Diverse Sampling for Normalizing Flow Based Trajectory Forecasting

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

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

Dinesh Jayaraman
University of Pennsylvania
dineshj@seas.upenn.edu

Osbert Bastani
University of Pennsylvania
obastani@seas.upenn.edu

Abstract

Normalizing flows have recently emerged as an attractive model for autonomous vehicle trajectory forecasting. However, a key drawback is that i.i.d. samples from flow models often do not adequately capture all the modes in the training distribution. We propose **D**iversity **S**ampling for **F**low (DSF), a post-hoc method to improve both the quality and the diversity of samples from a pre-trained flow model. To achieve these goals, DSF simultaneously draws a *set* of samples from the flow model; thus, it can spread the samples out to ensure that they cover all plausible modes of the underlying distribution. In particular, DSF trains a sampling model to balance two objectives: (i) the likelihood of the trajectories under the flow model, and (ii) a goal-based diversity objective. In our experiments, we show sampling with DSF significantly improves flow-based trajectory forecasters, performing on par with or better than the current state-of-the-art on the challenging nuScenes trajectory forecasting benchmark.

1 Introduction

Trajectory forecasting [4, 25, 21, 13, 18, 32, 27, 29] is an important task for autonomous driving. We focus on normalizing flows [24], a powerful family of generative models that have recently been applied to this task [26, 27, 28, 1]. However, due to natural biases in the real world, sampling i.i.d from a flow model's prior distribution may fail to cover all modes in the trajectory distribution. For instance, a large majority of cars might travel straight through an intersection, whereas a small fraction make a turn. As a consequence, an i.i.d. sample from the flow model will most likely travel straight; however, to ensure safety, we must also account for the minor mode corresponding to turning.

We propose a general, *post-hoc* approach, called **D**iversity **S**ampling for **F**low (DSF), for enhancing the quality and the diversity of samples from a pre-trained flow model. The key idea is that rather than drawing i.i.d. samples from the flow model, DSF learns a sampling distribution over an entire *set* of trajectories, which jointly maximizes two objectives: (i) the likelihood of the trajectories according to the flow model, and (ii) a 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. DSF is simple to implement, requiring fewer than 50 lines of code. On the challenging nuScenes trajectory forecasting benchmark [3], we show that DSF significantly improves both the accuracy and diversity of samples from flow-based trajectory forecasters, performing on par with or better than state-of-the-art methods.

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

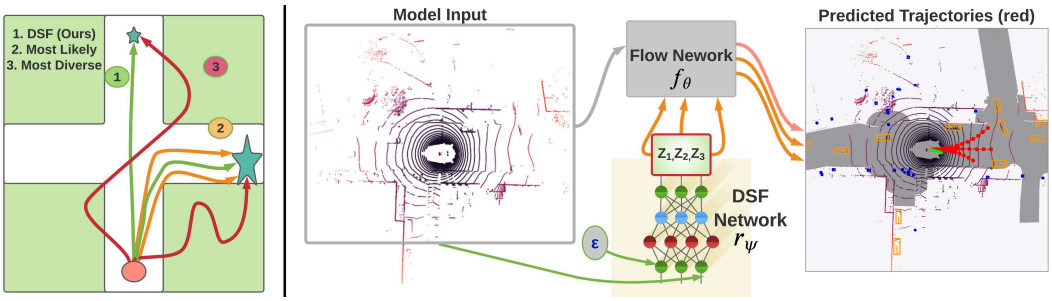

Figure 1: **Left**: **1**: DSF discovers set of paths that cover both modes and are realistic. **2**: Optimizing for only likelihood generates samples that are realistic but miss the minor mode (small star) at the top. **3**: Optimizing for only diversity generates samples that may cover both modes but are not realistic. **Right**: DSF architecture overview. DSF replaces flow's original sampling distribution (in red dotted box) with a learned distribution over set of samples in the latent space. These samples allow the pre-trained flow model to output a set of diverse and realistic trajectories.

