SMART: Simultaneous Multi-Agent Recurrent Trajectory Prediction

Sriram N N[1], Buyu Liu[1], Francesco Pittaluga[1], and Manmohan Chandraker[1,2]

[1] NEC Laboratories America
[2] UC San Diego

Abstract. We propose advances that address two key challenges in future trajectory prediction: (i) multimodality in both training data and predictions and (ii) constant time inference regardless of number of agents. Existing trajectory predictions are fundamentally limited by lack of diversity in training data, which is difficult to acquire with sufficient coverage of possible modes. Our first contribution is an automatic method to simulate diverse trajectories in the top-view. It uses pre-existing datasets and maps as initialization, mines existing trajectories to represent realistic driving behaviors and uses a multi-agent vehicle dynamics simulator to generate diverse new trajectories that cover various modes and are consistent with scene layout constraints. Our second contribution is a novel method that generates diverse predictions while accounting for scene semantics and multi-agent interactions, with constant-time inference independent of the number of agents. We propose a convLSTM with novel state pooling operations and losses to predict scene-consistent states of multiple agents in a single forward pass, along with a CVAE for diversity. We validate our proposed multi-agent trajectory prediction approach by training and testing on the proposed simulated dataset and existing real datasets of traffic scenes. In both cases, our approach outperforms SOTA methods by a large margin, highlighting the benefits of both our diverse dataset simulation and constant-time diverse trajectory prediction methods.

Keywords: Diverse trajectory prediction, multiple agents, constant time, scene constraints, simulation

1 Introduction

The ability to reason about the future states of multiple agents in a scene is an important task for applications that seek vehicle autonomy. Ideally, a prediction framework should have three properties. First, it must be able to predict multiple plausible trajectories in the dominant modes of motion. Second, these trajectories should be consistent with the scene semantics. Third, it is attractive for several applications if constant-time prediction can be achieved regardless of the number of agents in the scene. In this paper, we propose dataset creation and future prediction methods that help achieve the above three properties (Figure 1).

A fundamental limitation for multimodal trajectory prediction is the lack of training data with a diverse enough coverage of the possible motion modes

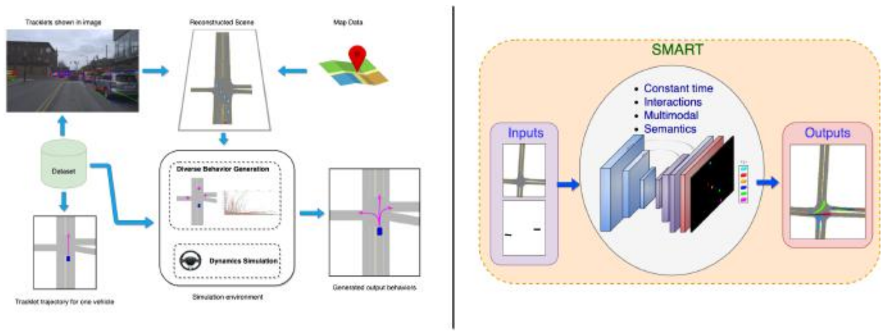


Fig. 1: Left: Given the map and tracklets, we propose to reconstruct the real world scene in top-view and simulate diverse behaviors for multiple agents w.r.t. scene context. Right: The proposed SMART algorithm that is able to generate context aware and multimodal trajectories for multiple agents.

in a scene. Our first main contribution is a simulation strategy to recreate driving scenarios from real world data, which generates multiple driving behaviors to obtain diverse trajectories for a given scene. We construct a graph-based simulation environment that leverages scene semantics and maps to execute realistic vehicle behaviors in the top-view. We sample reference velocity profiles from trajectories executing similar maneuvers in the real world data. Then we use a variant of the Intelligent Driver Model [37,24] to model the dynamics of vehicle driving patterns and introduce lane-change decisions for simulated vehicles based on MOBIL [38]. We show that training with our simulated datasets leads to large improvements in prediction outcomes compared to the real data counterparts that are comparatively limited in both scale and diversity reflected by a Wasserstein metric.

