Propagating State Uncertainty Through Trajectory Forecasting

Boris Ivanovic\textsuperscript{1,}, Yifeng (Richard) Lin\textsuperscript{1,}, Shubham Shrivastava\textsuperscript{2,}, Punarjay Chakravarty\textsuperscript{2,}, Marco Pavone\textsuperscript{1,3}

Abstract—Uncertainty pervades through the modern robotic autonomy stack, with nearly every component (e.g., sensors, detection, classification, tracking, behavior prediction) producing continuous or discrete probabilistic distributions. Trajectory forecasting, in particular, is surrounded by uncertainty as its inputs are produced by (noisy) upstream perception and its outputs are predictions that are often probabilistic for use in downstream planning. However, most trajectory forecasting methods do not account for upstream uncertainty, instead taking only the most-likely values. As a result, perceptual uncertainties are not propagated through forecasting and predictions are frequently overconfident. To address this, we present a novel method for incorporating perceptual state uncertainty in trajectory forecasting, a key component of which is a new statistical distance-based loss function which encourages predicting uncertainties that better match upstream perception.

We evaluate our approach both in illustrative simulations and on large-scale, real-world data, demonstrating its efficacy in propagating perceptual state uncertainty through prediction and producing more calibrated predictions.

I. INTRODUCTION

Reasoning about perceptual uncertainty and its propagation through the autonomy stack is critical for the safe operation of autonomous vehicles. Failing to do so has unfortunately led to fatalities partially caused by perceptual errors propagated from vision \cite{1} and LiDAR-based \cite{2} systems. Currently, however, most trajectory forecasting approaches do not account for upstream uncertainty \cite{3}, instead taking only the most-likely value (e.g., mean) as input, neglecting measures of uncertainty (e.g., variance). As a result, perceptual uncertainties are not propagated and predictions are frequently overconfident, as shown in Fig. 1. Such overconfidence is dangerous as an autonomous vehicle may believe it has more free space than in reality, increasing the risk of collisions. A safer approach is to propagate perceptual uncertainty through forecasting systems, enabling planning components to make uncertainty-aware decisions \cite{4,5}.

Perception systems in autonomous driving are typically comprised of detectors and trackers \cite{6,7,8,9,10,11,12,13}. Broadly, detectors are responsible for identifying objects of interest from raw sensor data and trackers associate detections of the same object across different timesteps. Many trackers produce estimates of their uncertainty \cite{14}, however they have not yet been incorporated in trajectory forecasting \cite{3}. In the following, we provide an overview of existing approaches for trajectory forecasting and discuss their consideration of perceptual uncertainty.

Modular Trajectory Forecasting. Modular autonomy stacks decompose autonomous driving into distinct subproblems, typically perception, prediction, planning and control \cite{15}. This allows for each sub-task to be solved separately and combined through specified interfaces. A typical interface between perception and trajectory forecasting is to only communicate the most likely state estimate for each object detected and tracked by the perception system. Trajectory forecasting methods have thus traditionally assumed their inputs are known with certainty \cite{16}. In reality, sensors are imperfect and incorrect assumptions of certainty-equivalence in perception have been partially responsible for two separate fatalities \cite{1,2}.

To the best of our knowledge, prior forecasting work has not yet considered the explicit propagation of state uncertainty through modular systems, but there have been many developments \cite{3}. For instance, since forecasting is an inherently multimodal task (i.e., there are many possible future outcomes), several works have proposed multimodal probabilistic models, trained using exact-likelihood \cite{17,18} or variational inference \cite{19,20,21,22}. Generative Adversarial Networks (GANs) \cite{23} can generate empirical trajectory distributions by sampling multiple predictions \cite{24,25}. However, analytic distributions are often useful for gradient-based planning that minimize the likelihood of collisions \cite{15}. As a result, we focus on methods that predict analytic trajectory distributions.