2 Related Work

Diversity Sampling. Prior work on obtaining diverse samples from deep generative models [15, 31, 23, 11, 14, 8, 17, 7] require architectural modifications, and cannot be applied to a pre-trained model. Improving the diversity of a pre-trained model has also been explored—for example, finding the M-best solutions in a Bayesian posterior [2], or using deterministic point processes [34, 12, 9]. In contrast, our approach models complex sampling distributions through a neural network-based sampler. The closest work to ours is DLow [35], a recent post-hoc sampling method designed and demonstrated for conditional-VAEs (cVAEs) [30]. Similar to our method, DLow also jointly optimizes for the realism and the diversity of samples from the cVAE. Critically, DSF takes advantage of the ability of flow models to compute the exact likelihood of a trajectory. This enables gradient-based training to directly maximize the likelihood under the trajectory distribution rather than relying on stochastic samples from the distribution. It also permits transductive training without access to any labeled data. In our experiment, DSF outperforms DLow by significant margins.

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 $\mathbf{S} := \mathbf{S}_{1:T} \in \mathbb{R}^{T\times 2}$. At time $t = 0$, the agent has access to observation $\mathbf{o}$, which may include problem-dependent contextual features such as Lidar scans, physical attributes of the vehicle 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 normalizing 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:

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

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, 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 **D**iversity **S**ampling for **F**low (DSF), 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.

In particular, DSF 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:

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

which combines the negative log likelihood (NLL) loss from the flow model and a goal diversity loss L_{d}. Figure 1 (Left) provides intuition for these two terms, and we explain them in detail below.

Likelihood Objective. The NLL term is defined as:

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

where $\log p_\theta(\mathbf{S}_k|\mathbf{o})$ is computed as in Equation (2). The NLL loss incentivizes DSF to output a set of forecasts which all have high likelihood according to the flow model f_θ. Since f_θ is trained to maximize the likelihood of the training trajectories $\mathcal{D}$, this selects trajectories that are plausible and likely to occur. However, this set of forecasts need not be diverse and may easily concentrate around the major mode in the underlying trajectory distribution if trained without additional supervision, as in the "most likely" trajectories in Figure 1 (Left).

Diversity Objective. To combat this tendency, the diversity term L_{d} in Equation (3) incentivizes *diverse* trajectory samples that reach different parts of the state space. Specifically, we measure diversity as the *minimum* pairwise squared L_2 distance between trajectory predictions at the last time step:

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

The minimum formulation strongly incentivizes DSF to distribute its samples among different modes in the distribution, since any two predictions that are close to each other would significantly decrease L_{d}. Finally, to train r_ψ, DSF minimizes $\mathrm{L}_{\mathrm{DSF}}$ using stochastic gradient descent; it is summarized in Appendix B. In Appendix C, we also show how to extend DSF for *transductive learning*.

5 Experiments

We evaluate DSF on the nuScenes autonomous driving dataset [3]. Following prior work [6, 25, 10], 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).

Models. We train an autoregressive affine flow model (**AF**) that takes visual inputs as our underlying flow model [28] for trajectory prediction. On top of **AF**, we train two variants of DSF. The first is **DSF-AF**, the batch version in Algorithm 1. The second is **DSF-AF-TD**, the transductive version in Algorithm 2, which we train using small unlabeled minibatches from the test set. Our first baseline is **DLow-AF**, which replaces DSF with DLow [35] on top of the AF. Next, since DLow was originally designed for use with conditional VAEs (cVAEs), we include DLow-CVAE, where we pre-train a cVAE [30] and train a corresponding DLow model. Our third baseline is **MTP-Lidar**, which is based on **Multimodal Trajectory Predictions (MTP)** [6] as implemented in the nuScenes codebase, but modified to use Lidar observations to ensure fair comparison with our models. We also include existing results for state-of-the-art models as reported in [25] (rows with $*$ in Table 1).