Several recent works consider deep networks for trajectory prediction for humans [14,4,21,2,31] and vehicles [22,34,15,6,40,28]. Usually, they consider interactions among multiple agents, but still operate on single agent basis at inference time, requiring one forward pass for each agent in the scene. Vehicle motions are stochastic and depending on their goals, obtaining multimodal predictions for individual vehicles that are consistent with the scene significantly increases the time complexity. Our second main contribution addresses this through a novel approach, Simultaneous Multi-Agent Recurrent Trajectory (SMART) prediction. To the best of our knowledge, it is the first method to achieve multimodal, scene-consistent prediction of multiple agents in constant time.

Specifically, we propose a novel architecture based on Convolutional LSTMs (ConvLSTMs) [39] and conditional variational autoencoders (CVAEs) [33], where agent states and scene context are represented in the bird-eye-view. Our method predicts trajectories for n agents with a time complexity of $O(1)$ (Table 1). To realize this, we use a single top-view grid map representation of all agents in the scene and utilize fully-convolutional operations to model the output predictions. Our ConvLSTM models the states of multiple agents, with novel state pooling

Table 1: Comparison of our method with existing works in terms of complexity, scene context and interactions. n and K are number of agents and iterations.

Method	Social GAN[14]	Desire[22]	SoPhie[31]	INFER[34]	MATF GAN[40]	Ours
Complexity	$O(n)$	$O(nK)$	$O(n^2)$	$O(n)$	$O(n)$	$\boldsymbol{O(1)}$
Scene Context	✗	✓	✓	✓	✓	✓
Social Interactions	✓	✓	✓	✓	✓	✓

operations, to implicitly account for interactions among objects and handle dynamically changing scenarios. To obtain multimodal predictions, we assign labels to trajectories based on the type of maneuver they execute and query for trajectories executing specific behaviors at test time. Our variational generative model is conditioned on this label to capture diversity in executing maneuvers of various types.

We validate our ideas on both real and simulated datasets and demonstrate state-of-the-art prediction numbers on both. We evaluate the network performance based on *average displacement error*(ADE), *final displacement error*(FDE) and *likelihood*(NLL) of the predictions with respect to the ground truth. Our experiments are designed to highlight the importance of methods to simulate datasets with sufficient realism at larger scales and diversity, as well as a prediction method that accounts for multimodality while achieving constant-time outputs independent of the number of agents in the scene.

To summarize, our key contributions are:

– A method to achieve constant-time trajectory prediction independent of number of agents in the scene, while accounting for multimodality and scene consistency.
– A method to simulate datasets in the top-view that imbibe the realism of real-world data, while augmenting them with diverse trajectories that cover diverse scene-consistent motion modes.

2 Related Work

In this section, we briefly summarize datasets available for autonomous driving and talk about the existing forecasting techniques.

Simulators and Autonomous Driving Datasets: AirSim [32] and CARLA [10] are autonomous driving platforms with primary target towards testing learning and control algorithms. SYNTHIA [30] introduces a big volume of synthetic images with annotations for urban scenarios. Virtual KITTI [12] imitates KITTI driving scenarios with varying environmental conditions and provide both pixel level and instance level annotations. Fang et al.[11] shows detectors trained with augmented lidar point cloud from a simulator provide comparable results with methods trained on real data. [17] generates images for vehicle detection and show an improvement in result that a deep neural network trained with synthetic data performs better than a network trained on real data when the dataset is bigger. Similarly, our focus is also to obtain better driving predictions by virtue of diversity from simulated datasets which emulate realistic driving behaviors.

Until recently, KITTI [13] has been extensively used for evaluation of various computer vision applications like stereo, tracking and object detection, but has limited diverse behaviors for a scene. NGSIM [8] provides trajectory information of traffic participants but the scenes are only limited to highways with fixed lane traffic. CaliForecasting [29] (unreleased) contains 10K examples with approximately 1.5 hours of driving data but does not contain any information about the scene. There are several recently proposed autonomous driving datasets [15,7,6,5,26,1,18], some of which focus on trajectory forecasting [15,7,6,26]. Rules of the road [15] (unreleased) proposes a dataset with map information for approximately 83K trajectories in 88 distinct locations. Argoverse [7] proposes two datasets (Tracking and Forecasting) with HD semantic map information containing centerlines. Argoverse tracking contains a total of 113 scenes with tracklet information. While the forecasting dataset is sufficiently large enough with more than 300K trajectories, it contains 5 seconds trajectory data for only one vehicle in the scene. In our work, we simultaneously generate trajectories for all vehicles in the scene and provide trajectory information up to 7 seconds for each vehicle. NuScenes [5] provides data from two cities with complete sensor suite information, but the main focus of the dataset is towards object detection and tracking. We primarily focus on using Argoverse Tracking[7] for simulating diverse trajectories and to showcase better prediction ability, but also simulate diverse trajectories for KITTI[13] dataset (please see supplementary material). Our method can be extended to many published datasets such as Waymo Open Dataset [1] and Lyft [18].

