean diversity metrics in Appendix F.7.

Forking Paths. The training set results on ActEV/VIRAT for CAM-NF and various baseline models are included in Appendix G.5. To summarize the training set results, we find CAM-NF effective on this dataset, only outperformed by the current state-of-art method Multiverse [25], thus satisfying our goal of testing whether LDS can improve a strong backbone model. Now, we show the FP test results in Table 2 (Left). Note that the FP dataset comes with two different categories "45-Degree" and "Top Down" depending on the camera angle view of the human annotators. We report the average results between the two views for readability, and leave the full results split over each sub-category to Appendix G.6. We observe that CAM-NF already outperforms all prior methods on all metrics. With LDS, CAM-NF improves even further, outperforming all prior methods by a large margin. In comparison, DSF and DLow are not able to achieve the same level of performance boost, and in the case of DSF, the effect is even detrimental. The transductive variant LDS-TD-NN improves performance further on FDE metrics, while performing on par with LDS on ADE metrics; this results is promising as the transductive variant never observes the training set, and this dataset consists of a clear distributional shift between the training set and the testing set. In Appendix G.7, we provide additional analysis on Forking Paths results.

6.4. Qualitative Results

Next, we illustrate trajectories from LDS and baselines in the two benchmarks to demonstrate that LDS indeed outputs more diverse and plausible trajectories.

NuScenes examples. In Figure 3, we show visualizations of two separate frames of the same nuScenes instance, overlaid with predictions from LDS-AF, AF, and MTP. Overall, LDS produces the most diverse and plausible set of trajectories in both frames. In the first frame, AF exhibits a failure mode as some of its predictions go far off the road. This provides evidence that sampling i.i.d. from a vanilla flow model may fail to identify realistic trajectories. But when LDS is used to draw samples from the same flow model (i.e., LDS-AF), the trajectories become both more diverse and more realistic. In the second frame, MTP outputs a few trajectories that violate road constraints, while AF trajectories are concentrated in one cluster. Again, LDS-AF is the only model that predicts both diverse and plausible trajectories. In Appendix F.9, we provide additional visualizations including LDS-AF-TD-NN trajectories (Figure 8), as well as visualizations of different sets of trajectories LDS-AF outputs by varying the ϵ input to the model (Figure 9).

Method	Modes	$\text{minADE}_1(\downarrow)$	$\text{minADE}_5(\downarrow)$	$\text{minADE}_{10}(\downarrow)$	$\text{minFDE}_5(\downarrow)$	$\text{minFDE}_{10}(\downarrow)$	$\text{minASD}_5(\uparrow)$	$\text{minFSD}_5(\uparrow)$	$\text{minASD}_{10}(\uparrow)$	$\text{minFSD}_{10}(\uparrow)$
MultiPath* [5]	64	5.05	2.32	1.96	–	–	–	–	–	–
CoverNet* [31]	232	4.73	**2.14**	**1.72**	–	–	–	–	–	–
MTP [9]	5, 10	4.68 ± 1.04	2.61 ± 0.17	1.84 ± 0.04	5.80 ± 0.49	3.72 ± 0.07	1.74 ± 0.32	4.31 ± 1.60	0.97 ± 0.15	2.43 ± 0.34
CVAE	N/A	4.20 ± 0.03	2.71 ± 0.03	2.08 ± 0.02	6.20 ± 0.05	4.58 ± 0.05	1.28 ± 0.03	2.99 ± 0.07	0.57 ± 0.02	1.30 ± 0.04
DSF-CVAE [40]	5, 10	–	2.54 ± 0.21	2.02 ± 0.11	5.77 ± 0.51	4.44 ± 0.27	1.38 ± 0.22	3.33 ± 0.58	0.78 ± 0.04	1.85 ± 0.13
DLow-CVAE [41]	5, 10	–	2.23 ± 0.13	**1.75** ± 0.03	5.00 ± 0.29	**3.71** ± 0.08	2.64 ± 0.25	6.38 ± 0.65	1.18 ± 0.16	2.73 ± 0.43
LDS-CVAE (Ours)	5, 10	–	**2.16** ± 0.03	**1.75** ± 0.05	**4.82** ± 0.06	**3.71** ± 0.14	**3.02** ± 0.23	**7.46** ± 0.44	**1.74** ± 0.46	**4.07** ± 1.10
AF	N/A	4.01 ± 0.05	2.86 ± 0.01	2.19 ± 0.03	6.26 ± 0.05	4.49 ± 0.07	1.58 ± 0.02	3.75 ± 0.04	0.70 ± 0.01	1.63 ± 0.02
DSF-AF	5, 10	–	2.61 ± 0.12	2.23 ± 0.10	5.91 ± 0.33	4.80 ± 0.23	0.87 ± 0.13	2.14 ± 0.41	0.44 ± 0.05	1.11 ± 0.10
DLow-AF	5, 10	–	**2.11** ± 0.01	1.78 ± 0.05	**4.70** ± 0.03	3.77 ± 0.13	2.56 ± 0.12	6.45 ± 0.24	1.05 ± 0.11	2.55 ± 0.28
LDS-AF (Ours)	5, 10	–	**2.06** ± 0.09	**1.66** ± 0.02	**4.67** ± 0.25	**3.58** ±0.05	**3.13** ± 0.18	**8.19** ± 0.26	**2.11** ± 0.05	**6.22** ± 0.09
LDS-AF-TD-P (Ours)	5, 10	–	2.46 ±0.09	1.91 ± 0.04	5.21 ± 0.15	3.71 ± 0.11	2.39 ± 0.08	7.07 ± 0.18	1.60 ± 0.06	5.70 ± 0.10
LDS-AF-TD-NN (Ours)	5, 10	–	**2.06** ± 0.03	**1.65** ± 0.02	**4.62** ± 0.07	**3.50** ± 0.05	**3.09** ± 0.07	**8.15** ± 0.17	**1.98** ± 0.03	**5.91** ± 0.04

Table 1: NuScenes prediction error results (lower is better) and diversity results (higher is better), including previously reported results (top), and results of LDS variants and newly implemented baselines (bottom). LDS-based models produce the most plausible and diverse predictions throughout.

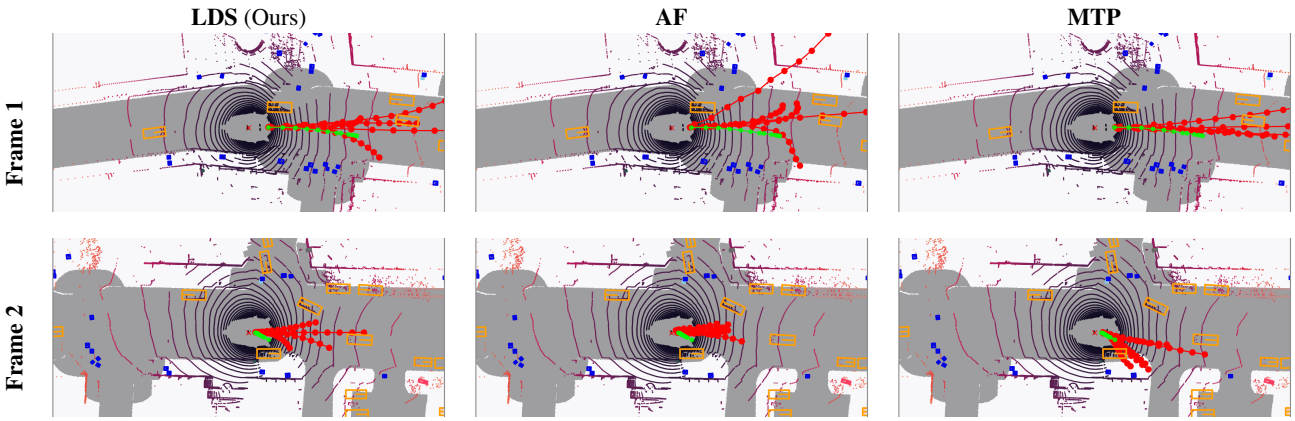

Figure 3: Model trajectory forecasts at two separate frames in the same scene. $K = 5$ predicted trajectories are shown in red, and the true recorded future trajectory from the dataset is shown in green. **LDS** predicts more diverse and plausible trajectories than both baselines.