Method	Modes	$\text{minADE}_1(\downarrow)$	$\text{minADE}_5(\downarrow)$	$\text{minADE}_{10}(\downarrow)$	$\text{minFDE}_5(\downarrow)$	$\text{minFDE}_{10}(\downarrow)$
MultiPath* [4]	64	5.05	2.32	1.96	–	–
MTP* [6]	16	4.55	3.32	3.25	–	–
CoverNet* [25]	232	4.73	2.14	1.72	–	–
MTP-Lidar	5, 10	4.68 ± 1.04	2.61 ± 0.17	1.84 ± 0.04	5.80 ± 0.49	3.72 ± 0.07
CVAE	N/A	4.20 ± 0.03	2.71 ± 0.03	2.08 ± 0.02	6.20 ± 0.05	4.58 ± 0.05
DLow-CVAE [35]	5, 10	–	2.23 ± 0.13	1.75 ± 0.03	5.00 ± 0.29	3.71 ± 0.08
AF	N/A	4.01 ± 0.05	2.86 ± 0.01	2.19 ± 0.03	6.26 ± 0.05	4.49 ± 0.07
DLow-AF	5, 10	–	**2.11** ± 0.01	1.78 ± 0.05	**4.70** ± 0.03	3.77 ± 0.13
DSF-AF (Ours)	5, 10	–	**2.06** ± 0.09	**1.66** ± 0.02	**4.67** ± 0.25	**3.58** ± 0.05
DSF-AF-TD (Ours)	5, 10	–	**2.06** ± 0.03	**1.65** ± 0.02	**4.62** ± 0.07	**3.50** ± 0.05

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

Metrics. We report minimum average displacement error $\textbf{minADE}_\text{K}$ and final displacement error $\textbf{minFDE}_\text{K}$ of K prediction samples $\hat{\mathbf{S}}_k$ compared to the ground truth trajectory. To explicitly measure prediction diversity on nuScenes, we also report the minimum average self-distance $\textbf{minASD}_\text{K}$ and minimum final self-distance $\textbf{minFSD}_\text{K}$ between pairs of predictions samples. See Appendix D for details.

Quantitative Results. In Table 1, we compare the prediction accuracy of DSF, DSF-TD, and the baselines described above. DSF and DSF-TD achieve the best overall performance. Comparing AF to MTP-Lidar, we see that although AF achieves better one-sample prediction performance (i.e., minADE_1), it performs significantly worse than MTP-Lidar when more predictions are made. This confirms our hypothesis that i.i.d. samples from a flow model do not adequately capture diverse modes, causing it to fail to cover the ground truth with good accuracy. However, when AF is augmented with DSF, it outputs the most accurate sets of predictions. Next, DSF-AF outperforms DLow-AF, especially at $K = 10$ samples, showing the advantage of exploiting the exact likelihood provided by the flow model. Finally, DSF-AF-TD outperforms DSF-AF on all metrics, suggesting that transductively training DSF produces better predictions through on-the-fly specialization. In Appendix G, we provide full results on the diversity metrics and show that DSF models achieve the most diverse of predictions.

Qualitative Results. We illustrate trajectories from DSF and baselines and show that DSF indeed outputs more diverse and plausible trajectories. See Appendix H for more qualitative results.

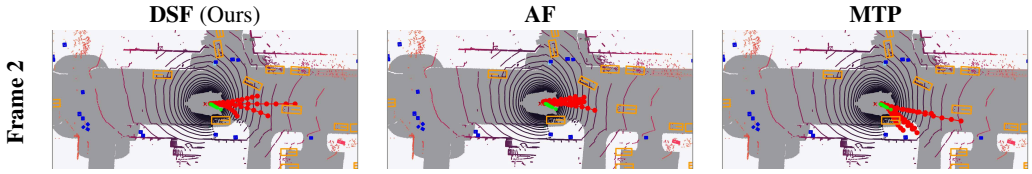

Figure 2: Model trajectory forecasts in nuScenes. $K = 5$ predicted trajectories are shown in red, and the true recorded future trajectory from the dataset is shown in green. **DSF** predicts more diverse and plausible trajectories than both baselines.

6 Conclusion

We have proposed Diversity Sampling for Flow (DSF), a learned sampling technique for pre-trained normalizing flow-based trajectory forecasting models. DSF learns a sampling distribution that induces diverse and plausible trajectory predictions. It is simple to implement, compatible with arbitrary pretrained flow-based models, and can even be trained on unlabeled data, allowing it to adapt to novel test instances on the fly. On a challenging trajectory forecasting benchmark, we demonstrate that DSF consistently achieves state-of-art stochastic trajectory prediction performance.

References

[1] Shubhankar Agarwal, Harshit Sikchi, Cole Gulino, and Eric Wilkinson. Imitative planning using conditional normalizing flow. *arXiv preprint arXiv:2007.16162*, 2020.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Uncertainty in Artificial Intelligence*, pages 1263–1273. PMLR, 2020.

[8] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7511–7522, 2019.

[9] 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.

[10] 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.

[11] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*, 2019.