Forecasting Methods: Motion forecasting has been extensively studied. Kitani et al. [19] proposes a method based on Inverse Optimal Control (IOC). Social LSTM [3] uses a recurrent network to model human-human interactions for pedestrian forecasting. Deo et al. [9] use a similar method as [3] to model interactions and predict an output distribution over future states for vehicles. DESIRE [23] uses a CVAE-based [33] approach to predict trajectories up to 4 seconds but requires multiple iterations to align its predictions with scene context. Sampling multiple trajectories that are semantically aligned might not be feasible.

Over the recent years, generative models [14,4,31,20,25] have shown significant improvements in pedestrian trajectory prediction. Human trajectories tend to be stochastic and random while vehicle motions are aligned with the scene context and are strongly influenced by surrounding vehicle's behavior. Outputs from [14,31] show that it produces more diverse outputs and the output predictions are spread over a larger area. While more advanced methods [20,16] show outputs that are tightly coupled with the ground truth. We do not intend to capture the data distribution in such a fashion but are more focused towards producing predictions in possible dominant choices of motion[35]. This also motivates us to use [14] to showcase the ability of simulation strategy in producing more diverse outputs. Our method capture both these indifference's by producing

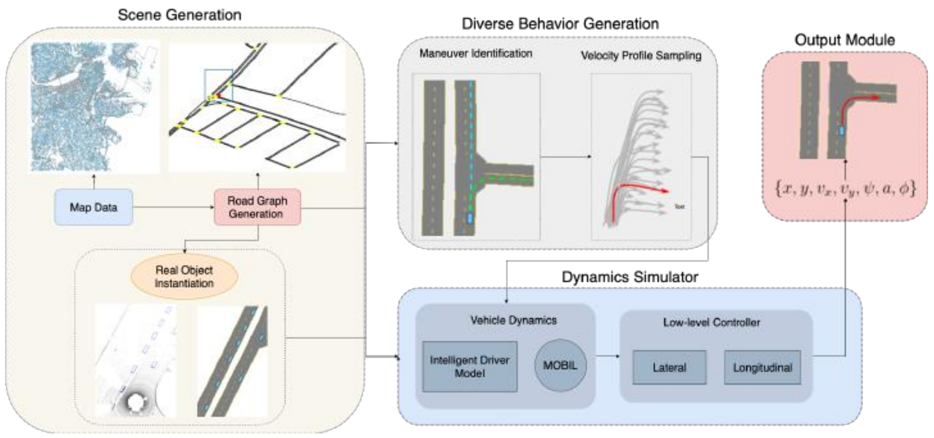


Fig. 2: The overall pipeline of the proposed simulation strategy. Scene generation module recreates the specific driving scenarios from datasets. Behavior generation samples a behavior for simulation and an appropriate reference velocity profile for every vehicle in the scene. The dynamics simulator tracks the reference velocity and provides lane changing decisions based on the current traffic condition.

quintessential trajectories that are diverse and at the same time closely aligned with the ground truth (indicated by our likelihood values).

Recently, MATF GAN[40] uses convolutions to model interactions between agents, but only shows results on straight driving scenarios and suffers in producing multimodal outputs. INFER [34] proposes a method based on ConvLSTMs and [15] uses convolutions to regress future paths. These methods nicely couple scene context with predicted output, but their predictions are entity-centric and do not incorporate multi-agent stochastic predictions.

