Trajectron++: Multi-Agent Generative Trajectory Forecasting With Heterogeneous Data for Control

Tim Salzmann*† Boris Ivanovic* Marco Pavone
Autonomous Systems Lab, Stanford University
{timsal, borisi, pavone}@stanford.edu

Abstract

Reasoning about human motion through an environment is an important prerequisite to safe and socially-aware robotic navigation. As a result, multi-agent behavior prediction has become a core component of modern human-robot interactive systems, such as self-driving cars. While there exist a multitude of methods for trajectory forecasting, many of them have only been evaluated with one semantic class of agents and only use prior trajectory information, ignoring a plethora of information available online to autonomous systems from common sensors. Towards this end, we present Trajectron++*, a modular, graph-structured recurrent model that forecasts the trajectories of a general number of agents with distinct semantic classes while incorporating heterogeneous data (*e.g. *semantic maps and camera images). Our model is designed to be tightly integrated with robotic planning and control frameworks; it is capable of producing predictions that are conditioned on ego-agent motion plans. We demonstrate the performance of our model on several challenging real-world trajectory forecasting datasets, outperforming a wide array of state-of-the-art deterministic and generative methods.*

1. Introduction

Predicting the future behavior of humans is a necessary part of developing safe human-interactive autonomous systems. Humans can naturally navigate through many social interaction scenarios because they have an intrinsic "theory of mind," which is the capacity to reason about other people's actions in terms of their mental states [14]. As a result, imbuing autonomous systems with this capability could enable more informed decision making and proactive actions to be taken in the presence of other intelligent agents, *e.g.* in human-robot interaction scenarios. Figure 1 illustrates a scenario where predicting the intent of other agents may inform an autonomous vehicle's path planning and decision making. Indeed, multi-agent behavior prediction has already become a core component of modern robotic systems, especially in safety-critical applications like self-driving vehicles which are currently being tested in the real world and targeting widespread deployment in the near future [42].

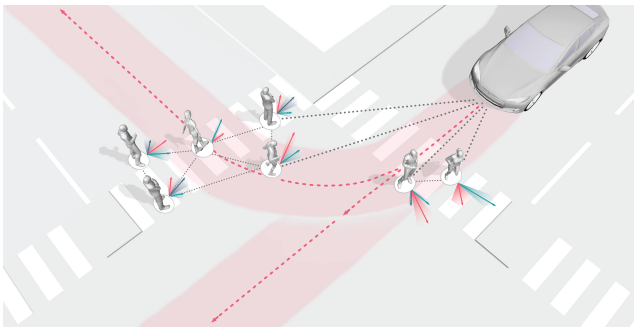

Figure 1. Exemplary road scene depicting pedestrians crossing a road in front of a vehicle which may continue straight or turn right. Our graph representation of the scene is shown on the ground, where each agent and their interactions are represented as nodes and edges, visualized as white circles and dashed black lines, respectively. Arrows depict potential future agent velocities, with colors representing different high-level future behavior modes.

There are many existing methods for multi-agent behavior prediction, ranging from deterministic regressors to generative, probabilistic models. However, many of them were developed without any considerations for real-world robotic use cases. Namely, we are interested in developing a multi-agent behavior prediction model that addresses the following desiderata: (1) Dynamically scales to model a time-varying number of agents with distinct semantic classes, such as when entities are observed from a sensor suite on an autonomous vehicle; (2) Explicitly accounts for the multimodality of human behavior, *i.e.* the potential for many high-level futures; (3) Accounts for longer-term interaction history, enabling the inference of latent behaviors; (4) Maintains a sense of interpretability; (5) Produces predictions conditioned on potential future robot trajectories, useful for intelligent planning taking into account human responses.

*Denotes equal contribution.
†Work done as a visiting student in the Autonomous Systems Lab.

(6) Provides a generally-applicable, open, and extensible approach which can accept heterogeneous data from the sensors available to the system on which it is applied.

The last two desiderata are especially important as most existing methods focus on predicting future behavior without accounting for the ego-agent's motion. Further, most approaches predict behavior from only past trajectory information, ignoring a plethora of sensor information to which modern robotic systems have access. The few methods that do include additional information are closed-source, tightly integrated with a specific robotic platform, or trained on privately-collected and unreleased data.

Our contributions are twofold: First, we present *Trajectron++*, an open and extensible approach which produces multimodal trajectory forecasts for multiple interacting agents of distinct semantic types from heterogeneous input data. *Trajectron++* is designed to be tightly integrated with downstream robotic modules such as motion planning, decision making, and control, with the ability to produce trajectories conditioned on future ego-agent plans. Second, we achieve new state-of-the-art results as *Trajectron++* outperforms an extensive selection of state-of-the-art deterministic and generative trajectory prediction methods on a variety of metrics and datasets. Additionally, we propose a new deterministic output generation scheme and evaluation metric for generative methods which use an explicit latent variable structure to capture multimodality.

2. Related Work

Deterministic Regressors. Many earlier works in human trajectory forecasting were deterministic regression models. One of the earliest, the Social Forces model [15], models humans as physical objects affected by Newtonian forces (*e.g.* with attractors at goals and repulsors at other agents). Since then, many approaches have been applied to the problem of trajectory forecasting, formulating it as a time-series regression problem and applying methods like Gaussian Process Regression (GPR) [34, 41], Inverse Reinforcement Learning (IRL) [27], and Recurrent Neural Networks (RNN) [1, 28, 40] to good effect. However, IRL relies on a unimodal assumption of interaction outcome [23, 29], making modeling multimodal data difficult. GPR falls prey to long inference times, whereas robotic use cases necessitate fast inference, *e.g.* for frequent replanning. While RNNs alone cannot model multimodality, they currently outperform all previous deterministic regression models. As a result, they are commonly found as a core component of human trajectory models [1, 21, 40].

Generative, Probabilistic Approaches. Recently, generative approaches have emerged as state-of-the-art trajectory forecasting methods due to recent advancements in deep generative models [38, 12]. Notably, they have caused a shift from focusing on predicting the single best trajectory to producing a *distribution* of potential future trajectories. This is advantageous in autonomous systems as full distribution information is more useful for downstream tasks, *e.g.* motion planning and decision making where information such as variance can be used to make safer decisions. Most works in this category use a deep recurrent backbone architecture with a latent variable model, such as a Conditional Variational Autoencoder (CVAE) [38], to explicitly encode multimodality [26, 19, 11, 37, 18, 35], or a Generative Adversarial Network (GAN) [12] to implicitly do so [13, 36, 24].