End-to-End Approaches. End-to-end prediction methods operate directly from raw sensor data, performing detection, tracking, and prediction jointly. FaF \cite{26} introduced the approach of projecting LiDAR points into a bird’s eye view (BEV) grid, generating predictions by inferring detections multiple timesteps in the future. This approach was extended...
by IntentNet [27], which incorporated HD map information as an input, and predicted agent intent as well. SpAGNN [28] modeled agent interactions using a graph neural network (GNN), and ILVM [29] extended this direction further by modeling the joint distribution over future trajectories using a latent variable model. PTP [30] also uses a GNN, performing both tracking and prediction in parallel. These methods only incorporate state uncertainty implicitly, however, making it difficult to transparently analyze, probe (e.g., via counterfactual “what-if” analyses), and understand the effects of perceptual uncertainty on the rest of the autonomy stack.

**Uncertainty Propagation in Learning.** Existing approaches for uncertainty propagation in machine learning typically view inputs as noisy samples of an underlying data distribution, applying Bayesian neural networks [31], [32] and Markov models [33] to estimate the true input distribution and propagate its uncertainty to the output. Our work differs in that it does not need to perform estimation; upstream detectors and trackers typically already characterize their output confidence, e.g., as a Gaussian distribution over states, providing it for use in downstream modules.

Recently, there have been significant efforts on estimating uncertainty in deep learning [34], [35], [36], especially so in the context of planning [37], [38]. However, these works mainly focus on output uncertainty estimation rather than input uncertainty propagation. Our work tackles the latter.

With such a plethora of available trajectory forecasting approaches, an immediate approach for incorporating state uncertainty might be to augment a model’s input with uncertainty information (e.g., concatenating the input’s variance). However, as we will show, such an approach is not sufficient.

**Contributions.** Our key contributions are threefold: First, we show that there are structural pitfalls in the standard training pipeline of generative trajectory forecasting methods that hinder the simple incorporation and propagation of state uncertainty (e.g., just adding uncertainty as an input). Based on this insight, we propose a new training loss that incorporates uncertainty in trajectory forecasting with minimal effects on prediction accuracy (sometimes even improving it). Finally, we ground our theoretical hypothesis with extensive experiments on illustrative scenarios as well as real-world data.

II. PROBLEM FORMULATION

In this work, we aim to generate future trajectory distributions for a time-varying number \(N(t)\) of diverse interacting agents \(A_1, \ldots, A_{N(t)}\), each of which has a semantic type \(C_i\) (e.g., Car, Bicycle, Pedestrian). At time \(t\), agent \(A_i\)'s dynamic state \(s_i(t) \in \mathbb{R}^D\) (e.g., position, velocity, orientation) is noisily estimated by an upstream perception system as \(\hat{s}_i(t) \in \mathbb{R}^D\). We assume that the perception system also produces Gaussian state uncertainty information (i.e., state uncertainty covariance \(\Sigma_i(t) \in \mathbb{R}^{D \times D}\)), with the knowledge that many multi-object trackers are based on recursive Bayesian filters [14], which produce such information.

At time \(t\), given the estimated state \(\hat{s}_i(t)\), associated uncertainty \(\Sigma_i(t)\), and their histories \(\hat{x}_i(t) = \hat{s}_i(t-H:t) \in \mathbb{R}^{(H+1) \times N(t) \times D}\) and \(\Sigma_i(t) = \Sigma_i(t-H:t) \in \mathbb{R}^{(H+1) \times N(t) \times D \times D}\) for the previous \(H\) timesteps for each agent, our goal is to produce an accurate and calibrated distribution over all agents’ future states for the next \(T\) timesteps, \(y(t) = s(t+1:T) \in \mathbb{R}^{T \times N(t) \times D}\), which we denote as \(p(y(t) | \hat{x}(t), \Sigma(t))\). Note that the output distribution is conditioned on \(\Sigma(t)\), differing from prior work. We drop the time superscripts in the rest of the paper for brevity.