Forking Paths examples. Visualizations of LDS (on top of CAM-NF), CAM-NF, and Multiverse predictions on the FP test set are shown in Figure 4. Additional visualizations are provided in Figure 11 in Appendix G. Again, LDS outperforms the other two methods and is the only approach that comes close to covering the diverse ground-truth futures.

6.5. Ablation Study

We conduct an ablation study to understand the impact of the various design choices in LDS—specifically, the importance of the different parts of the LDS loss function to its empirical performance. To this end, we train four ablations of LDS-AF on nuScenes. The first two omit one of the terms in the LDS objective—i.e., one without the diversity loss (**Ours w.o. Diversity**), and one without the likelihood loss (**Ours w.o. Likelihood**). The latter two modify the two LDS loss terms: one replaces the NLL loss (Equation 4) with DLow's Reconstruction+KL losses (see Appendix F for details) (**Ours w. Rec**), and the other replaces the minimum in the diversity loss (Equation 5) with the mean (**Ours w. meanDiv**). All four of these models are trained using the same procedure as LDS. As shown in Table 2 (Right), the first two ablations significantly deteriorate performance. As expected, **Ours w.o. Diversity** records close to zero diversity, and **Ours w.o. Likelihood** achieves high diversity but at the cost of plausibility. Note that **Ours w.o. Diversity** also performs poorly in terms of accuracy; this result shows that diversity is necessary to achieve good accuracy due to the stochastic nature of future trajectories. Thus, both terms in the LDS objective are integral to its success, and taking away either completely erases its benefits.

Next, we find that **Ours w. Rec** also reduces performance. This result demonstrates that leveraging the generative model's likelihood better captures the ground truth trajectory future. Finally, **Ours w. meanDiv** significantly reduces overall performance—the prediction error increases two-fold while the diversity metrics collapse. This result demonstrates the importance of using the more robust mini-

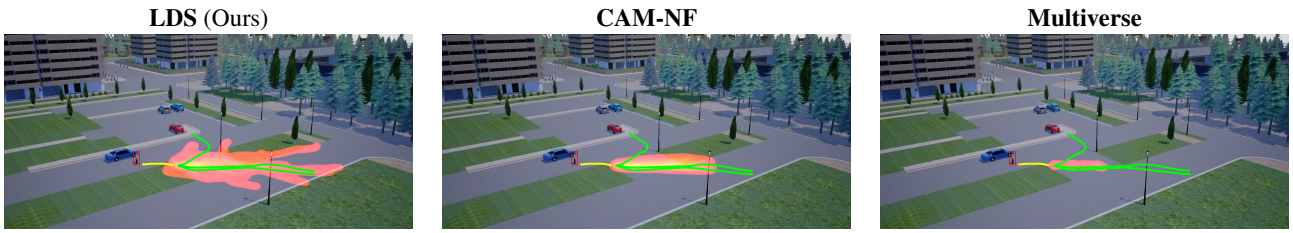

Figure 4: Visualization of predictions from various models on a single scene from the Forking Paths dataset. The red-yellow heatmap corresponds to the visited state density from the 20 predicted trajectories from each model; yellow indicates higher density. The green lines are the ground-truth human annotated futures. **LDS** produces diverse forecasts that cover the diverse futures, while the other two methods appear to have collapsed to a single output.

Method	$\text{minADE}_{20}(\downarrow)$	$\text{minFDE}_{20}(\downarrow)$
Linear*	205.0 ± 0.0	388.0 ± 0.0
LSTM*	192.4 ± 2.2	368.3 ± 3.4
Social-LSTM* [1]	189.0 ± 1.7	363.7 ± 3.0
Social-GAN* [15]	179.9 ± 4.3	334.4 ± 9.0
Next* [26]	176.8 ± 2.4	343.4 ± 6.1
Multiverse* [25]	163.3 ± 2.3	325.2 ± 3.5
CAM-NF [30]	148.0 ± 2.3	293.6 ± 4.7
DSF [40]	162.8 ± 2.0	320.6 ± 3.6
DLow [41]	137.8 ± 5.9	273.4 ± 14.0
LDS (Ours)	**98.6** ± 5.8	**182.0** ± 14.5
LDS-TD-NN (Ours)	**100.0** ± 3.2	**178.1** ± 7.6

Method/Metric	$\text{mADE}_5(\downarrow)$	$\text{mFDE}_5(\downarrow)$	$\text{minASD}_5(\uparrow)$	$\text{minFSD}_5(\uparrow)$
Ours	**2.06**	**4.62**	3.09	8.15
Ours w.o. Diversity	5.95	14.63	0.16	0.37
Ours w.o. Likelihood	8.06	19.40	**10.16**	**24.83**
Ours w. Rec	2.16	4.89	3.29	9.00
Ours w. meanDiv	4.85	11.43	0.17	0.36
Method/Metric	$\text{mADE}_{10}(\downarrow)$	$\text{mFDE}_{10}(\downarrow)$	$\text{minASD}_{10}(\uparrow)$	$\text{minFSD}_{10}(\uparrow)$
Ours	**1.65**	**3.50**	1.98	5.91
Ours w.o. Diversity	4.97	12.52	0.07	0.15
Ours w.o. Likelihood	5.61	13.20	**4.55**	**11.02**
Ours w. Rec	1.78	3.85	2.92	5.73
Ours w. meanDiv	4.92	11.66	0.03	0.05

Table 2: **Left**: Forking Paths results. LDS-augmented CAM-NF significantly outperforms all other methods, including Multiverse and DLow-augmented CAM-NF. **Right**: LDS ablation results on NuScenes. In row 2-3, one of the LDS losses is removed, and performance significantly deteriorates. In row 4-5, one of the LDS losses is replaced by an alternative, and performance again declines sharply.

mum diversity metric compared to the mean diversity in the objective. In particular, the mean diversity does not penalize the degenerate case where most of the forecasts collapse to one trajectory, but one outlier forecast is very distant from the others. Together, these ablations all validate the key design choices in LDS. We include additional ablation studies assessing LDS's sensitivity to the pre-trained model, dependence on ϵ, and its training stability in Appendix F.8.

Finally, we have set the number of modes K based on predefined metrics for each dataset. However, this choice may not always be easy to make in practice. In general, a good strategy is to select a K large enough and then discard samples based on ascending likelihood. Because LDS exhibits mode-seeking behavior, a large K will likely ensure that the modes are included in the samples. Then, we can use likelihood as a reasonable proxy for the plausibility of each sample to guide the discarding process. In Appendix F.8, we illustrate an example of selecting a K larger than the number of modes and discuss potential pitfalls.

7. Conclusion

We have proposed Diversity Sampling for Flow (LDS), a post-hoc learning-based diverse sampling technique for pre-trained generative trajectory forecasting models. LDS leverages the likelihood under the pre-trained generative model and a robust diversity loss to learn a sampling distribution that induces diverse and plausible trajectory predictions. Though intended for normalizing flow models, LDS is also compatible with VAEs, and independent of this choice, consistently achieves the best results compared to other sampling techniques and multi-modal models on two distinct forecasting benchmarks. Beyond its simple "plug-in" improvement nature, through extensive ablation studies, we validate our method's design choices responsible for its effectiveness. Finally we introduce the transductive learning problem for trajectory forecasting, and show that LDS can be readily used to adapt to test instances on the fly and present a competitive solution to this novel problem setting.

Acknowledgements and Disclosure of Funding.