[12] 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.

[13] 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.

[14] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

[15] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[16] 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.

[17] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

[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.

[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.

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

[21] 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.

[22] 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.

[23] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

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

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

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

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

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 [24] 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 [28, 10]:

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

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 [24]. 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 DSF Full Algorithm

Algorithm 1 Batch DSF Training

Input: Flow f_θ, Observation Batch $\{\mathbf{o}\}$
Initialize DSF 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 $\mathrm{L}_{\mathrm{DSF}}$ (Equation 3)
Output: Trained DSF model r_ψ

C DSF Transductive Algorithm

Note that the DSF loss function does not depend on access to any ground truth future trajectories, and relies only on the pre-trained flow model and unlabeled inputs $\mathbf{o}$. In this sense, DSF is an unsupervised algorithm. It can therefore be used *transductively*—i.e., it can adapt on the fly to a given new observation $\mathbf{o}$ through "test-time training" [33, 19]. Given an unlabeled input $\mathbf{o}$, we can train a DSF model r_ψ tailored to $\mathbf{o}$. This algorithm, which we call **DSF-TD**, is summarized in Algorithm 2. Compared to vanilla "batch" DSF, DSF-TD does not need to be trained offline with a large dataset,

making it 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 addition, in some of our experiments, we find DSF-TD outperforms DSF since it tailors its predictions to the test data.

Algorithm 2 Transductive DSF Training

Input: Flow f_θ, Context $\mathbf{o}$
Initialize DSF 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: $\mathcal{S} = f_\theta(\mathbf{Z}_1), .., f_\theta(\mathbf{Z}_K)$

D 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 $\mathbf{S}_1, ..., \mathbf{S}_J$ [32, 4, 22]:

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

These metrics are widely used in stochastic prediction tasks [32, 13] 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 one. 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:

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

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 DSF objective (Equation (5)).

E DLow Details

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

$$\begin{aligned}
&\text{Reconstruction Loss} : E_r(\hat{\mathbf{s}}) = \min_{k \in K} \|\hat{\mathbf{s}}_k - \mathbf{s}\|^2 \\
&\text{Diversity Loss} : E_d(\hat{\mathbf{s}}) = \frac{1}{K(K-1)} \sum_{i \neq j \in K} \exp\Big(- \frac{\|\hat{\mathbf{s}}_i - \hat{\mathbf{s}}_j\|^2}{\sigma_d} \Big) \\
&\text{KL Loss} : L_\text{KL}(\mathbf{z}) = \sum_{k=1}^{K} \text{KL}(p_\psi(\mathbf{z}_k|\mathbf{o}) || p(\mathbf{z}_k))
\end{aligned} \tag{8}$$

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 $\sigma_d = 1$.

F NuScenes Experimental Details

F.1 Dataset Details

NuScenes [3] 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 based models and MTP-Lidar) 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 [27] that is fed into the models.