3 Simulated Dataset

The overall pipeline of the proposed simulation strategy is shown in Figure 2. Our simulation engine consists of three main components: scene generation module, behavior generation module and a dynamics simulation engine. Given a dataset to recreate and simulate, the scene generation module takes lane centerline information that can be either acquired through openly available map information[27] or provided by the dataset. We utilize this information to create a graph data structure that consists of nodes and edges representing end points of the lane and lane centerline respectively. This when rendered provides us with a Birds-Eye-View reconstruction of the local scene. We call this as *road graph*. The object instantiation module uses the tracklet's information from the dataset to project them on to the generated road graph. We do so by defining a coordinate system with respect to the ego vehicle and find the nearest edge occupied by the objects in our graph. This completes our scene reconstruction task. Now, for every vehicle that was instantiated in the scene, we find various possible maneuvers that it can execute given the traffic conditions and road structure from which, we uniformly sample different vehicle behaviors for our simulation. We refer to *behaviors* as vehicles executing different maneuvers like *straight, left turn, right turn and lane changes*. To execute such diverse behaviors that are significantly realistic, we sample appropriate velocity profiles from real dataset as references that closely resemble the intended behavior that vehicle is planning to execute. The dynamics simulation module utilizes this reference velocity to execute the right behavior for every vehicle but at the same time considers the scene layout and the current traffic conditions to provide a safe acceleration that can be executed. We simulate every scene for 7 seconds and generate a maximum of 3 diverse behaviors (Figure 3). The simulation is performed at 10Hz and output from our simulation consists of vehicle states $\{\mathbf{x}, \mathbf{v}, \psi, a, \phi\}_1^T$ which represent position, velocity, heading, acceleration and steering over the course of our simulation. We will now provide a brief description of each component and refer readers to supplementary material for additional details.

Scene Generation We utilize the lane information from OpenStreetMaps (OSM) [27] or from datasets like [7] for creating the road graph. For our purposes, we make use of the road information such as centerline, number of lanes and one-way information for each road segment. Every bi-directional road centerline is split based on the specified number of lanes and one-way information. The vehicle pose information from the dataset is used to recreate exact driving scenarios.

Diverse Behavior Generation Given a particular lane ID (node) on the local road graph for every vehicle, we depth first explore K possible leaf nodes that can be reached within a threshold distance. We categorize plausible maneuvers from any given node into three different categories {*left, right, straight*}. Prior to the simulation, we create a pool of reference velocity profiles from the real data. At simulation time, after sampling a desired behavior, we obtain a Nearest Neighbor velocity profile for the current scene based on features such as distance before turn and average velocity, for turn and straight maneuvers respectively.

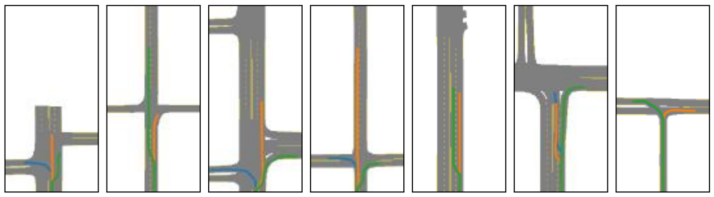


Fig. 3: Example trajectories executed by a single vehicle under different scenes in simulation.As shown in this figure, our simulation strategy is able to generate diverse yet realistic trajectories that align well with scene context.

Dynamics Simulation The dynamics module utilizes road graph, a behavior from a pool of diverse plausible ones and a reference velocity that needs to be tracked for the appropriate behavior. Our dynamics engine is governed by Intelligent Driver Model (IDM)[37] and MOBIL[38]. Acceleration and lane change decisions obtained from this dynamics module is fed to a low-level controller that tries to track and exhibit appropriate state changes in the vehicle behavior. In order to limit the acceleration under safety limit for the any traffic situation and to incorporate interactions among different agents in the scene we use an IDM[37] behavior for the simulated vehicles. The input to an IDM consists of distance to the leading vehicle s, the actual velocity of the vehicle v, the velocity difference with the leading vehicle Δv and provides an output a_{IDM} that is considered safe for the given traffic conditions. It is given by the equation,

$$a_{IDM}(s, v, \Delta v) = a\left(1 - \left(\frac{v}{v_o}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right), \tag{1}$$

where, a is the comfortable acceleration and v_o is the desired reference velocity. δ is an exponent that influences how acceleration decreases with velocity. The deceleration of the vehicle depends on the ratio of desired minimum gap s^* to actual bumper distance s with the leading vehicle.