III. PROPAGATING STATE UNCERTAINTY THROUGH TRAJECTORY FORECASTING

Our method\(^1\) for obtaining the desired \(p(y \mid \hat{x}, \Sigma)\) distribution builds upon Trajectron++ [22], a state-of-the-art multi-agent trajectory forecasting method with publicly-available code. At a high-level, our model (visualized in Fig. 2) is a graph-structured recurrent encoder-decoder which makes use of a discrete latent space to explicitly model multimodality. In this section, we describe the core components of the algorithm and highlight our contributions that enable the incorporation and propagation of state uncertainty.

**Input Representation.** We first abstract the scene as a directed spatiotemporal graph \(G = (V, E)\), where nodes represent agents and edges represent their directed interactions (allowing for asymmetric influence). As in prior work [39], [22], we use the \(\ell_2\) distance as a proxy for agent interaction: an edge connects \(A_i\) and \(A_j\) if \(\|\hat{p}_i - \hat{p}_j\|_2 \leq d(C_i, C_j)\) where \(\hat{p}_i, \hat{p}_j \in \mathbb{R}^2\) are the 2D positions of agents \(A_i, A_j\), respectively, and \(d(C_i, C_j)\) is the inter-class distance threshold between agent types \(C_i\) and \(C_j\) (the order of classes in \(d(\cdot, \cdot)\) matters, since we model interactions asymmetrically).

**Encoder.** The encoder models agent history, inter-agent interaction, and any provided scene context (e.g., maps), although they are not used in this work to ensure fair comparisons to prior work in Section IV). In particular, an agent’s observed trajectory history (a concatenation of estimated states and their uncertainty) is fed into a Long Short-Term Memory (LSTM) network [40] with 32 hidden dimensions. To model neighboring agents’ influence on the modeled agent, we follow [22] and aggregate edge features from neighboring agents with an element-wise sum. These aggregated states are then fed into an LSTM with 8 hidden dimensions, yielding a single vector representing the influence that neighboring nodes have on the modeled agent. The node history and edge influence encodings are then concatenated to produce the node representation vector, \(e_x\).

**Multimodal Latent Variable.** Our model leverages the Conditional Variational Autoencoder (CVAE) latent variable framework [41] to explicitly account for multimodality in future human behavior (i.e., the possibility for many distinct outcomes). It produces the target \(p(y \mid \hat{x}, \Sigma)\) distribution by introducing a discrete Categorical latent variable \(z \in \mathbb{Z}\) which encodes high-level latent behavior and allows for the prior distribution \(p(y \mid \hat{x}, \Sigma)\) to be expressed as:

\[
p(y \mid \hat{x}, \Sigma) = \sum_{z \in \mathbb{Z}} p(z \mid \hat{x}, \Sigma)p_{\theta}(y \mid \hat{x}, z, \Sigma),
\]

where \(|\mathbb{Z}| = 25\) and \(\psi, \theta\) are network weights. We chose \(|\mathbb{Z}|\) as such because it allows for the modeling of a wide variety of high-level latent behaviors and any unused latent classes will be ignored by the CVAE [42].

\(^1\)Code online at [https://github.com/StanfordASL/PSU-TF](https://github.com/StanfordASL/PSU-TF).
Decoder. The latent variable $z$ and encoder output $e_x$ are then passed into the decoder, a 128-dimensional Gated Recurrent Unit (GRU) [43]. Each GRU cell outputs the parameters of a multivariate Normal distribution over controls $\hat{u}^{(t)}$ (e.g., acceleration and steering rate). The agent’s linearized system dynamics are then integrated with the forecasted controls $\hat{u}^{(t)}$ to obtain trajectories in position space [22]. Importantly, the only source of uncertainty in the agent’s dynamics is the decoder’s output. Thus, the decoder’s output uncertainty can be directly propagated to position space [44], [45]. Predicting agent controls and integrating them through dynamics has been shown to improve forecasting accuracy and ensures that the predictions are dynamically feasible [22].