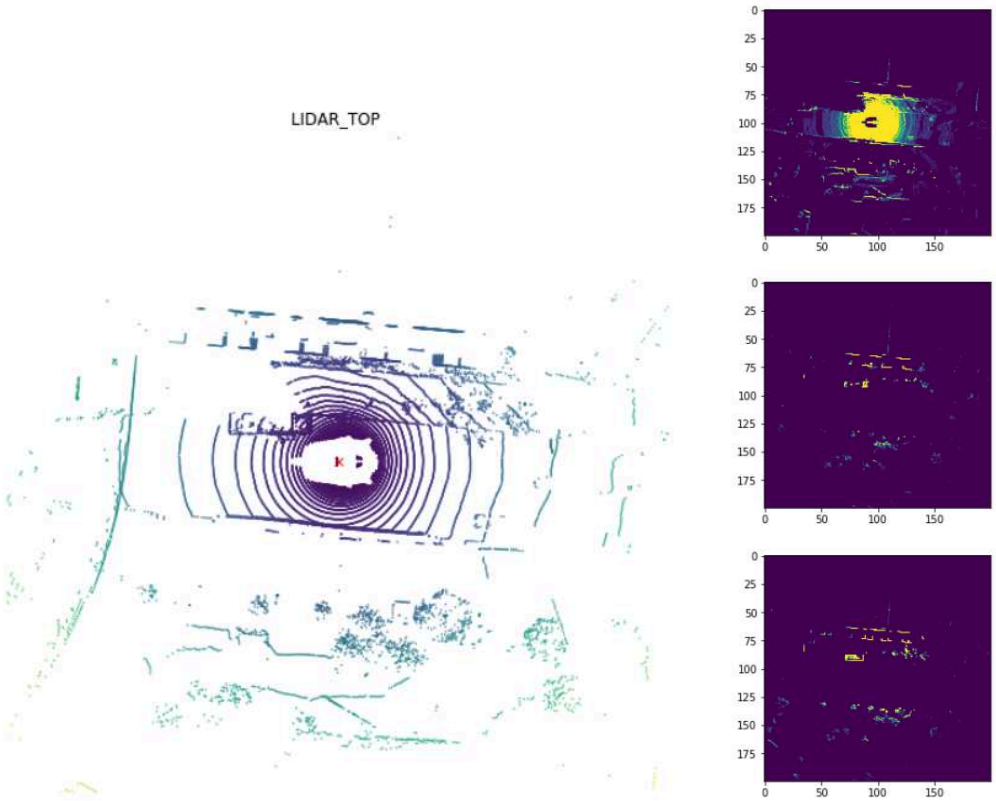

Figure 3: LiDAR inputs in nuScenes.

The Lidar scans are first processed by a pre-trained MobileNet-v128 [16] 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[1] provided in [10]. Here, we describe it in high level and leave the details to the architecture table provided below. 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 [5] 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:

$$\begin{aligned} \mathbf{s}_t - \mathbf{s}_{t-1} &= \tau_\theta(\mathbf{z}_t; \mathbf{z}_{<t}) \\ &= \underbrace{\mu_\theta(\mathbf{s}_{1:t-1}, \phi)}_{\text{MLP}_1(\mathbf{h}_t)} + \underbrace{\sigma_\theta(\mathbf{s}_{1:t-1}, \phi)}_{\text{EXP}\Big(\text{MLP}_2(\mathbf{h}_t)\Big)} \mathbf{z}_t \end{aligned} \tag{9}$$

Table 2: AF Architecture Overview

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

Table 3: CVAE Architecture Overview

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

F.4 CVAE Details

Our CVAE implementation is adapted from the implementation[2] in [35]. 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.

F.5 DSF Details

DSF 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 $K = 10$.

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

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

Table 4: DSF Architecture and Hyperparameters Overview

Attributes	Values
DSF Architecture	Linear(Input Size, 64) Linear(64,32) Linear(32, $2 \times \mathrm{T} \times \mathrm{K}$)
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. DSF-AF iterates through the full training set once, while DSF-AF-TD directly optimizes on the test set with a minibatch size of 64 and 400 adaptation iterations for every minibatch. For all DSF models, we set $\lambda_d = 1$ and do not experiment with further hyperparameter tuning. DSF 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.

G Diversity Evaluation Results

We compare the models in terms of their prediction diversity in Table 5. DSF 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 DSF 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 DSF also produces accurate predictions, these results provide strong evidence that DSF is able to simultaneously optimize accuracy and diversity.

Method	$K = 5$		$K = 10$	
	minASD_5 ($\uparrow$)	minFSD_5 ($\uparrow$)	minASD_{10} ($\uparrow$)	minFSD_{10} ($\uparrow$)
MTP-Lidar	1.74 ± 0.32	4.31 ± 1.60	0.97 ± 0.15	2.43 ± 0.34
CVAE	1.28 ± 0.03	2.99 ± 0.07	0.57 ± 0.02	1.30 ± 0.04
DLow-CVAE	2.64 ± 0.25	6.38 ± 0.65	1.18 ± 0.16	2.73 ± 0.43
AF	1.58 ± 0.02	3.75 ± 0.04	0.70 ± 0.01	1.63 ± 0.02
DLow-AF	2.56 ± 0.12	6.45 ± 0.24	1.05 ± 0.11	2.55 ± 0.28
DSF-AF (Ours)	$\mathbf{3.13} \pm 0.18$	$\mathbf{8.19} \pm 0.26$	$\mathbf{2.11} \pm 0.05$	$\mathbf{6.22} \pm 0.09$
DSF-AF-TD (Ours)	$\mathbf{3.09} \pm 0.07$	$\mathbf{8.15} \pm 0.17$	$\mathbf{1.98} \pm 0.03$	$\mathbf{5.91} \pm 0.04$

Table 5: NuScenes prediction diversity results. DSF models produce much more diverse predictions than other methods.