Of these, the Trajectron [18] and Social-BiGAT [24] are the best-performing CVAE-based and GAN-based models, respectively, on standard trajectory forecasting benchmarks [33, 25]. They both present strong frameworks that address many of our desiderata, however, they crucially do not account for heterogeneous input data and lack experimental validation in the presence of multiple semantic classes of agents (their experimental suites only contain pedestrians as agents).

Currently, these deterministic and generative lines of work are disparate in that models are solely designed to either produce one trajectory or a distribution of trajectories. A generally-applicable approach should be able to produce either output structure depending on the desired use case. While probabilistic models may produce these intrinsically (*e.g.* predicting the mean or mode), questions arise regarding multimodal distributions (which mode is "best") and if the mean corresponds to feasible trajectories. Further, many recent methods are GAN-based and thus unable to obtain any distributional information beyond sampling.

Accounting for Heterogeneous Data. There are few works that account for and make use of data modalities outside of prior trajectory information. This is mainly because standard trajectory forecasting benchmarks seldom include any other information, a fact that will surely change following the recent release of autonomous vehicle-based datasets with rich multi-sensor data [43, 6, 9, 22]. The few works that do incorporate additional data produce trajectory forecasts from raw point clouds, high-definition semantic maps, and their histories, all showing performance improvements over previous approaches [44, 8, 20, 7]. These approaches generally make use of end-to-end learning architectures that encode raw sensor observations with Convolutional Neural Networks (CNNs) and are trained to optimize multi-task objectives. A downside of such architectures is that they can only operate on a fixed time history (CNN weights have fixed size), whereas recurrent architectures are able to take into account all available historical information. Additionally, these approaches are closed-source and trained on private datasets, making it virtually impossible to reproduce or extend the proposed methods.

3. Problem Formulation

We aim to generate plausible trajectory distributions for a time-varying number $N(t)$ of interacting agents $A_1, ..., A_{N(t)}$. Each agent A_i has a semantic class S_i, *e.g.* Car, Bus, or Pedestrian. At time t, given the state of each agent $\mathbf{x}_i^{(t)} \in \mathbb{R}^D$, $i \in \{1, \dots, N(t)\}$, all of their histories for the previous H timesteps $\mathbf{x}_{1,\dots,N(t)}^{(t-H:t-1)} \in \mathbb{R}^{H \times N(t) \times D}$, and additional information available to each agent $I_{1,\dots,N(t)}^{(t)}$, we seek a distribution over all agents' future states for the next T timesteps $\mathbf{y} = \mathbf{y}_{1,\dots,N(t)}^{(t+1:t+T)} \in \mathbb{R}^{T \times N(t) \times D}$, which we denote as $p(\mathbf{y} \mid \mathbf{x}, I)$. Note that the number of past and future timesteps are variable, *i.e.* $H = f(A_i, S_i, t)$ and $T = g(A_i, S_i, t)$, to incorporate limited history due to sensor range and variable foresight requirements depending on the agent type S_i and scene.

Unlike previous works where only an agent's past and present position are input, we wish to use more of the heterogeneous data that modern robotic sensor suites provide. Specifically, for each agent i at time t we also assume that geometric semantic maps are available around A_i's position, $M_i^{(t)} \in \mathbb{R}^{(C/r) \times (C/r) \times L}$, with context size $C \times C$, spatial resolution r, and L semantic channels. Depending on the dataset, these maps can range in sophistication from simple obstacle occupancy grids to multiple layers of human-annotated semantic information (*e.g.* marking out sidewalks, road boundaries, and crosswalks).

One of our key desiderata is the ability to produce trajectories that take into account ego-agent motion plans, for downstream use in motion planning, decision making, and control. Thus, we also assume that we know the ego-agent's future motion plan for the next T timesteps, $\mathbf{y}_R^{(t+1:t+T)}$, a quantity that is readily available online from previous motion plans or current motion hypotheses, *e.g.* if using *Trajectron++* to evaluate a set of possible motion plans.

When these additional data modalities are available, we instead focus on producing $p(\mathbf{y} \mid \mathbf{x}, M, \mathbf{y}_R)$, meaning $I = \{M, \mathbf{y}_R\}$ from above. For brevity, we will omit the M and $\mathbf{y}_R$ symbols in the rest of this paper.

4. Trajectron++

Our model[1] is visualized in Figure 2. At a high level, we create a spatiotemporal graph representation of the scene from its topology, from which we generate a similarly-structured deep learning architecture that forecasts the evolution of node attributes, *i.e.* agent trajectories. Since our model extends the existing Trajectron [18] framework, we name it *Trajectron++*.

[1]All of our source code, trained models, and data can be found online at https://github.com/StanfordASL/Trajectron-plus-plus

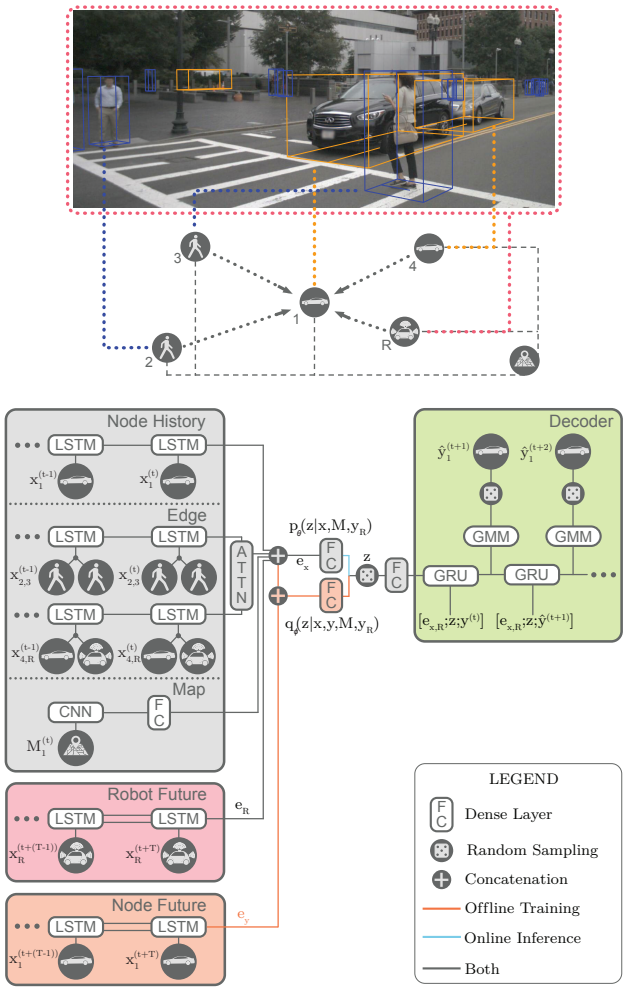

Figure 2. **Top**: Our approach represents a scene as a directed spatiotemporal graph. Nodes and edges represent agents and their interactions, respectively. **Bottom**: Our corresponding network architecture for node 1.