Specifying the Loss Function. We base our model’s loss function on the discrete InfoVAE [46] objective used for Trajectron++ [22]. To ensure that the model makes use of the encoded state uncertainty, we introduce an additional term that penalizes the statistical distance between the predicted distribution and the tracked GT data. The specific choice of statistical distance $D_{SD}$ will be discussed in Section III. Formally, we train the neural network weights $\phi, \theta, \psi$ to maximize the following for each agent $A_i$,

$$E_{z \sim q_\psi(x_i, y_i)} \left[ \log p_\phi(y_i \mid \tilde{x}_i, z, \tilde{\Sigma}_i) \right] - D_{SD}(p_\psi(y_i \mid \hat{x}_i, z, \hat{\Sigma}_i) \parallel N(y_i, \Sigma_i)) \right]$$ (2)

$$-\beta D_{KL}(q_\psi(z \mid \hat{x}_i, y_i) \parallel p_\psi(z \mid \hat{x}_i, \Sigma_i)) + I_q(\hat{x}_i; z),$$

where $I_q$ is the mutual information between $\hat{x}_i$ and $z$ under the distribution $q_\psi(\hat{x}_i, z)$ and $D_{SD}$ is a measure of the distance between two probability distributions. During training, a bi-directional LSTM with 32 units encodes a node’s GT future trajectory, producing $q_\psi(z \mid x, y)$ [41].

To summarize, our method differs from Trajectron++ [22] by its introduction of state uncertainty information as an input (in red) and the addition of a statistical distance loss term (in blue). In the remainder of this section, we describe why both of these components are necessary.

Introducing state uncertainty information as an input is important as it conditions our model’s predictions on the existing level of uncertainty. However, only adding state uncertainty information does not yield changes in the model’s output uncertainty because the log-probability term in Eq. (2) encourages overconfidence (shown in Section IV). In particular, it is maximized when the output distribution lies directly on top of the GT future position with an infinitesimal uncertainty (a Dirac delta distribution). This trend towards overconfidence is a general downside of training by maximizing log-probability. Adding a statistical distance term balances the loss function by encouraging the predicted distribution to be closer to the GT tracked distribution, and thus also more calibrated (shown in Section IV) since statistical distance is minimized when the predicted and GT distributions are the same.

Choice of Statistical Distance. There are many options for the statistical distance $D_{SD}$ in Eq. (2). To be practical, $D_{SD}$ must be: (1) a measure of distance between distributions, (2) differentiable over the space of distributions, and (3) efficient to compute. The first consideration is required as $D_{SD}$ computes the distance between distributions in Eq. (2) and the second is necessary from a learning perspective. Finally, efficiency is necessary as $D_{SD}$ will be computed many times during training. While not a core desideratum, we also found that a symmetric $D_{SD}$ is desirable because over- and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

To choose $D_{SD}$, we implemented common statistical distances that satisfy the above desiderata and have closed-form expressions when evaluated between Gaussians, namely the Symmetric Kullback-Leibler (SKL), Hellinger (He), and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

To choose $D_{SD}$, we implemented common statistical distances that satisfy the above desiderata and have closed-form expressions when evaluated between Gaussians, namely the Symmetric Kullback-Leibler (SKL), Hellinger (He), and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.

In particular, the Symmetric Kullback-Leibler (SKL) and Bhattacharyya (Bh) distances. To understand each measure’s behavior, we computed distances between manually-specified mixtures and underestimates are then viewed as equally inaccurate. Asymmetrical evaluation has been studied in prediction [47], [48], and can also be included in this work via asymmetric statistical distance measures. However, we leave the decision to include asymmetry to practitioners.
IV. EXPERIMENTS AND ANALYSES

Our method is evaluated on an illustrative charged particle system and three publicly-available datasets: The ETH [51], UCY [52], and nuScenes [53] datasets. We also implement a detection-tracking perception system from state-of-the-art components and show the performance of our approach in a practical perception-prediction stack, using raw nuScenes data for sensor observations. The charged particle system serves as a controlled experiment and demonstrates that our approach indeed takes state uncertainty into account when generating predictions. The ETH, UCY, and nuScenes datasets evaluate our approach’s ability to model real-world pedestrians and vehicles.