This work was supported by gift funding from GE Research and NEC Laboratories America, as well as NSF award CCF 1910769. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 5, 8, 16, 17

[2] George Awad, Asad Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzad Godil, David Joy, Andrew Delgado, Alan Smeaton, Yvette Graham, et al. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. 2018. 5

[3] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in markov random fields. In *European Conference on Computer Vision*, pages 1–16. Springer, 2012. 2

[4] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 1, 2, 4, 12

[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. 2, 5, 7

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 1

[7] Chiho Choi. Shared cross-modal trajectory prediction for autonomous driving. *arXiv preprint arXiv:2004.00202*, 2020. 2, 4

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 13

[9] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 2, 4, 5, 7

[10] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018. 2

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017. 5

[12] Mohamed Elfeki, Camille Couprie, Morgane Riviere, and Mohamed Elhoseiny. Gdpp: Learning diverse generations using determinantal point processes. In *International Conference on Machine Learning*, pages 1774–1783. PMLR, 2019. 2

[13] Angelos Filos, Panagiotis Tigas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? *arXiv preprint arXiv:2006.14911*, 2020. 2, 4, 11, 12

[14] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in neural information processing systems*, pages 2069–2077, 2014. 2

[15] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 1, 2, 5, 8, 16, 17

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 16

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 12

[18] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2375–2384, 2019. 1, 2

[19] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 290–297, 2003. 4

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 13

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[22] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012. 2

[23] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 11

[24] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 1, 2

[25] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020. 1, 2, 4, 5, 6, 8, 16, 17

[26] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2019. 5, 8, 16, 17

[27] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien

Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, pages 759–776. Springer, 2020. 2, 4

[28] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160. IEEE, 2011. 5

[29] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019. 2, 11

[30] Seong Hyeon Park, Gyubok Lee, Manoj Bhat, Jimin Seo, Minseok Kang, Jonathan Francis, Ashwin R Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse and admissible trajectory forecasting through multimodal context understanding. *arXiv preprint arXiv:2003.03212*, 2020. 1, 2, 5, 8, 17

[31] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 2, 4, 5, 7

[32] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015. 1, 2

[33] 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. 1, 2

[34] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019. 1, 2, 4, 12

[35] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018. 1, 2, 4, 11

[36] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *arXiv preprint arXiv:2001.03093*, 2020. 1, 2, 5, 14

[37] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15424–15434, 2019. 1, 2, 5

[38] Vladimir Vapnik and Vlamimir Vapnik. Statistical learning theory wiley. *New York*, 1:624, 1998. 2, 4

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 16

[40] Ye Yuan and Kris Kitani. Diverse trajectory forecasting with determinantal point processes. *arXiv preprint arXiv:1907.04967*, 2019. 2, 5, 7, 8, 17

[41] Ye Yuan and Kris Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. *arXiv preprint arXiv:2003.08386*, 2020. 2, 5, 7, 8, 13, 17

[42] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 2, 4

A. Normalizing Flow Models for Trajectory Forecasting

In this section, we review some preliminaries on normalizing flow based trajectory forecasting models. We refer readers to [29] for a comprehensive review of normalizing flows.

Normalizing flows learn a bijective mapping between a simple base distribution (e.g. Gaussian) and complex target data distribution through a series of learnable invertible functions. In this work, we denote the flow model as f_θ, where θ represents its learnable parameters. The base distribution is a multivariate Gaussian $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^{T\times 2}$, which factorizes across timesteps and (x, y) coordinates. Then, the bijective relationship between $\mathbf{Z}$ and $\mathbf{S}$ is captured by the following forward and inverse computations of f_θ:

$$\mathbf{S} = f_\theta(\mathbf{Z}; \mathbf{o}) \sim p_\theta(\mathbf{S} \mid \mathbf{o}), \quad \mathbf{Z} = f_\theta^{-1}(\mathbf{S}; \mathbf{o}) \sim P_\mathbf{Z} \tag{6}$$

We further impose the structural dependency between $\mathbf{S}$ and $\mathbf{Z}$ to be an invertible autoregressive function, τ_θ, between the stepwise relative *offset* of the trajectory and the corresponding $\mathbf{z}$ sample [35, 13]:

$$\mathbf{s}_t - \mathbf{s}_{t-1} = \tau_\theta(\mathbf{z}_t; \mathbf{z}_{<t})$$

The flow model can be trained using maximum likelihood. Because τ_θ is autoregressive ($\mathbf{z}_t$ does not depend on any $\mathbf{z}_k$ where $k > t$), its Jacobian is a lower-triangular matrix, which admits a simple log-absolute-determinant form [29]. The negative log-likelihood (NLL) objective is

$$\begin{aligned} & -\log p_\theta(\mathbf{S}; \mathbf{o}) \\ =& -\log\left(p(\mathbf{Z}) \left|\det \frac{\mathrm{d}f_\theta}{\mathrm{d}\mathbf{Z}}|_{\mathbf{Z}=f_\theta^{-1}(\mathbf{S};\mathbf{o})}\right|^{-1}\right) \\ =& -\left(\sum^{T}\sum^{D} \log p(\mathbf{Z}_{t,d}) - \sum^{T}\sum^{D} \log\left|\frac{\partial \tau_\theta}{\partial \mathbf{Z}_{t,d}}\right|\right) \end{aligned} \tag{7}$$

Once the model is trained, both sampling and *exact* inference are simple. To draw one trajectory sample $\mathbf{S}$, we sample $\mathbf{Z} \sim P_\mathbf{Z}$ and compute $\mathbf{S} = f_\theta(\mathbf{Z}; \mathbf{o})$ Additionally, the exact likelihood of *any* trajectory $\mathbf{S}$ under the model f_θ can be computed by first inverting $\mathbf{Z} = f_\theta^{-1}(\mathbf{S}; \mathbf{o})$ and then computing its transformed log probability via the change of variable formula, as in the second line of Equation 7.

B. Toy Example Details

In the example given in Figure 1(a), we simulate a vehicle either going straight or turning right at the intersection. The vehicle obeys the Bicycle dynamics [23] and uses a MPC-based controller with intermediate waypoints to guide it to its goal. In each simulation, the vehicle starts behind the bottom entrance of the intersection with a fixed initial position perturbed by white noise, and the simulation ends when the vehicle reaches its goal. We collect 100 simulations where the vehicle's final goal is the top exit of the intersection) and 900 simulations where the vehicle's final goal is the right exit of the intersection. They combine into a training dataset of 1000 trajectories. Each trajectory is of 100 simulation steps. We downsample it by a factor of 10 and use the first two steps as the history input to the flow model and the task is to forecast the next 8 steps. With the first 2 steps "burned" in, the vehicle is exactly at the bottom entrance of the intersection, creating a bi-modal dataset that proves to be difficult to model for a normalizing flow model.

After the data collection, we train an autoregressive affine flow as in Appendix F.3 using the entire dataset until the log likelihood converges; Then, we sample 100 times from the flow model using i.i.d unit Gaussian inputs to create trajectories as in Figure 1(b). We train a LDS model with $K = 2$ on top of this flow model and generate 2 samples using one Gaussian noise ϵ to generate the trajectories in Figure 1(c).

C. LDS-CVAE Objective

As shown in our experiments, LDS can also be applied to CVAE models. Doing so requires changing the NLL term in the LDS objective (Equation 3 to ELBO. Because ELBO is a lower bound of the true likelihood, optimizing for this modified objective would still achieve optimizing for the original objective. Formally, let $q_\phi(\mathbf{Z}|\mathbf{o})$ be the approximate latent posterior computed from the encoder and $p_\theta(\mathbf{S}|\mathbf{Z})$ be the likelihood computed from the decoder, we have

$$\mathrm{L}_{\text{LDS-CVAE}}(\psi) := \mathrm{ELBO}(\psi) - \lambda_d \mathrm{L_d}(\psi) \tag{8}$$

where $\mathrm{ELBO}(\psi) = \mathbb{E}_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{o})}[\log p_\theta(\mathbf{S}|\mathbf{Z})] - \mathrm{KL}(q_\phi(\mathbf{Z}|\mathbf{o})|p(\mathbf{Z}))$. The $\mathbf{Z}$s fed into CVAE are generated from $r_\psi(\epsilon; \mathbf{o})$, while the prior distribution is $p(\mathbf{Z}) \sim \mathcal{N}(0, \mathbf{I})$.