Scene Representation. We begin by abstracting the current scene as a spatiotemporal graph $G = (V, E)$. Nodes represent agents and edges represent their interactions. As a result, in the rest of the paper we will use the terms "node" and "agent" interchangeably. Each node also has a semantic class matching the class of its agent (*e.g.* Car, Bus, Pedestrian). The ordered pair $(A_i, A_j) \in E$ if A_i influences A_j. In this work, the ℓ_2 distance is used as a proxy for whether agents are influencing each other or not. Formally, we direct an edge from A_i to A_j if $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq d_{S_j}$ where $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^2$ are the 2D world positions of agents A_i, A_j, respectively, and d_{S_j} is a distance that encodes the perception range of agents of semantic class S_j. While more sophisticated methods can be used to construct edges (*e.g.* [40]), they usually incur extra computational overhead by requiring a complete scene graph. Figure 2 shows an example of our scene abstraction.

An important benefit of abstracting the original scene in this way is that it enables our approach to be applied to various problem domains, provided they can be modeled as spatiotemporal graphs. We specifically choose to model the scene as a directed graph, in contrast to an undirected one as in previous approaches [21, 1, 13, 40, 19, 18], because a directed graph can represent a more general set of scenes and interaction types, *e.g.* asymmetric influence. Additionally, we desire the ability to simultaneously model agents with different perception ranges, *e.g.* the driver of a car looks much farther ahead on the road than a pedestrian does while walking on the sidewalk.

Modeling Agent History. Once a graph of the scene is constructed, we need to model a node's current state, its history, and how it is influenced by its neighboring nodes. To encode the observed history of the modeled agent, we feed their current and previous states into a Long Short-Term Memory (LSTM) network [17] with 32 hidden dimensions. Since we are interested in modeling trajectories, our input $\mathbf{x}_{1,\dots,N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1)\times N(t)\times 6}$ is ultimately six-dimensional, since we use $\mathbf{x}_i^{(t)} = [\mathbf{p}_i^{(t)}, \dot{\mathbf{p}}_i^{(t)}, \ddot{\mathbf{p}}_i^{(t)}]^T$, a standard set of dynamics quantities. Such information is readily available online, either through state estimation or numerical differentiation of agent positions.

Ideally, we would specialize our states to better match their semantic class S_i. For example, we would prefer to model other vehicles on the road using a bicycle model [31]. However, estimating the bicycle model parameters of another vehicle from online observations is very difficult as it requires estimation of the vehicle's center of mass, wheelbase, and front wheel steer angle. Instead, we model all agents as point-mass single integrators.

Encoding Agent Interactions. To model neighboring agents' influence on the modeled agent, we encode graph edges in two steps. First, we aggregate edge information from neighboring agents of the same semantic class. In this work, we use an element-wise sum as our aggregation operation. We choose to combine features in this way rather than with concatenation or an average to handle a variable number of neighboring nodes with a fixed architecture while preserving count information [3, 19, 21]. We feed these aggregated states into an LSTM with 8 hidden dimensions where its weights are shared across all edge instances of the same type, *e.g.* all Pedestrian-Bus edge LSTMs share the same weights.

We then aggregate the encodings from all edge types that connect to the modeled node to obtain one "influence" representation vector, representing the effect that all neighboring nodes have. For this, we use an additive attention module [2]. The node history and edge influence encodings are then concatenated to produce a single node representation vector, $e_{\mathbf{x}}$.

Incorporating Heterogeneous Data. Modern sensor suites are able to produce much more information than tracked trajectories of other agents. Notably, high-definition semantic maps are used by many real-world systems to aid localization as well as inform navigation. To make use of this information, for each modeled agent we encode a local map, rotated to match the agent's heading, with a CNN. The map has a context size of 120×120 pixels with a 0.33 m per pixel resolution. The CNN has 3 layers each with 5×5 filters and strides of $\{2, 3, 2\}$, respectively. These are followed by a dense layer with 512 hidden dimensions, the output of which is concatenated with the node history and edge influence representation vectors.

More generally, one can include further additional information (*e.g.* raw LIDAR data, camera images, pedestrian skeleton or gaze direction estimates) in this framework by encoding it as a vector and adding it to this backbone of representation vectors, $e_{\mathbf{x}}$.

Encoding Future Ego-Agent Motion Plans. Producing predictions which take into account future ego-agent motion is an important capability for robotic decision making and control. Specifically, it allows for the evaluation of motion plans ahead of time, taking into account the response of other agents. We encode the future T timesteps of the autonomous agent's motion plan using a bi-directional LSTM with 32 hidden dimensions. We use a bi-directional LSTM due to their strong performance on other sequence summarization tasks [5]. The final hidden states are then concatenated into the backbone of representation vectors, $e_{\mathbf{x}}$.

Explicitly Accounting for Multimodality. *Trajectron++* explicitly handles multimodality by leveraging the CVAE latent variable framework. It produces the target $p(\mathbf{y} \mid \mathbf{x})$ distribution by introducing a discrete latent variable z which encodes high-level latent behavior and allows for $p(\mathbf{y} \mid \mathbf{x})$ to be expressed as [38]

$$p(\mathbf{y} \mid \mathbf{x}) = \sum_z p_\psi(\mathbf{y} \mid \mathbf{x}, z) p_\theta(z \mid \mathbf{x}) \tag{1}$$

where ψ and θ are deep neural network weights that parameterize their respective distributions. z being discrete also aids in interpretability, as we can visualize which high-level behaviors belong to each z by sampling trajectories.