Our model was implemented with PyTorch on a desktop computer running Ubuntu 18.04 containing an AMD Ryzen 1800X CPU and two NVIDIA GTX 1080 Ti GPUs. We trained the model for 100 epochs on the particle and pedestrian datasets and 20 epochs on the nuScenes dataset.

**Baselines.** We compare our work to Trajecton++ [22] (“T++”), whose loss function only maximizes the log-probability of the GT under the predicted distribution. We also compare against a model that only minimizes the statistical distance function $D_{SD}$ (“$D_{SD}$ Only”). Other domain-specific baselines are introduced in their respective sections.

**Methodology.** For the three real-world datasets, GT state uncertainty is obtained by running an Extended Kalman Filter (EKF) on vehicles (modeled as bicycles [54]), and a Kalman Filter (KF) on pedestrians (modeled as single integrators). All filter covariances are initialized as identity. Since the state uncertainties are estimated and not provided by the datasets, we do not use them in our evaluation. Instead, we evaluate our model with the following metrics:

1. Negative Log Likelihood (NLL): Mean NLL of the GT trajectory under a distribution generated by the model.
2. Final Displacement Error (FDE): $\ell_2$ distance between the predicted mean final position and the GT final position at the prediction horizon $T$.
3. Delta Empirical Sigma Value ($\Delta ESV_{i}$) [55]: The difference in the fraction of GT positions that fall within the $i$-level set (e.g., $1\sigma$, $2\sigma$, $3\sigma$) of the predicted distribution and the fraction from an ideal Gaussian.

In particular, $\Delta ESV_{i}$ is a useful metric for identifying over- or underconfidence, as $\Delta ESV_{i} := \sigma_{\text{pred},i} - \sigma_{\text{ideal},i}$ where $\sigma_{\text{pred},i}$ is the empirical fraction of GT positions that lie within the $i$-sigma level set of the prediction distribution and $\sigma_{\text{ideal},i}$ is the expected fraction from a perfectly-calibrated bivariate Gaussian, where $\sigma_{\text{ideal},1} \approx 0.39$, $\sigma_{\text{ideal},2} \approx 0.86$, and $\sigma_{\text{ideal},3} \approx 0.99$ [56]. Thus, $\Delta ESV_{1} < 0$ indicates overconfidence and $\Delta ESV_{1} > 0$ signifies underconfidence.

### A. Illustrative Charged-Particle Simulation

In this evaluation, we replace complex interactions between real-world agents with well-understood, controlled dynamics and verify that our method is able to incorporate state uncertainty and propagate such information to its output.

**Dataset.** We simulate a charged particle system with 3 agents, where the particles (agents) are modeled as double integrators and interact according to the Social Forces Model [57]. A state uncertainty $\Sigma$ is generated for each particle agent, where the variance in the $x$ and $y$ directions is sampled from a Gaussian distribution. Additional Gaussian noise is added to the variances at each timestep of the trajectory to simulate noise produced by an actual perception system. We collect train, validation, and test sets by initializing agents with random positions and velocities and simulating 250, 75, and 50 randomized scenarios, respectively, for 30s at 10Hz.

### Quantitative Results.** Table I shows that, as expected, T++ has the best NLL and FDE values across all prediction horizons. This makes sense because its loss function solely optimizes NLL and does not account for state uncertainty. As a result, T++ produces overconfident distributions (evidenced by its negative $\Delta ESV_{i}$ values).

In comparison, the $D_{SD}$ Only model has the worst NLL and FDE values, but the best (lowest) NLL standard deviation and highest $\Delta ESV_{i}$ values. This makes sense as the model focuses only on matching its output distribution to the estimated GT state uncertainty, resulting in large positive $\Delta ESV_{i}$ values as $D_{SD}$ Only produces underconfident distributions to try and cover the estimated GT distribution.