D. Transductive LDS Algorithms

Algorithm 2 LDS-TD-NN Training

Input: Flow f_θ, Context $\mathbf{o}$
Initialize LDS model r_ψ
for i=1,... **do**
  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  Compute $\mathbf{Z}_1, ..., \mathbf{Z}_K = r_\psi(\epsilon; \mathbf{o})$
  Transform $f_\theta(\mathbf{Z}_1), .., f_\theta(\mathbf{Z}_K)$
  Compute losses using Equations 4 & 5
  Update ψ w.r.t gradient of Equation 3
end for
Compute $\mathbf{Z}_1, ..., \mathbf{Z}_K = r_\psi(\epsilon; \mathbf{o})$
Output: $\mathcal{S} = f_\theta(\mathbf{Z}_1), .., f_\theta(\mathbf{Z}_K)$

Algorithm 3 LDS-TD-P Training

Input: Flow f_θ, Context $\mathbf{o}$

Randomly initialize $\mathbf{Z}_1, ..., \mathbf{Z}_K \sim \mathcal{N}(0, \mathbf{I})$
for i=1,... **do**
  Transform $f_\theta(\mathbf{Z}_1), .., f_\theta(\mathbf{Z}_K)$
  Compute losses using Equations 4 & 5
  Update $\mathbf{Z}_1, ..., \mathbf{Z}_K$ w.r.t gradient of Equation 3
end for

Output: $\mathcal{S} = f_\theta(\mathbf{Z}_1), .., f_\theta(\mathbf{Z}_K)$

E. DLow & DSF Details

Our implementations of DLow and DSF utilizes the same architecture as LDS. The main difference is the loss functions of the two methods. The DLow objective includes three terms:

$$\text{Reconstruction Loss}: E_r(\hat{\mathbf{s}}) = \min_{k \in K} \|\hat{\mathbf{s}}_k - \mathbf{s}\|^2$$

$$\text{Diversity Loss}: E_d(\hat{\mathbf{s}}) = \frac{1}{K(K-1)} \sum_{i \neq j \in K} \exp\Big(- \frac{\|\hat{\mathbf{s}}_i - \hat{\mathbf{s}}_j\|^2}{\sigma_d} \Big)$$

$$\text{KL Loss}: L_{\text{KL}}(\mathbf{z}) = \sum_{k=1}^{K} \text{KL}(p_\psi(\mathbf{z}_k|\mathbf{o}) || p(\mathbf{z}_k)) \tag{9}$$

and the whole objective is:

$$L_{\text{DLow}}(\psi) = \lambda_r E_r + \lambda_d E_d + \lambda_{\text{KL}} L_{\text{KL}}$$

We tune the hyperparameters of DLow and find the following setting to work the best: $\lambda_d = 0.5, \lambda_r = 1, \lambda_{\text{KL}} = 1$, and $\sigma_d = 1$.

DSF is a bit more involved. First, suppose that given input $\mathbf{o}$, DSF network ψ learns a set of latent samples $\mathbf{z}_1, ..., \mathbf{z}_K$, and the flow model decodes them to $\mathbf{s}_1, ..., \mathbf{s}_K$. DSF constructs a deterministic point process (DPP) kernel $\mathbf{L} = \text{Diag}(\mathbf{r}) \cdot \mathbf{S} \cdot \mathbf{Diag}(\mathbf{r})$, where $\mathbf{S}_{ij} = \exp\big(-k \cdot \|\mathbf{s}_i - \mathbf{s}_j\|^2\big)$ for some $k > 0$. Each entry in the quality vector $\mathbf{r}$ is defined as $r_i = \omega \exp\big(- \mathbf{z}_i^\top \mathbf{z}_i + R^2\big)$ if $\|\mathbf{z}_i\| \leq R$; otherwise, $r_i = \omega$. R is chosen as the x-th quantile of the chi-squared distribution with degree of freedom equal to the dimension of $\mathbf{z}_i$. Finally, DSF objective is $L_{\text{DSF}}(\psi) = -\text{trace}(\mathbf{I} - (\mathbf{L}(\psi) + \mathbf{I})^{-1})$, and stochastic gradient is back-propagated with respect to DSF parameters ψ. We choose $k = 1$ and $x = 90$ to be consistent with the original work; however, we find DSF to be sensitive to these hyperparameter choices and fail to scale to large-dimension tasks (e.g. Forking Paths).

F. NuScenes Experimental Details

F.1. Dataset Details

NuScenes [4] is a large urban autonomous driving dataset. The dataset consists of instances of vehicle trajectories coupled with their sensor readings, such as front camera images and lidar scans. The instances are further collected from 1000 distinct traffic scenes, testing forecasting models' ability to generalize. Following the official dataset split provided by the nuScenes development kit, we use 32186 instance for training, 8560 instances for validation, and report results on the 9041 instances in the test set.

F.2. Model Inputs

Model Inputs. All models we implement (AF, CVAE baseline models and MTP) accept the same set of contextual information

$$\mathbf{o} = \{\text{Lidar scans, velocity, acceleration, yaw}\}$$

of the predicting vehicle at time $t = 0$. Below we visualize an example Lidar scan and its histogram version [34] that is fed into the models.

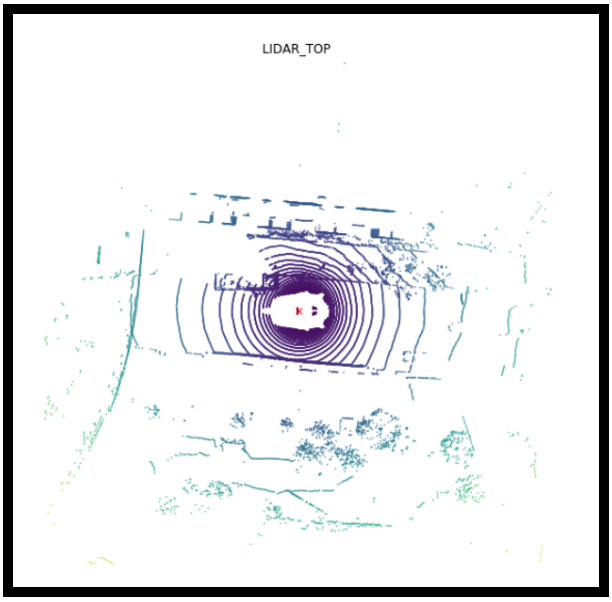

Figure 5: LiDAR inputs in nuScenes.

The Lidar scans are first processed by a pre-trained MobileNet-v128 [17] to produce visual features. These features, concatenated with the rest of the raw inputs, are passed through a neural network to produce input features for the models.

F.3. Autoregressive Affine Flow Details

Our architecture is adapted from the implementation[2] provided in [13]. Here, we describe it in high level and leave the details to the architecture table provided below.

[2] https://github.com/OATML/oatomobile/blob/alpha/oatomobile/torch/networks/sequence.py

AF consists of first a visual module that transforms the observation information $\mathbf{o}$ into a feature vector $\mathbf{h}_0$. Then, $\mathbf{h}_0$ is processed sequentially through a GRU network [8] to produce the per-step *conditioner* $\mathbf{h}_t$ of the affine transformation: $\mathbf{h}_t = \text{GRU}(\mathbf{s}_t, \mathbf{h}_{t-1})$. Finally, we train a neural network (MLP) on top of $\mathbf{h}_t$ to produce the modulators μ, σ of the affine transformation:

$$\begin{aligned}\mathbf{s}_t - \mathbf{s}_{t-1} &= \tau_\theta(\mathbf{z}_t; \mathbf{z}_{<t}) \\ &= \underbrace{\mu_\theta(\mathbf{s}_{1:t-1}, \phi)}_{\text{MLP}_1(\mathbf{h}_t)} + \underbrace{\sigma_\theta(\mathbf{s}_{1:t-1}, \phi)}_{\text{EXP}\left(\text{MLP}_2(\mathbf{h}_t)\right)} \mathbf{z}_t \end{aligned} \tag{10}$$

Table 3: AF Architecture Overview

Attributes	Values
Visual Module	MobileNet(200 × 200 × 3, 128) Linear(128+3,64) Linear(64,64) Linear(64,64)
Autoregressive Module	GRUCell(64)
MLP Module	ReLU ∘ Linear(64,32) Linear(32, 4)
Base Distribution	$\mathcal{N}(0, \mathbf{I})$

F.4. CVAE Details

Our CVAE implementation is adapted from the implementation[3] in [41]. It takes the same set of inputs as our AF model except the addition of a one-frame history input. The history is encoded using a GRU network of hidden size 64 to produce h_0, which is then concatenated with the rest of the inputs. This concatenated vector is then encoded through a 2-layer fully-connected network. To encode the future, our CVAE model uses a GRU network of the same architecture as the GRU encoder for the history. Finally, the encoded input and output (i.e. future) is concatenated and passed through another 2-layer network to give the mean and the variance of the approximate posterior distribution. For the decoder, we first sample a latent vector z using the reparameterization trick. Then, z is concatenated with the encoded inputs to condition the per-step GRU roll-out of the reconstructed future. The model is trained to maximize ELBO.