During training time, we also use a bi-directional LSTM with 32 hidden dimensions to encode a node's ground truth future trajectory. The encoded output yields the recognition $q_\phi(z \mid \mathbf{x}, \mathbf{y})$ [38].

Producing Trajectories. After sampling a latent variable z, we feed it and the backbone representation vector $e_{\mathbf{x}}$ into our decoder, a 512-dimensional Gated Recurrent Unit (GRU) [10]. Each GRU cell outputs the parameters of a bivariate Gaussian Mixture Model (GMM) with 16 components, from which we sample trajectories. Specifically, the GMM produces a distribution over velocities $\dot{\mathbf{p}}_i^{(t)} \in \mathbb{R}^2$ which we numerically integrate to produce $p_\psi(\mathbf{y} \mid \mathbf{x}, z)$ from which we sample $\mathbf{y}_i^{(t)} = \mathbf{p}_i^{(t)} \in \mathbb{R}^2$.

Training the Model. We adopt the InfoVAE [45] objective function, and modify it to use discrete latent states in a conditional formulation (since we use a CVAE). For the mutual information term I_q, we approximate $q_\phi(z \mid \mathbf{x}_i, \mathbf{y}_i)$ with $p_\theta(z \mid \mathbf{x}_i)$ and obtain the unconditioned latent distribution by summing out $\mathbf{x}_i$ over the batch $\mathbf{x}_b$. Since we are also minimizing the Kullback–Leibler (KL) divergence between q and p, this is a valid approximation. Formally, we aim to solve

$$\begin{aligned}\max_{\phi,\theta,\psi} \sum_{i=1}^{N} & \mathbb{E}_{z\sim q_\phi(z|\mathbf{x}_i,\mathbf{y}_i)}\big[\log p_\psi(\mathbf{y}_i \mid \mathbf{x}_i, z)\big] \\ & - \beta D_{KL}\big(q_\phi(z \mid \mathbf{x}_i, \mathbf{y}_i) \parallel p_\theta(z \mid \mathbf{x}_i)\big) \\ & + \alpha I_q(\mathbf{x}; z)\end{aligned} \tag{2}$$

where I_q is the mutual information between $\mathbf{x}$ and z under the distribution $q_\phi(\mathbf{x}, z)$. As shown in [16], the β parameter weighting the KL penalty term is important to disentangle the latent space. A good value for this hyperparamter varies with the the size of input $\mathbf{y}$, condition $\mathbf{x}$, and latent space z. Therefore, we adjust β depending on the size of our encoder's output $e_{\mathbf{x}}$. For example, we increase the value of β when encoding map information in the condition. Additionally, we anneal β as an increasing sigmoid [4]. Thus, a low β factor is used during early training iterations so that the model learns to encode as much information in z as possible. As training continues, we gradually increase β to shift the role of information encoding from $q_\phi(z \mid \mathbf{x}, \mathbf{y})$ to $p_\theta(z \mid \mathbf{x})$. For α, we found that a constant value of 1.0 works well.

When encoding map information, we first pre-train the map-encoding CNN to predict trajectories solely based on map information. While these predictions are far from accurate, this task enables the CNN to learn the first few feature detectors in its filters. During pre-training, we try to maximize $\sum_{i=1}^{N} \log p_\psi(\mathbf{y}_i \mid M)$. We found that this compensates for the difference in training complexity between the CNN and the rest of the model. Using a randomly-initialized CNN, the model quickly converges to local minima where the map M is ignored since there is no valuable map information provided by the CNN.

5. Experiments

We evaluate our method on three publicly-available datasets: The ETH [33], UCY [25], and nuScenes [6] datasets. The ETH and UCY datasets consist of real pedestrian trajectories with rich multi-human interaction scenarios. In total, there 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. However, they only contain pedestrians ($\forall\ i,\ S_i =$ Pedestrian), so we also evaluate on the recently-released nuScenes dataset. It is a large-scale dataset for autonomous driving with 1000 scenes in Boston and Singapore. Each scene is 20s long ($dt = 0.5s$), containing up to 23 semantic object classes as well as high-quality geometric semantic maps of the environments in each scene.

We evaluate our model's performance in the following three output configurations:

1. *Full*: Our model's full output, where z and y are sampled sequentially according to
$$\begin{aligned} z &\sim p_\theta(z \mid \mathbf{x}) \\ \mathbf{y} &\sim p_\psi(\mathbf{y} \mid \mathbf{x}, z) \end{aligned} \tag{3}$$

2. z_{mode}: Predictions from our model's most-likely high-level latent behavior mode, where
$$\begin{aligned} z_{\text{mode}} &= \arg\max_z p_\theta(z \mid \mathbf{x}) \\ \mathbf{y} &\sim p_\psi(\mathbf{y} \mid \mathbf{x}, z_{\text{mode}}) \end{aligned} \tag{4}$$

3. *Mode-Mode (MM)*: Our model's deterministic and most-likely single output. The high-level latent behavior mode and output trajectory are the modes of their respective distributions,
$$\begin{aligned} z_{\text{mode}} &= \arg\max_z p_\theta(z \mid \mathbf{x}) \\ \mathbf{y} &= \arg\max_{\mathbf{y}} p_\psi(\mathbf{y} \mid \mathbf{x}, z_{\text{mode}}) \end{aligned} \tag{5}$$

We trained our model for 4000 iterations in all of the following sections, much less than other deep learning approaches because of our extensive weight sharing scheme. *Trajectron++* was implemented in PyTorch [32] and experiments were conducted on a desktop computer running Ubuntu 18.04 containing an AMD Ryzen 1800X CPU and two NVIDIA GTX 1080 Ti GPUs. Training was performed on the same computer as well as an Amazon Web Services (AWS) Elastic Cloud Compute (EC2) p2.8xlarge machine.

Evaluation Metrics. As in prior work [1, 13, 18, 36, 24], we evaluate our method for trajectory forecasting with the following four error metrics.

1. *Average Displacement Error (ADE)*: Mean ℓ_2 distance from the ground truth and our predicted trajectories.

2. *Final Displacement Error (FDE)*: ℓ_2 distance between the predicted final destination and the ground truth final destination after the prediction horizon T.