Our approach combines the best of both worlds, achieving the best or second-best results across all metrics. As mentioned in Section III, NLL is minimized when Dirac delta functions are placed on top of GT positions, therefore correctly propagating uncertainty should lead to a reduction in pure-NLL performance (and is why our method performs second-best on NLL). By the same argument, our model performs better than $D_{SD}$ Only on the $\Delta ESV_{i}$ metrics because Eq. (2) balances distributional and prediction accuracy, improving calibration overall.

### Generalization to Unseen Uncertainty Scales.** Introducing $D_{SD}$ to Eq. (2) affords the model zero-shot generalization capabilities to different state uncertainty scales. To verify this, we trained a version of our model on the particles dataset with only (manually-specified) large GT covariances, testing it on large and small covariance data. In Fig. 3, we see that, despite the model having only seen large uncertainties during training, it is able to generalize and maintain performance with small-scale uncertainty.

#### B. Pedestrian Motion Forecasting

**Dataset.** The ETH [51] and UCY [52] datasets consist of real pedestrian trajectories with rich multi-human interaction scenarios captured at 2.5 Hz ($\Delta t = 0.4s$). In total, there

| Particles | NLL (nats) |
|-----------|------------|
| $T^{++}$ [22] | $-5.70 \pm 0.96$ |
| $D_{SD}$ Only | $0.44 \pm 0.13$ |
| Ours | $-5.53 \pm 0.09$ |

| $\Delta ESV_{1}$ | FDE (m) |
|------------------|---------|
| $T^{++}$ [22] | $0.04 - 0.32 - 0.31 - 0.31$ |
| $D_{SD}$ Only | $0.01 - 0.04 - 0.09 - 0.12$ |
| Ours | $0.05 0.10 0.22 0.42$ |

**Table I**

**Bolded is best, underlined is second-best.**
are 5 sets of data, 4 unique scenes, and 1536 unique pedestrians. They are a standard benchmark in the field, containing challenging behaviors such as couples walking together, groups crossing each other, and groups forming and dispersing. We simulate an upstream perception system by running an EKF on the GT pedestrian positions and a KF on the GT pedestrian positions to obtain $\Sigma$ (initialized as $I$).

**Quantitative Results.** Table II summarizes the model’s performance for predicting vehicles. In addition to the original baselines, we also compare to existing works which predict vehicle trajectories from tracked GT detections [24], [59], [60], [61]. We can see that our model is still the best or second-best across all metrics. In this case, our method actually outperforms T++ on FDE for longer horizons. The model also consistently outperforms T++ on the $\Delta E SV_i$ metrics. These results decidedly show that our work is able to meaningfully propagate state uncertainty and still maintain high prediction accuracy.

**C. Autonomous Driving**

**Dataset.** The nuScenes dataset is a large-scale dataset for autonomous driving with 1000 scenes in Boston and Singapore [53]. Each scene is annotated at 2 Hz ($\Delta t = 0.5s$) and is 20s long, containing up to 23 semantic object classes.

To evaluate the performance of our method in a realistic setting (i.e., not using GT inputs), we implemented a perception system based on a state-of-the-art detector and tracker. In particular, we use the CenterPoint detector [62] and AB3DMOT tracker [63] to obtain agent tracks from nuScenes’ raw LIDAR data on which our method and its ablations are trained and evaluated. Note that these only affect the inputs to the models, predictions are still compared to GT future agent trajectories. Table IV summarizes the results, and shows first few timesteps. We also observe that T++ produces overconfident GMM component distributions with roughly equal positions and mixing probabilities. This suggests that T++ uses its latent variable (which produces the GMM mixing probabilities) to model uncertainty, while producing tight distributions to minimize the GT’s NLL. $D_{SD}$ Only produces much larger distributions and, as a result, its predictions are imbalanced. Our method produces tighter distributions relative to $D_{SD}$ Only, and some other latent modes are visible, visualized by the faint ellipses around the most-likely prediction (opacity is proportional to mode probability).