[3] https://github.com/Khrylx/DLow/blob/master/models/motion_pred.py

Table 4: CVAE Architecture Overview

Attributes	Values
History Encoder	GRU(2, 64)
Visual Module	MobileNet(200 × 200 × 3, 128)
Full Input Encoder	Linear(128+64+3, 64) Linear(64, 64) Linear(64, 64)
Full Output Encoder	GRU(2, 64)
Input Output Merger	Linear(64+64, 64) Linear(64, 32+32)
μ, σ	$\mathbb{R}^{32}, \mathbb{R}^{32}$
Decoder	GRU(2+32+64, 64) Linear(64, 64) Linear(64, 32, 2)

F.5. LDS Architecture Details

LDS r_ψ is a single multi-layer neural network with K heads, the number of modes pre-specified. To ensure stable training, we clip the diversity loss to be between $[0, 40]$ for $K = 5$ and $[0, 30]$ for $K = 10$. DSF and DLow use the same architecture as LDS.

Table 5: LDS Architecture and Hyperparameters Overview

Attributes	Values
LDS Architecture	Linear(Input Size, 64) Linear(64,32) Linear(32, 2 × T × K)
Learning Rate	0.001
λ_d	1
Diversity function clip value	40/30

F.6. Training Details

We train the "backbone" forecasting models AF, CVAE, MTP-Lidar for 20 epochs with learning rate 10^{-3} using Adam [20] optimizer and batch size 64. LDS-AF iterates through the full training set once, while LDS-AF-TD directly optimizes on the test set with a minibatch size of 64 and 400 adaptation iterations for every minibatch. For all LDS models, we set $\lambda_d = 1$ and do not experiment with further hyperparameter tuning. LDS training also uses Adam. For all models, we train 5 separate models using random seeds and report the average and standard deviations in our results.

F.7. Additional Results

Comparison with Trajectron++ To test the limit of our approach, we additionally compare against Trajectron++ [36]. Different from other baselines we include in the main paper, Trajectron++ utilizes additional inputs such as spatio-temporal graph and vehicle/pedestrian dynamics and evaluates on a shorter horizon (i.e. 8 frames). Here, we re-train an AF model by truncating all trajectories to 8 frames, and then train a LDS model on top, giving us LDS-AF-8. Trajectron++ only reports minFDE_{10} and obtains 2.24. LDS-AF-8 obtains $\text{minFDE}_{10} : 2.28, \text{minADE}_{10} : 1.09, \text{minASD}_{10} : 2.18, \text{minFSD}_{10} : 5.80$, where we include the other three metrics for completeness. As shown, LDS-AF-8 is slightly worse than Trajectron++ on minFDE_{10}; however, giving that the backbone model LDS utilizes here is a much weaker model than Trajectron++. this result is encouraging and suggests that even a weak model can produce competitive predictions when it is augmented with an effective sampling mechanism.

minASD vs. meanASD. Here, we illustrate the robustness of minASD compared to meanASD. Consider the following two sets of 10 predictions. In the first set, the pairwise distance between all pairs is exactly 1. In the second set, 9 predictions are identical, but their distance to the remaining one is 100. The first set achieves identical diversity value 1 under the two metrics, whereas the second set achieves 0 minASD but $\frac{20}{9}$ meanASD. Therefore, we might incorrectly conclude that the second set is more diverse if we were to solely rely on mean metrics for diversity.

Mean diversity results. Here, we report the meanASD and meanFSD metrics for diversity. As shown, LDS still achieves the highest diversity on these metrics and provide greater diversity boost than DLow; however, the relative differences among models are much smaller. Additionally, by examining the tables from $K = 5$ to $K = 10$, we no longer find the pattern that LDS being the only model whose diversity does not deteriorate as we did in Table 1 (Right). This is because the mean metric only captures the average behavior and not the worst case behavior. Thus, our hypothesis that the min metrics are more informative than the mean metrics are supported by the following results.

Method	Samples	meanASD (↑)	meanFSD (↑)
MTP-Lidar-5	5	5.74 ± 0.79	13.80 ± 2.00
CVAE	5	5.38 ± 0.09	12.28 ± 0.16
DLow-CVAE	5	6.66 ± 0.21	15.43 ± 0.44
AF	5	6.21 ± 0.02	14.48 ± 0.04
DLow-AF-5	5	**7.41** ± 0.29	**17.90** ± 0.67
LDS-AF-5	5	**7.89** ± 0.29	**19.06** ± 0.58

Method	Samples	meanASD (↑)	meanFSD (↑)
MTP-Lidar-10	10	5.24 ± 0.30	12.71 ± 0.59
CVAE	10	5.38 ± 0.10	12.28 ± 0.18
DLow-CVAE	10	6.96 ± 0.20	16.16 ± 0.45
AF	10	6.21 ± 0.01	14.48 ± 0.04
DLow-AF-10	10	**7.88** ± 0.57	**19.49** ± 1.36
LDS-AF-10	10	**7.90** ± 0.28	**19.71** ± 0.74

Table 6: NuScenes prediction mean diversity results.

F.8. Additional Ablation Results

In this section, we provide some additional ablation studies to further understand the effectiveness of LDS. First, we aim to understand: **how sensitive is LDS to the quality of the underlying flow model?** To answer this question, we train an additional AF model with half the number of epochs as the original one (**AF**$^-$), and then train LDS as before. The comparisons are shown in Table 7. With only half the training time, AF$^-$ performs considerably worse than AF, yet LDS is still able to provide a significant performance boost, achieving **33**% reduction in both minADE and minFDE. This reduction is greater than that of LDS applied to the stronger AF model (27%), suggesting the utility of LDS is greater for weaker pre-trained model and highlighting that its overall effectiveness is robust to the quality of the underlying flow.

Method	S	mADE_5	↓%	mFDE_5	↓%	minASD	↑%	minFSD	↑%
AF$^-$	5	3.49 ± 0.16	-	7.79 ± 0.41	-	1.99 ± 0.15	-	4.58 ± 0.46	-
LDS-AF$^-$	5	2.31 ± 0.19	**34**%	5.17 ± 0.39	**34**%	2.91 ± 0.05	146%	8.25 ± 0.32	180%
LDS-AF$^-$-TD	5	2.35 ± 0.16	**33**%	5.24 ± 0.33	**33**%	3.00 ± 0.16	151%	8.36 ± 0.14	183%
AF	5	2.86 ± 0.01	-	6.26 ± 0.05	-	1.58 ± 0.02	-	3.75 ± 0.04	-
LDS-AF	5	2.06 ± 0.09	28%	4.67 ± 0.25	25%	3.13 ± 0.18	98%	8.19 ± 0.26	118%
LDS-AF-TD	5	2.06 ± 0.02	28%	4.62 ± 0.07	26%	3.09 ± 0.07	95%	8.15 ± 0.17	117%

Table 7: LDS ablations on the pre-trained flow models.

LDS training stability. We track the state of minADE and minASD on the mini nuScenes validation set over the course of LDS training. On the mini version of the nuScenes dataset, we train a LDS-AF ($K = 5$) model, and for every training iteration, we compute the current LDS-AF model's minADE and minASD on the mini validation set. The mini version is a much smaller dataset with only 1000 instances, making this procedure manageable. The sequence of (ASD, ADE) pairs are traced as a trajectory on a 2D plane, where the x-axis corresponds to minASD_5 and the y-axis corresponds to minADE_5.

The entire trajectory is visualized in Figure 6. We additionally visualize both minADE and minASD individually over the course of training in the middle and the right panels. In all three plots, the initial AF model's (ASD, ADE) point is colored in red, and the final LDS-AF's (ASD,ADE) point is colored in blue. Focusing on the left panel, we

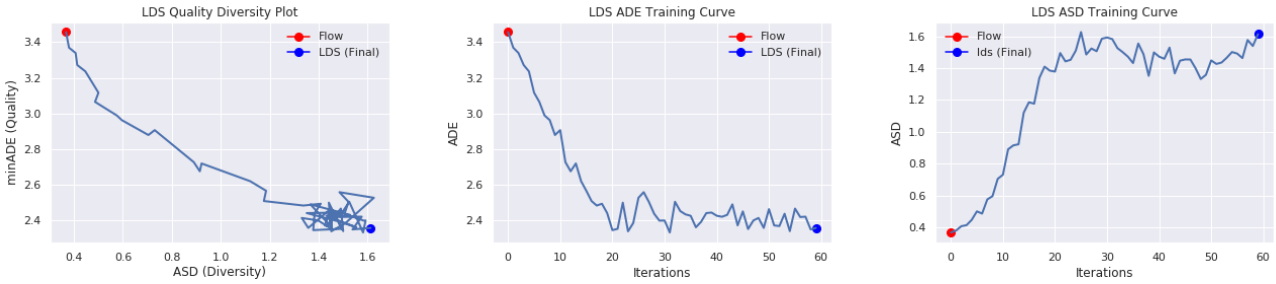

Figure 6: **Left**: LDS (ASD, ADE) plot on nuScenes mini. LDS offers stable and fast improvements to flow outputs in both accuracy and diversity during its training. **Middle**: LDS ADE over the course of training. **Right**: LDS minASD over the course of training.

note that the initial point at the top left corner represents the ADE/ASD of the pre-trained flow backbone, which has relatively high error and low diversity. However, LDS quickly discovers good sampling distribution and offers fast and near "monotonic" improvements along both axis. This provides evidence that the joint objective is effective and helps avoiding potential local minima in the loss landscape. The rapid improvement early on in training also explains why LDS's transductive adaptation procedure can be effective. Finally, an early stopping mechanism may be effective given the fast convergence to the optimum (i.e. the bottom right corner); we leave it to future work to investigate the precise stopping criterion.