3. *Kernel Density Estimate-based Negative Log Likelihood (KDE-based NLL)*: Mean NLL of the ground truth trajectory under a distribution created by fitting a kernel density estimate on trajectory samples [18, 39].

4. *Best-of-N (BoN)*: The minimum ADE and FDE from N randomly-sampled trajectories.

Along with a variety of metrics, it is important to also compare to a wide range of previous work. Broadly, the methods we compare to can be clustered into deterministic and generative approaches.

Deterministic Baselines. We compare our method against the following deterministic baselines: (1) *Linear*: A linear regressor with parameters estimated by minimizing least square error. (2) *LSTM*: An LSTM network with only agent history information. (3) *Social LSTM* [1]: Each agent is modeled with an LSTM and nearby agent hidden states are pooled at each timestep using a proposed social pooling operation. (4) *Social Attention* [40]: Same as [1], but all other agent hidden states are incorporated via a proposed social attention operation.

Generative Baselines. On the ETH and UCY datasets, we compare our method against the following generative baselines: (1) *S-GAN* [13]: Each agent is modeled with an LSTM-GAN, which is an LSTM encoder-decoder whose outputs are the generator of a GAN. The generated trajectories are then evaluated against the ground truth trajectories with a discriminator. (2) *S-GAN-P* [13]: Same as the above, but also with their proposed global pooling scheme. (3) *SoPhie* [36]: An LSTM-GAN with the addition of a proposed physical and social attention module. (4) *Social-BiGAT* [24]: An LSTM-GAN with the addition of a Graph Attention Network (GAT) to encode agent relationships. (5) *Trajectron* [18]: An LSTM-CVAE encoder-decoder which is explicitly constructed to match the spatiotemporal structure of the scene. Its scene abstraction is similar to ours, but uses undirected edges.

On the nuScenes dataset, we also compare against the following methods: (6) *Convolutional Social Pooling (CSP)* [11]: An LSTM-based approach which explicitly considers a fixed number of movement classes and predicts which of those the modeled agent is likely to take. (7) *CAR-Net* [37]: An LSTM-based approach which encodes scene context with visual attention. (8) *SpAGNN* [7]: A CNN encodes raw LIDAR and semantic map data to produce object detections, from which a Graph Neural Network (GNN) produces probabilistic, interaction-aware trajectories.

Evaluation Methodology. For the ETH and UCY datasets, we evaluate using a leave-one-out strategy, training our model on four datasets and testing it on the held-out fifth, similar to previous works [1, 13, 18, 24, 36].

For the nuScenes dataset, there are explicit train, validation, and test splits. However, the ground truth test annotations are not public. Instead, we use the Megvii 3D object detector and tracker [46], winner of the 2019 nuScenes 3D Object Detection Challenge [30], to obtain tracking annotations on the test set. This matches how *Trajectron++* would be used in the real-world, with an object detector, classifier, and tracker providing positions $\mathbf{p}_i^{(t)}$ and semantic labels S_i for each agent.

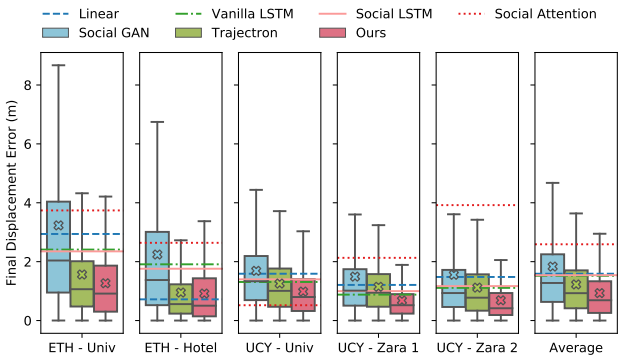

Figure 3. FDE results of all methods per dataset, as well as their average performance. Boxplots are shown for all generative models since they produce distributions of trajectories. 2000 trajectories were sampled per model at each prediction timestep, with each sample's FDE included in the boxplots. Our model's z_{mode} output is plotted here. X markers indicate the mean FDE. Mean FDEs from deterministic baselines are visualized as horizontal lines.

5.1. ETH and UCY Datasets

We first evaluate our approach on the ETH [33] and UCY [25] Pedestrian Datasets, against deterministic methods with standard trajectory forecasting metrics. To fairly compare against prior work, we do not use map encoding or any encoding of future motion plans. We only use the node history and edge encoders for our model's encoder. Additionally, we use our model's deterministic Mode-Mode output scheme, which produces the model's most likely single trajectory. Table 1 summarizes these results and shows that our model is competitive with state-of-the-art deterministic regressors, even without training on a squared-error loss (on which the ADE and FDE metrics are based).

While ADE and FDE are important metrics for deterministic, single-trajectory methods, any deeper probabilistic information available from generative methods is destroyed when taking the mean over the dataset. Instead, we focus on evaluation methods which maintain such information. The first method aims to directly compare deterministic and generative methods by still using ADE and FDE, but directly plotting the full error distributions for generative methods, as in [18]. This provides an idea as to how close and concentrated the predictions are around the ground truth. Figure 3 shows both generative and deterministic methods' FDE performance; ADE results can be found in the appendix. In both metrics, our method's error distribution is lower and more concentrated than other generative approaches, even outperforming state-of-the-art deterministic methods.

To more concretely compare generative methods, we use the KDE-based NLL metric proposed in [18, 39], an approach that maintains full output distributions and compares the log-likelihood of the ground truth under different methods' outputs. Table 2 summarizes these results and shows that our method notably outperforms others. Unfor-

Table 1. Our model's deterministic Mode-Mode output performs strongly for modeling pedestrians on displacement error metrics, even if it was not originally trained to minimize displacement error. Bold indicates the best values.

Dataset	ADE / FDE (m)				
	Linear	LSTM	Social LSTM [1]	Social Attention [40]	Ours (MM)
ETH	1.33 / 2.94	1.09 / 2.41	1.09 / 2.35	**0.39** / 3.74	0.50 / **1.19**
Hotel	0.39 / 0.72	0.86 / 1.91	0.79 / 1.76	0.29 / 2.64	**0.24 / 0.56**
Univ	0.82 / 1.59	0.61 / 1.31	0.67 / 1.40	**0.20 / 0.52**	0.36 / 0.89
Zara 1	0.62 / 1.21	0.41 / 0.88	0.47 / 1.00	0.30 / 2.13	**0.29 / 0.72**
Zara 2	0.77 / 1.48	0.52 / 1.11	0.56 / 1.17	0.33 / 3.92	**0.27 / 0.67**
Average	0.79 / 1.59	0.70 / 1.52	0.72 / 1.54	**0.30** / 2.59	0.34 / **0.84**

Table 2. Mean KDE-based NLL for each dataset. Lower is better. 2000 trajectories were sampled per model at each prediction timestep. Bold indicates the best values.