**Realistic Perception Stack Results.** To evaluate the performance of our method in a realistic setting (i.e., not using GT inputs), we implemented a perception system based on a state-of-the-art detector and tracker. In particular, we use the CenterPoint detector [62] and AB3DMOT tracker [63] to obtain agent tracks from nuScenes’ raw LIDAR data on which our method and its ablations are trained and evaluated. Note that these only affect the inputs to the models, predictions are still compared to GT future agent trajectories. Table IV summarizes the results, and shows
that our method significantly outperforms T++ on FDE while still performing second-best on the ΔESV\textsubscript{i} metrics, mirroring prior trends. Fig. 6 shows that the same trends from before persist in a realistic perception scenario. Namely, T++ produces overconfident predictions (plotted faintly because T++ predicts multiple low-probability modes and prediction opacity is proportional to mode probability), \(D_{SD}\) Only generates underconfident predictions which veer out of lanes or cross sidewalks, whereas our method’s predictions stay within lanes and have sensible uncertainty growth that aligns closely with agents’ GT future trajectories.

V. CONCLUSIONS

In this work, we present a method for incorporating and propagating perceptual state uncertainty through trajectory forecasting. At its core, a new statistical-distance based loss function term incentivizes our model to better utilize input uncertainty information and produce more calibrated output trajectory distributions. Experiments on both illustrative and real-world datasets show that the added term effectively addresses existing generative trajectory forecasting models’ overconfidence, sometimes even improving mean prediction accuracy.

While this work focuses on perceptual state uncertainty, there are many other sources of upstream uncertainty that can be propagated through trajectory forecasting, e.g., agent classification uncertainty and map error, each of which are interesting areas of future work.
[50] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[51] S. Pellegrini, A. Ess, K. Schindler, and L. v. Gool, “You’ll never walk alone: Modeling social behavior for multi-target tracking,” in *IEEE Int. Conf. on Computer Vision,* 2009.

[52] A. Lerner, Y. Chrysanthou, and D. Lischinski, “Crowds by example,” *Computer Graphics Forum,* vol. 26, no. 3, pp. 655–664, 2007.

[53] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishna, Y. Pan, G. Baldan, and O. Beijbom, “nuScenes: A multimodal dataset for autonomous driving,” in *IEEE Conf. on Computer Vision and Pattern Recognition,* 2020.

[54] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles,* vol. 1, no. 1, pp. 33–55, 2016.

[55] A. Postnikov, A. Gamayunov, and F. Gonzalo, “CovarianceNet: Conditional generative model for correct covariance prediction in human motion prediction,” in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems,* 2021.

[56] P. Bajorski, “Properties of the mahalanobis distance,” in *Statistics for Imaging, Optics, and Photonics,* 1st ed. John Wiley & Sons, 2011, ch. 5.7.2.

[57] D. Helbing and P. Molnár, “Social force model for pedestrian dynamics,” *Physical Review E,* vol. 51, no. 5, pp. 4282–4286, 1995.

[58] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in *Proc. IEEE Conf. on Robotics and Automation,* 2018.

[59] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, “TraPHic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions,” in *IEEE Conf. on Computer Vision and Pattern Recognition,* 2019.

[60] N. Deo and M. M. Trivedi, “Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms,” in *IEEE Intelligent Vehicles Symposium,* 2018.

[61] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, “Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms,” *IEEE Robotics and Automation Letters,* vol. 5, no. 3, pp. 4882–4890, 2020.

[62] T. Yin, X. Zhou, and P. Krahenbuhl, “Center-based 3d object detection and tracking,” in *IEEE Conf. on Computer Vision and Pattern Recognition,* 2021.

[63] X. Weng, J. Wang, D. Held, and K. Kitani, “3D multi-object tracking: A baseline and new evaluation metrics,” in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems,* 2020.