Dependence on ϵ. One may eliminate the dependence on ϵ by making the mapping procedure deterministic: $\{\mathbf{Z}_1, ..., \mathbf{Z}_K\} = r_\psi(\mathbf{o})$. In theory, this simplified formulation should optimize for the same objective as the objective itself does not explicitly depend on ϵ. However, in practice, we find this ablation reduces performance as the trained LDS model may be overfitting to some deterministic set of trajectories to multiple different inputs. We report the average results over 5 seeds on LDS-AF for $K = 10$ where we do not utilize the ϵ sampling procedure (Step 3 and 4 in Algorithm 1): $\text{minADE}_{10} : 1.74, \text{minFDE}_{10} : 3.73, \text{minASD}_{10} : 2.06, \text{minFSD}_{10} : 6.03$. These results are slightly worse than the original LDS-AF results reported in Table 1. Therefore, we confirm that adding innate stochasticity to the training procedure with ϵ boosts performance, validating our original formulation.

Mismatched K. Here, we perform a controlled experiment analyzing LDS outputs when the number of samples K it learns to output mismatches the number of modes in the underlying distribution. To do this, we return to our toy intersection environment as in Figure 1(a). Instead of training a LDS with $K = 2$ as done in Figure 1(c), we train one with $K = 5$. This LDS's set of 5 samples are shown in Figure 7. Among the 5 trajectories, the two modes are still captured. Importantly, the major mode (turning right) is captured twice. However, the remaining two trajectories represent the average of the two modes and correspond to potentially unrealistic behaviors. Note that this set of five trajectories would achieve very low minADE errors in both single and multiple-future evaluations, since at least one trajectory in the predictions is close to both modes. However, for planning settings, this set of predictions may not be optimal. Hence, choosing K carefully is important in practice, and we leave it to future work for further investigations.

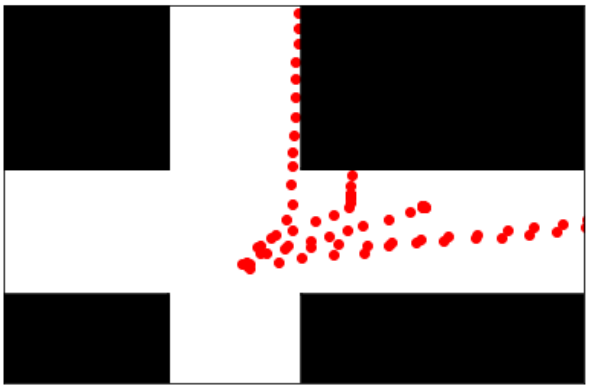

Figure 7: LDS outputs for $K = 5$.

F.9. Additional Visualizations

LDS-AF vs. AF vs. MTP-Lidar. Additional visualizations of LDS-AF, AF, and MTP-Lidar outputs (Figure 10).

LDS-AF vs. LDS-AF-TD-NN. Visualizations of LDS-AF (red) vs. LDS-AF-TD-NN (blue) are provided in Figure 8. As shown, LDS-AF-TD-NN generally adapt its predictions to the current observation better and produce more realistic trajectories with respect to the ground truth.

Varying ϵ. Finally, we visualize different sets of LDS-AF

trajectories by randomly sampling different $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. The visualizations are in 9.

G. Forking Paths Experimental Details

G.1. Dataset Details

Here, we briefly describe the Forking Paths evaluation dataset [25]. FP semi-automatically reconstruct static scenes and their dynamic elements (e.g. pedestrians) from real-world videos in ActEV/VIRAT and ETH/UCY in the CARLA simulator. To do so, it converts the ground truth trajectory annotations from the real-world videos to coordinates in the simulator using the provided homography matrices of the datasets. Then, 127 "controlled agents" (CA) are selected from the 7 reconstructed scenes. For each CA, there are on average 5.9 human annotators to control the pedestrian to the pre-defined destinations in a "natural" way that mimics real pedestrian behaviors. The scenes are also rendered to the annotators in different camera angles ranging from 'Top-Down' to '45-Degree'. The annotation can last up to 10.4 seconds, which is far longer than the 4.8 seconds prediction window in the original datasets, making the forecasting problem considerably harder. In total, there are 750 trajectories after data cleaning.

G.2. CAM-NF Model Details

Compared to the perception-based flow model we use in nuScenes, CAM-NF only uses the historical trajectory of the agent and other surrounding agents in the scene, consistent with various prior methods [15, 1] in pedestrian forecasting; we leave exploring the utility of added perception modules as in [26, 25] to future work. Hence, $\mathbf{o} = \{\mathbf{S}^a_{-T_{\text{hist}}:0}\}_{a=1}^{A}$, where A is the number of pedestrians in the scene and T_{hist} the length of history. CAM-NF first encodes the history of all pedestrians in the scene using a LSTM encoder. Then, it computes cross-pedestrian attention feature vectors using self-attention [39] to model the influences of nearby pedestrians, and uses these attention vectors as features for a normalizing flow decoder, which outputs future trajectories of the predicted pedestrian. The decoder is an autoregressive affine flow similar to the one used for nuScenes. Here, we briefly describe the architectures and refer interested readers to the original work.

The encoder first uses an LSTM of hidden dimension 512 [16] to extract a history embedding for every agent up to the most recent timestep $t = 0$:

$$h^a_t = \text{LSTM}(\mathbf{s}^a_{t-1}, h^a_{t-1}) \quad t = T_{\text{hist}} - 1, ..., 0$$

Then, the history embedding of all agents $h^0_t, ..., h^A_t$ are aggregated to compute a corresponding cross-agent attention embedding using self-attention. This embedding is then combined with the history embedding to form the inputs to the normalizing flow decoder:

$$\tilde{h}^a = h^a_0 + \text{SELF-ATTENTION}(Q^a, \mathbf{K}, \mathbf{V})$$

where (Q^a, K^a, V^a) is the query-key-value triple for each agent, and the bold versions are their all-agent aggregated counterparts. Note that $\tilde{h}^a$ is passed through a linear layer of hidden dimension 256 before passed to the decoder.

The decoder is similar to the autoregressive affine flow used for nuScenes with a few minor changes. First, $\sigma_\theta \in \mathbb{R}^{2\times 2}$ to model correlation between the two dimensions (i.e. x, y) of the states. Second, a velocity smoothing term $\alpha(\mathbf{s}_{t-1} - \mathbf{s}_{t-2})$ is added to the step-wise update. That is,

$$\begin{aligned}\mathbf{s}_t - \mathbf{s}_{t-1} &= \tau_\theta(\mathbf{z}_t; \mathbf{z}_{<t}) \\ &= \alpha(\mathbf{s}_{t-1} - \mathbf{s}_{t-2}) + \underbrace{\mu_\theta(\mathbf{s}_{1:t-1}, \phi)}_{\text{MLP}_1(\mathbf{h}_t)} + \underbrace{\sigma_\theta(\mathbf{s}_{1:t-1}, \phi)}_{\text{EXP}\left(\text{MLP}_2(\mathbf{h}_t)\right)} \mathbf{z}_t\end{aligned} \tag{11}$$

We set $\alpha = 0.5$ as in the original work. Our implementation is adapted from the original implementation[4].