Dataset	KDE-based NLL		
	S-GAN [13]	Trajectron [18]	Ours (Full)
ETH	15.70	**2.99**	4.81
Hotel	8.10	2.26	**−0.98**
Univ	2.88	1.05	**−0.35**
Zara 1	1.36	1.86	**−1.76**
Zara 2	0.96	0.81	**−1.67**
Average	3.68	1.30	**−0.33**

Table 3. Frequency of obstacle violations in our predictions with and without map encoding (M.E.). Bold indicates the best values.

Metric	Obs. Violations (%)	
	Ours	Ours + M.E.
Overall Percentage	4.6	**1.0**
Percentage near Obstacles	21.5	**4.9**

tunately, at this time there is no publicly-released code for SoPhie [36] or Social-BiGAT [24], so we cannot compare to them on the KDE-based NLL metric. Instead, we evaluate *Trajectron++* with the Best-of-N metric used in their works. These results are summarized in the appendix, and show that our method *significantly* improves over the previous state-of-the-art [24], achieving a 55% lower average ADE and FDE.

Map Encoding. To evaluate the effect of incorporating heterogeneous data, we compare a version of our model with and without the map encoder in our architecture. Specifically, we compare the frequency of obstacle violations on the ETH - University scene, which has a provided obstacle occupancy grid. Table 3 shows the resulting reduction in the frequency of pedestrian-object collisions. We also study how much of a reduction there is for pedestrians that are especially close to an obstacle (*i.e.* they have at least one obstacle-violating trajectory in their Full output), an example of which is shown in Figure 4.

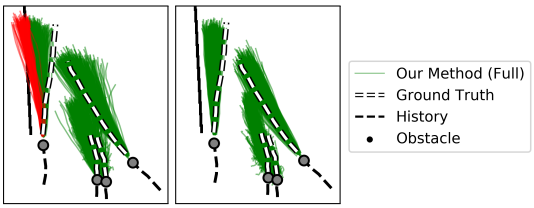

Figure 4. **Left:** When only using trajectory data, our model does not know of obstacles and makes predictions into walls (in red). **Right:** Encoding a local map of the agent's surroundings significantly reduces the frequency of obstacle-violating predictions.

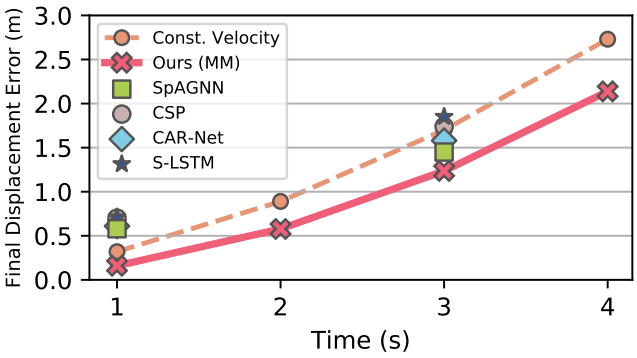

Figure 5. Vehicle-only FDE across time for our model with edge and map encoding compared to other single-trajectory and probabilistic approaches.

5.2. nuScenes Dataset

To further evaluate our model's ability to use heterogeneous data and simultaneously model multiple semantic classes of agents, we evaluate it on the nuScenes dataset [6]. Again, we use our deterministic Mode-Mode output scheme to fairly compare with other single-trajectory predictors. We forecast the trajectories of both Pedestrians and Cars, two semantic object classes which incorporate many of the 23 possible object classes present in the dataset. To obtain an estimate of our prediction quality degradation over time, we compute our model's FDE at $t = \{1, 2, 3, 4\}s$ for all tracked objects (with at least $4s$ of available future data). Figure 5 visualizes our result in comparison with state-of-the-art vehicle trajectory prediction models. Results for

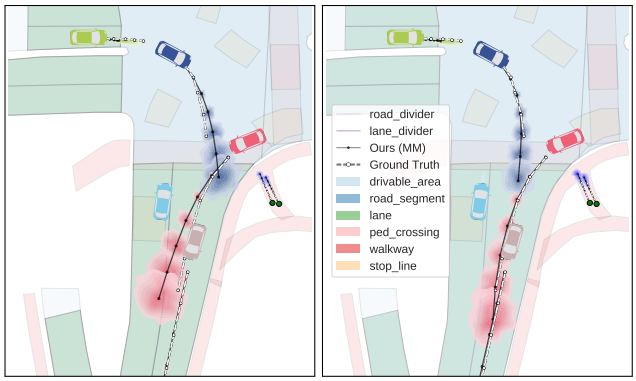

Figure 6. **Left:** Without map information, the model tends to under- or over-shoot turns. **Right:** Encoding a local map of the agent's surroundings notably increases our method's accuracy and confidence in turns. Pedestrian predictions are also visible in the middle-right of the image.

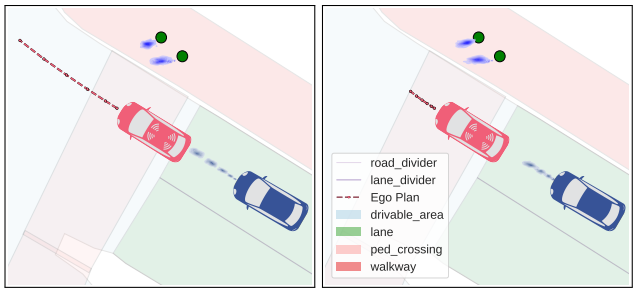

Figure 7. An autonomous ego-vehicle (in red) and following human-driven vehicle (in blue) are waiting at a traffic light. **Left:** If our ego-vehicle plans to accelerate forward, we predict that the human driver will similarly accelerate. **Right:** In contrast, if the ego-vehicle plans a much smaller forward acceleration then *Trajectron++* predicts that the human driver will similarly crawl forward to maintain a sufficient minimum distance.

pedestrians can be found in the appendix. We also implement a constant velocity baseline, which simply maintains the agent's heading and speed for the prediction horizon. Note that our model was only trained to predict $3s$ into the future, thus our performance at $4s$ also provides a measure of our model's capability to generalize beyond its training configuration. As can be seen, our model outperforms existing approaches without facing a sharp degradation in performance after $3s$.