H Additional Qualitative Results

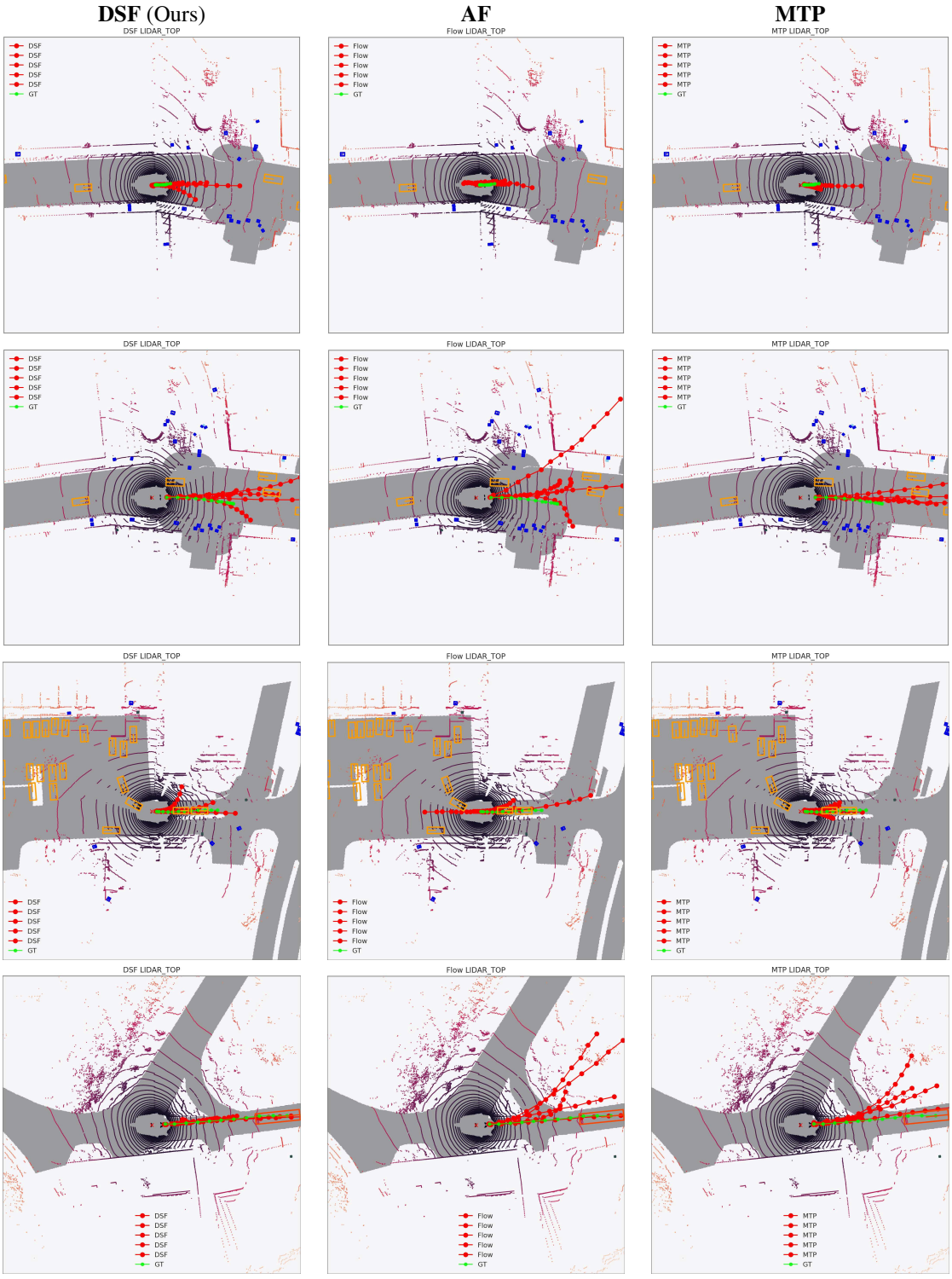

Figure 4: Additional model visualizations. The models from left to right: **DSF**, **AF**, and **MTP-Lidar**.