G.3. LDS Architecture Details

Since trajectories in FP have different lengths, we set $T = 25$ in the LDS architecture to ensure that a bijective mapping between $\mathbf{Z} \in \mathbb{R}^{T\times D}$ exists for all samples in FP. This also means that during training, we optimize LDS for up to 25 steps. We also slightly increase the size of LDS neural network and set the maximum diversity loss to be 80, which we find to work well empirically. As before, DLow and DSF use the same architecture as LDS. The whole architecture is as follows:

Table 8: LDS Architecture and Hyperparameters Overview

Attributes	Values
LDS Architecture	Linear(Input Size, 128) Linear(128,64) Linear(64, 2 × 25 × K)
Learning Rate	0.001
λ_d	10
Diversity function clip value	80

G.4. Training Details

We train CAM-NF for 200 epochs with learning rate 10^{-3} using Adam optimizer and batch size 64. We train LDS, LDS-TD, and DLow models using mode hyperparameter $K = 20$. LDS and DLow iterate through the full training set once, while LDS-AF-TD directly optimizes on the test set with a minibatch size of 64 and 200 adaptation iterations for every minibatch. For all LDS models, through

[4] https://github.com/kami93/CMU-DATF

a hyperparameter search , we set $\lambda_d = 10$. For all models, we train 5 separate models using random seeds and report the average and standard deviations.

G.5. ActEV/VIRAT Results

Method	$\text{minADE}_1(\downarrow)$	$\text{minFDE}_1(\downarrow)$
Linear*	32.19	60.92
LSTM*	23.98	44.97
Social-LSTM*	23.10	44.27
Social-GAN*	23.10	44.27
Next*	19.78	42.43
Multiverse*	**18.51**	**35.84**
CAM-NF	19.69 ± 0.15	39.12 ± 0.31

Table 9: Training Results on ActEV/VIRAT. Our backbone flow model CAM-NF achieves comparable performance to the current state-of-art Multiverse.

G.6. Forking Paths Full Sub-Category Split Results

Method	$\text{minADE}_{20}(\downarrow)$		$\text{minFDE}_{20}(\downarrow)$	
	45-Degree	Top Down	45-Degree	Top Down
Linear*	213.2	197.6	403.2	372.9
LSTM*	201.0 ± 2.2	183.7 ± 2.1	381.5 ± 3.2	355.0 ± 3.6
Social-LSTM* [1]	197.5 ± 2.5	180.4 ± 1.0	377.0 ± 3.6	350.3 ± 2.3
Social-GAN* [15]	187.1 ± 4.7	172.7 ± 3.9	342.1 ± 10.2	326.7 ± 7.7
Next* [26]	186.6 ± 2.7	166.9 ± 2.2	360.0 ± 7.2	326.6 ± 5.0
Multiverse* [25]	168.9 ± 2.1	157.7 ± 2.5	333.8 ± 3.7	316.5 ± 3.4
CAM-NF [30]	155.2 ± 2.4	140.8 ± 2.2	305.0 ± 4.6	282.2 ± 4.9
DSF [40]	169.7 ± 1.8	155.77 ± 2.1	331.7 ± 3.7	309.5 ± 3.5
DLow [41]	144.5 ± 3.8	131.0 ± 8.1	284.6 ± 8.4	262.1 ± 20.5
LDS (Ours)	**103.8** ± 6.9	**93.4** ± 4.8	**190.6** ± 16.3	**173.4** ± 12.8
LDS-TD-NN (Ours)	**105.1** ± 4.3	**94.9** ± 2.1	**188.7** ± 10.4	**167.4** ± 4.8

Table 10: Evaluation results on Forking Paths. LDS-augmented CAM-NF significantly outperforms all other methods, including Multiverse and DLow-augmented CAM-NF.

G.7. Additional Analysis on Forking Paths

As shown in Table 2, compared to nuScenes experiments, LDS exhibits larger improvement over DLow on the FP dataset. We believe that this is because the ground truth futures in the FP test set tend to be longer than in the ActEV/VIRAT training set. Since DLow optimizes for the L_2 reconstruction loss between its forecasts and the ground-truth future trajectories in the *training* set, it is limited to improving diversity over the horizon of these training trajectories. Thus, it is unable to produce diverse predictions for longer horizon trajectories, such as those in the test set. In contrast, since LDS directly optimizes the likelihood of future trajectory according to the flow model, it does not rely on ground truth futures. Thus, it can improve diversity over much longer time horizons than in the training data. This contrast further highlights the flexibility of LDS. Finally, we note that we were not able to achieve positive results for DSF; this is likely due to the much larger latent sample dimension $(2 \times 20 = 40)$ for this dataset, which as stated in Section 2, would be an issue for DSF.

G.8. Additional Visualizations

In this section, we provide some additional visualizations of LDS, CAM-NF, and Multiverse outputs (Figure 11).

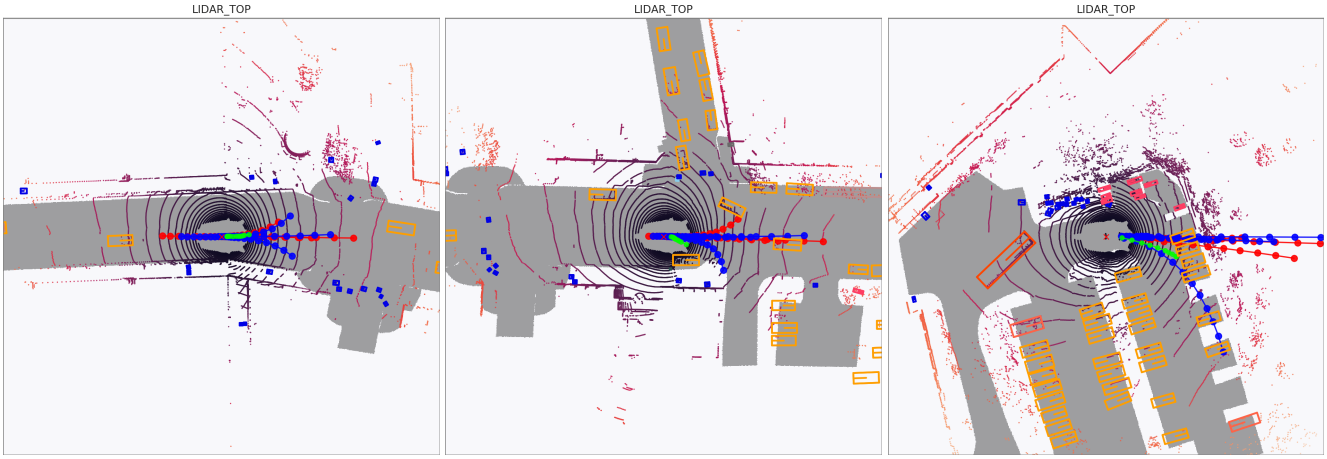

Figure 8: **LDS-AF** (red) vs. **LDS-AF-TD-NN** (blue) in NuScenes.

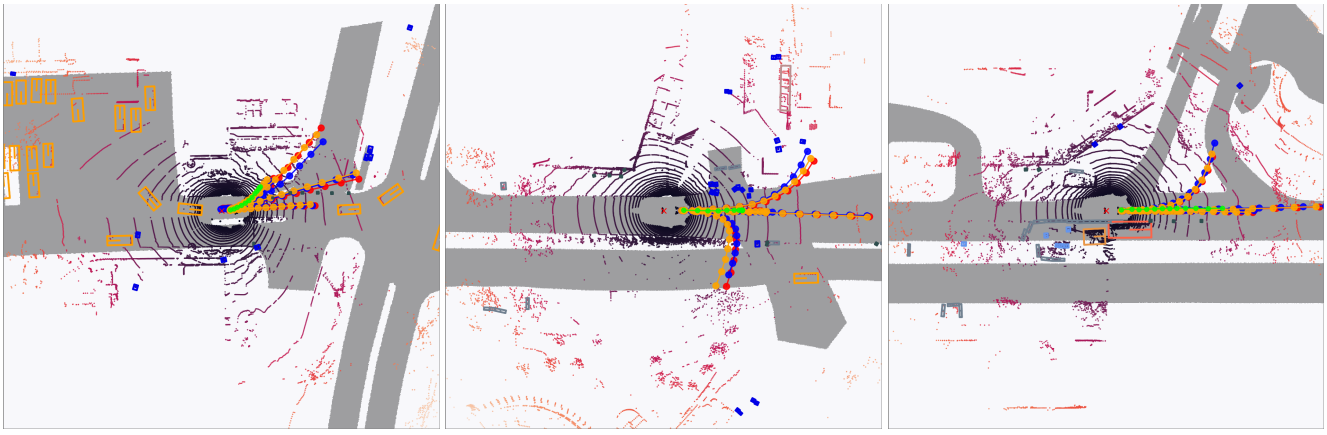

Figure 9: Effect of varying ϵ on **LDS-AF** in NuScenes. The three colors (red, blue, orange) denote three different sets of trajectories by randomly sampling ϵ.