Ablation Study. To develop an understanding of which model components influence performance, we perform a comprehensive ablation study in Table 4. We start from a base version of the model which only uses a node's trajectory history to predict its future. We then incrementally evaluate our base model with the addition of edge encoding only and map encoding only, before reporting the performance of the full *Trajectron++*. In addition to our deterministic Mode-Mode output, we report the performance of our model's Full output as it shows the entire range of predictions that our model makes. As can be seen in the first row of Table 4, even our base model's deterministic Mode-Mode output performs strongly relative to current state-of-the-art approaches for vehicle trajectory forecasting [7]. Adding edge encoding onto our base model yields improvements across all metrics. Adding map encoding to the base model, on the other hand, yields significant FDE improvements in later timesteps, along with an up to 30% reduction in the frequency of road boundary violations. The full *Trajectron++* model retains the best of both additions, with even lower FDE values at later timesteps and an up to 40% reduction in road boundary violation frequency compared to the base model. Overall, map encoding is our dominant performance-improving module, making our predictions more accurate and feasible.

Map Encoding. This is also visible qualitatively, Figure 6 shows the difference between our base model with and without map encoding. In it, we can see that the base model undershoots the turn for the red car, predicting that it will end up in the opposing lane. Further, much of its probability mass extends across lane boundaries. With map encoding, our predictions are not only more accurate, but nearly all probability mass now lies within lane boundaries.

Future-Conditional Predictions. Figure 7 shows an example of future-conditional predictions from our model. When the red ego-vehicle's plan is to accelerate forward, our model predicts that the blue car will also accelerate forward. Conversely, if the red ego-vehicle plans to slightly crawl forward, we predict the same motion for the blue car. Thus, our model is able to make use of the encoded robot future motion plan as well as relational information from its edge encoders to accordingly adjust its predictions. This enables using our model for decision making online while taking into account the potential responses of humans, *e.g.* to evaluate a set of motion plans for safety.

6. Conclusion

We have presented *Trajectron++*, a generative multi-agent trajectory forecasting approach which uniquely addresses our desiderata for a generally-applicable, open, and extensible framework. It accepts heterogeneous data beyond common dynamics quantities and can produce future-conditional predictions which maintain full probabilistic information, which is especially important for use in downstream robotic tasks. It achieves state-of-the-art prediction performance in a variety of metrics on standard and new real-world multi-agent human behavior datasets.

Future directions include incorporating the human behavior predictions from *Trajectron++* in robotic motion planning, decision making, and control frameworks, each of which are core tasks that are solved online by autonomous systems. Another key future direction is using *Trajectron++* in simulation, treating it as a human behavior

Table 4. Ablation study of our model. The vehicle-only FDE and frequency of road boundary violations are evaluated for attenuated versions of our model. Both the model's Full generative output and Mode-Mode deterministic output schemes are evaluated for each model version.

Configuration		**FDE Full (m)**				**FDE MM (m)**				**Road Boundary Viols. (%)**			
Edge	Map	@1s	@2s	@3s	@4s	@1s	@2s	@3s	@4s	@1s	@2s	@3s	@4s
		0.16	0.73	1.62	2.82	0.12	0.56	1.25	2.23	1.0	2.9	6.1	10.3
✓		0.16	0.72	1.61	2.80	0.11	0.54	1.25	2.23	1.0	2.7	5.7	9.9
	✓	0.19	0.71	1.50	2.54	0.14	0.57	1.25	2.18	1.0	2.0	4.2	7.7
✓	✓	0.20	0.71	1.46	2.45	0.16	0.58	1.24	2.14	0.9	1.8	3.6	6.1

engine and generating realistic human trajectories in simulated environments. Finally, there is still a whole host of heterogeneous data that can be incorporated into this model, *e.g*. raw LIDAR data, raw camera images, 2D/3D semantic segmentation, pedestrian gaze direction estimates, which are left as future work.

Acknowledgment. We thank Punarjay Chakravarty for his overall advising and guidance, technical insights, and many fruitful discussions, as well as Matteo Zallio for his help in visualizing our vision and results. The authors were supported by funding from the Ford-Stanford Alliance.

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 *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 2, 4, 5, 6, 7

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Int. Conf. on Learning Representations*, 2015. 4

[3] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Conf. on Neural Information Processing Systems*, 2016. 4

[4] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. Annual Meeting of the Association for Computational Linguistics*, 2015. 5

[5] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. In *Proc. of Conf. on Empirical Methods in Natural Language Processing*, pages 1442–1451, 2017. 4

[6] 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, 2019. 2, 5, 7

[7] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. SpAGNN: Spatially-aware graph neural networks for relational behavior forecasting from sensor data, 2019. 2, 6, 8

[8] Sergio Casas, Wenjie Luo, and Raquel Urtasun. IntentNet: Learning to predict intention from raw sensor data. In *Conf. on Robot Learning*, pages 947–956, 2018. 2

[9] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 2

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. of Conf. on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014. 4

[11] Ming-Fang Deo and John Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *IEEE Intelligent Vehicles Symposium*, 2018. 2, 6

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Conf. on Neural Information Processing Systems*, 2014. 2

[13] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018. 2, 4, 5, 6, 7, 13

[14] Hyowon Gweon and Rebecca Saxe. Developmental cognitive neuroscience of theory of mind. In *Neural Circuit Development and Function in the Brain*, chapter 20, pages 367–377. Academic Press, 2013. 1

[15] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995. 2

[16] 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. In *Int. Conf. on Learning Representations*, 2017. 5

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 4

[18] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *IEEE Int. Conf. on Computer Vision*, pages 2375–2384, 2019. 2, 3, 4, 5, 6, 7, 13

[19] Boris Ivanovic, Edward Schmerling, Karen Leung, and Marco Pavone. Generative modeling of multimodal multi-human behavior. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2018. 2, 4

[20] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. In *Conf. on Robot Learning*, 2019. 2

[21] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-RNN: Deep learning on spatio-temporal graphs. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 2, 4

[22] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft Level 5 AV Dataset 2019. https://level5.lyft.com/dataset/, 2019. 2

[23] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *Int. Journal of Robotics Research*, 32(11):1238 – 1274, 2013. 2

[24] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S. Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks. In *Conf. on Neural Information Processing Systems*, 2019. 2, 5, 6, 7, 13

[25] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2014. 2, 5, 6

[26] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. 2

[27] Namhoon Lee and Kris M. Kitani. Predicting wide receiver trajectories in American football. In *IEEE Winter Conf. on Applications of Computer Vision*, 2016. 2

[28] Jeremy Morton, Tim A. Wheeler, and Mykel J. Kochenderfer. Analysis of recurrent neural networks for probabilistic

modeling of driver behavior. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 18(5):1289–1298, 2017. 2

[29] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Int. Conf. on Machine Learning*, 2000. 2

[30] nuTonomy. nuscenes 3d object detection challenge. https://www.nuscenes.org/object-detection?externalData=all&mapData=all&modalities=Any, 2019. 6

[31] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016. 4

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017. 5

[33] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *IEEE Int. Conf. on Computer Vision*, 2009. 2, 5, 6

[34] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press, first edition, 2006. 2

[35] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *IEEE Int. Conf. on Computer Vision*, 2019. 2

[36] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, S. Hamid Rezatofighi, and Silvio Savarese. SoPhie: An attentive GAN for predicting paths compliant to social and physical constraints. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 2, 5, 6, 7, 13

[37] Amir Sadeghian, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese. CAR-Net: Clairvoyant attentive recurrent network. In *European Conf. on Computer Vision*, 2018. 2, 6

[38] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Conf. on Neural Information Processing Systems*, 2015. 2, 4

[39] Luca Anthony Thiede and Pratik Prabhanjan Brahma. Analyzing the variety loss in the context of probabilistic trajectory prediction. In *IEEE Int. Conf. on Computer Vision*, 2019. 5, 6

[40] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *Proc. IEEE Conf. on Robotics and Automation*, 2018. 2, 3, 4, 6, 7

[41] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 30(2):283–298, 2008. 2

[42] Waymo. Safety report, 2018. Available at https://waymo.com/safety/. Retrieved on November 9, 2019. 1

[43] Waymo. Waymo Open Dataset: An autonomous driving dataset. https://waymo.com/open/, 2019. 2

[44] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 2

[45] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Balancing learning and inference in variational autoencoders. In *Proc. AAAI Conf. on Artificial Intelligence*, 2019. 5

[46] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. Technical report, Megvii, 2019. 6

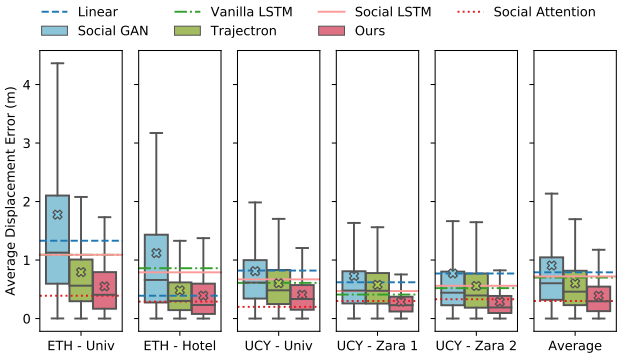

Figure 8. ADE results of all methods per dataset, as well as their average performance. Boxplots are shown for all generative models since they produce distributions of trajectories. 2000 trajectories were sampled per model at each prediction timestep, with each sample's ADE included in the boxplots. Our model's z_{mode} output is plotted here. X markers indicate the mean ADE. Mean ADE from deterministic baselines are visualized as horizontal lines.

A. Average Displacement Error Evaluation

In addition to FDE, we also evaluate *Trajectron++* using ADE. Figure 8 summarizes these results and, as in Figure 3, shows that our model is competitive with state-of-the-art deterministic methods.

B. Best of N Evaluation

We also compare our method to other generative approaches with a best-of-N metric. Table 5 summarizes these results, and shows that our method significantly outperforms others.

C. nuScenes Pedestrian Evaluation

Our method's pedestrian prediction quality is reported in Table 6. We can see a similar trend as in Figure 5 in terms of performance degradation over time, however, even 4 seconds into the future our model's error is reasonable.

Table 5. Quantitative ADE and FDE results, using a best-of-N metric where $N = 20$. Our model's Full output scheme was used here. Bold indicates the best values.

Dataset	**ADE / FDE, Best of 20 Samples (m)**					
	S-GAN [13]	S-GAN-P [13]	SoPhie [36]	Social-BiGAT [24]	Trajectron [18]	Ours (Full)
ETH	0.81 / 1.52	0.87 / 1.62	0.70 / 1.43	0.69 / 1.29	0.59 / 1.14	**0.35 / 0.77**
Hotel	0.72 / 1.61	0.67 / 1.37	0.76 / 1.67	0.49 / 1.01	0.35 / 0.66	**0.18 / 0.38**
Univ	0.60 / 1.26	0.76 / 1.52	0.54 / 1.24	0.55 / 1.32	0.54 / 1.13	**0.22 / 0.48**
Zara 1	0.34 / 0.69	0.35 / 0.68	0.30 / 0.63	0.30 / 0.62	0.43 / 0.83	**0.14 / 0.28**
Zara 2	0.42 / 0.84	0.42 / 0.84	0.38 / 0.78	0.36 / 0.75	0.43 / 0.85	**0.14 / 0.30**
Average	0.58 / 1.18	0.61 / 1.21	0.54 / 1.15	0.48 / 1.00	0.56 / 1.14	**0.21 / 0.45**

Table 6. Pedestrian-only error evaluation. Results from both our model's Full generative output and Mode-Mode deterministic output schemes are reported. 2000 trajectories were sampled per pedestrian for the Full output.

Metric (Output)	@1s	@2s	@3s	@4s
ADE (Full)	0.05	0.11	0.20	0.30
ADE (MM)	0.04	0.09	0.16	0.25
FDE (Full)	0.05	0.22	0.43	0.69
FDE (MM)	0.04	0.17	0.35	0.58
KDE NLL (Full)	-0.33	-1.05	-0.74	-0.19