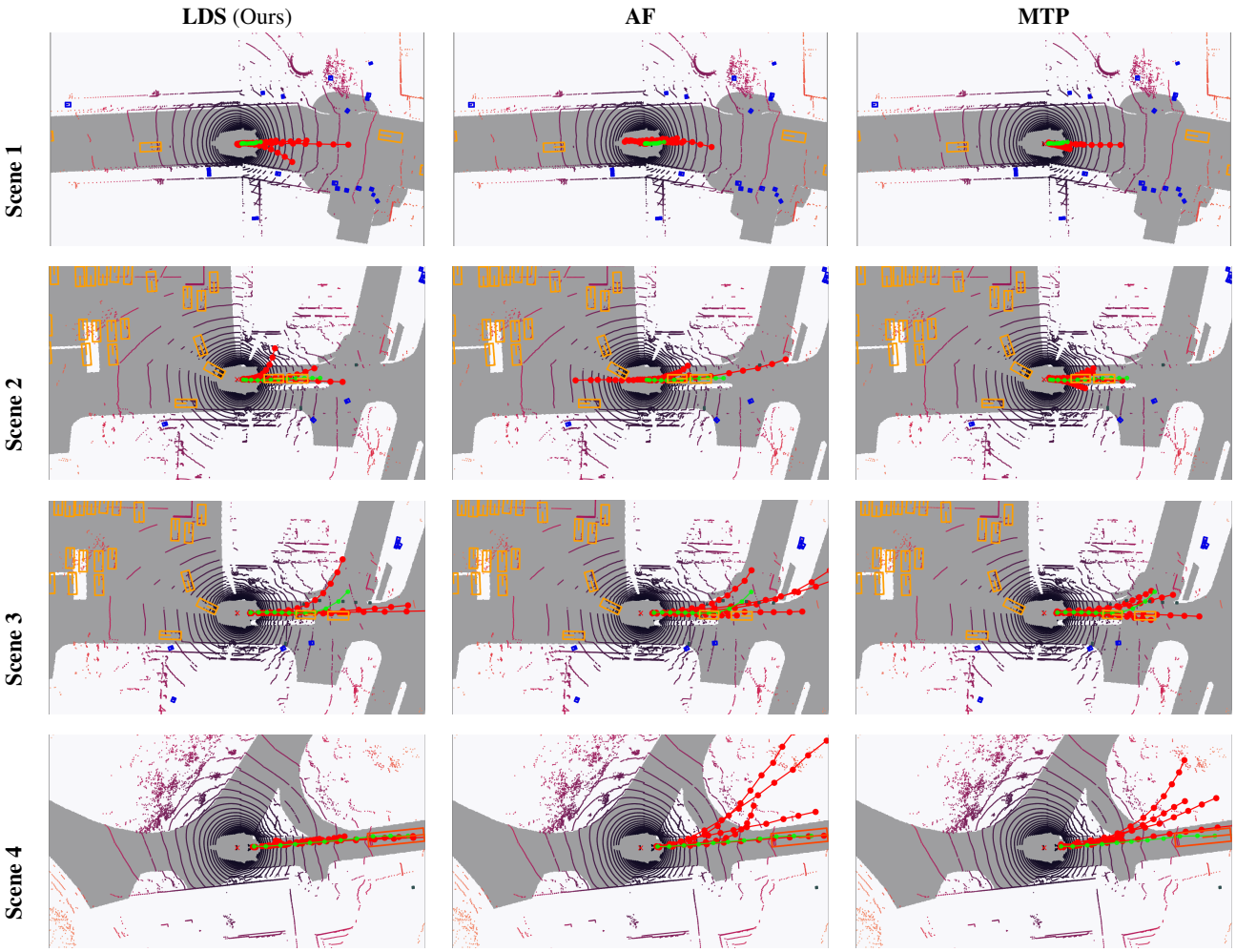

Figure 10: Additional model visualizations. The models from left to right: **LDS**, **AF**, and **MTP-Lidar**.

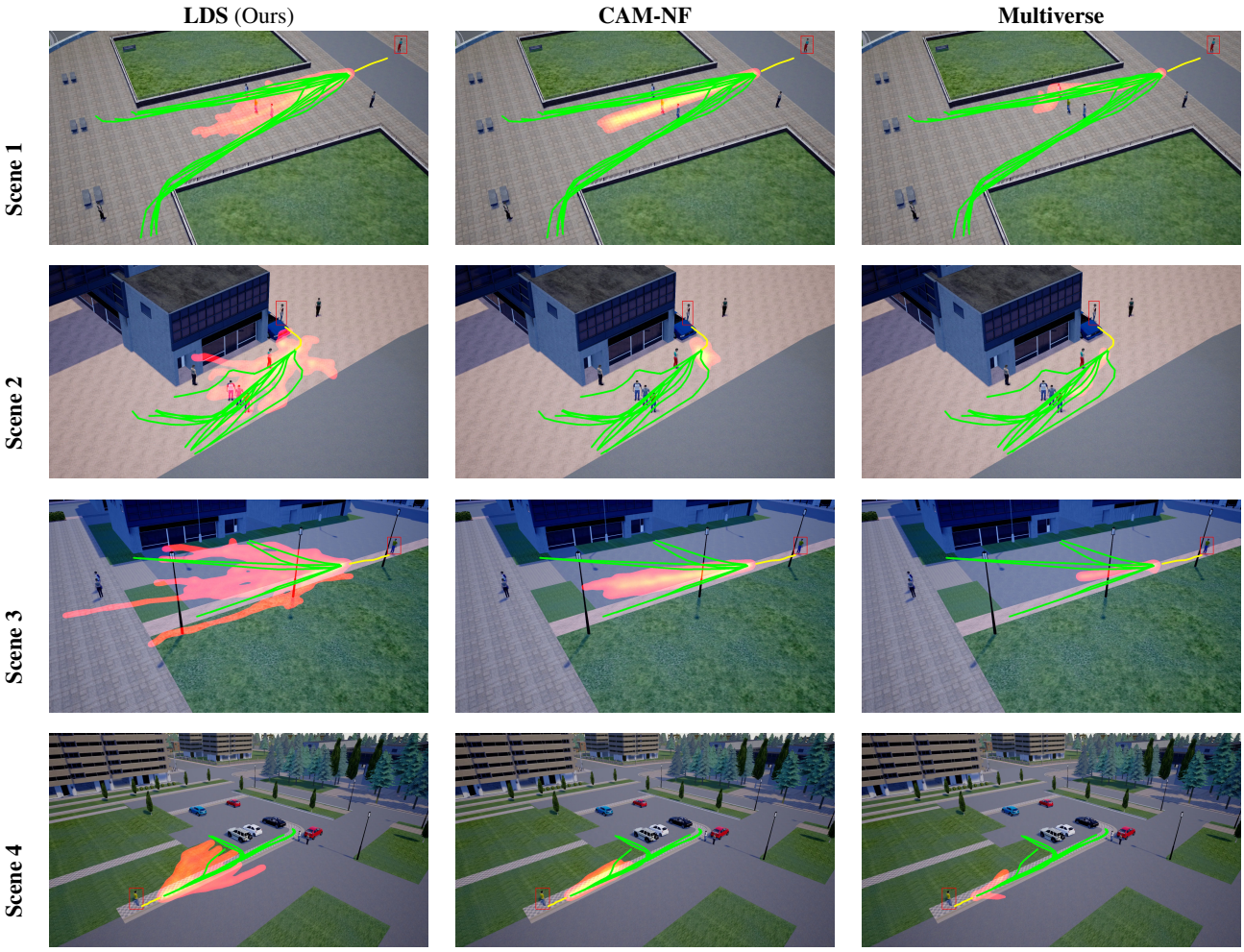

Figure 11: Additional model visualizations. The models from left to right: **LDS**, **CAM-NF**, and **Multiverse**.