Lane Change Decisions: We also consider lane changing behavior to add additional diversity in vehicle trajectories apart from turn based maneuver trajectories. Lane changing behaviors are modeled based on MOBIL algorithm from [38]. The following are the parameters that control lane changing behavior: politeness factor p that influences lane changing if there's acceleration gain for other agents, lane changing acceleration threshold Δa_{th}, maximum safe deceleration b_{safe} and bias for particular lane Δa_{bias}. The following equations govern whether a lane change can be executed,

$$\tilde{a}_c - a_c + p\big\{(\tilde{a}_n - a_n) + (\tilde{a}_o - a_o)\big\} > \Delta a_{th} - \Delta a_{bias}, \tag{2}$$

$$(\tilde{a}_n - a_n) > -b^n_{safe}, (\tilde{a}_c - a_c) > -b^c_{safe}. \tag{3}$$

Here, a is the current acceleration and $\tilde{a}$ represents the new acceleration after lane change. c, n, o subscripts denote current, new vehicle and old vehicles respectively.

4 SMART

In this section, we will introduce a single representation model to predict trajectories for *multiple* agents in a road scene such that our predictions are *context aware*, *multimodal* and have *constant inference time* irrespective of number of agents. We formulate the trajectory prediction problem as per frame regression of agents locations over the spatial grid. We will describe our method below in details.

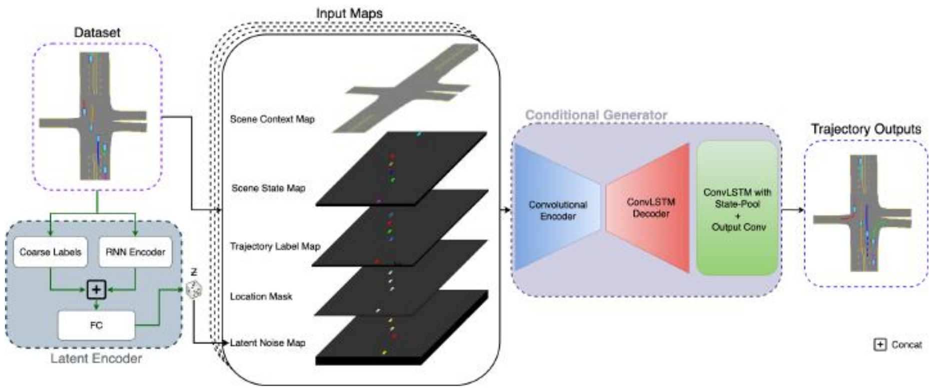


Fig. 4: The overall architecture for SMART framework. The components connected in green are used only during the training phase. It takes in a single representation of the scene, to regress per timestep coordinates for the all agents at their respective location in the spatial grid.

4.1 Problem formulation

Given the lane centerline information $L^{1 \cdots m}$ for a scene, we render them in top view representations such that our scene context map $\mathcal{I}$ is of HxWx3 where channel dimension represents one-hot information of each pixel corresponding to $\{$*road*, *lane*,*unknown*$\}$ road element. Let $\mathbf{X}_i = \{X_i^1, X_i^2, ..., X_i^T\}$ denote trajectory information of i^{th} vehicle from timestep $1...T$ where each $X_i^t = (x_i, y_i)^t$ represents spatial location of the agent in the scene. Our network takes input in the form of relative coordinates ${}^R\mathbf{X}_i$ with respect to agent's starting location. For the i^{th} agent in the scene, we project ${}^R\mathbf{X}_i$ at corresponding $\mathbf{X}_i$ locations to construct a spatial location map of states $\mathcal{S}^{1 \cdots T}$ such that $\mathcal{S}^t[X_i^t]$ contains relative coordinate of i^{th} agent at timestep t. ${}^R\mathbf{Y}_i = {}^R\ \mathbf{X}_i^{t_{obs} \cdots T}$ represents ground truth trajectory. And we further denote $\mathcal{M}^t$ as the location mask representing configuration of agents in the scene. To keep track of vehicles across timesteps, we construct a vehicle IDs map $\mathcal{V}^{1 \cdots T}$ where $\mathcal{V}^t[X_i^t] = i$. Furthermore, we associate each trajectory $X_i^{t_{obs}, \cdot\cdot T}$ with a label c_i that represents the behavioral type of the trajectory from one of $\{$*straight, left, right*$\}$ behaviors. And trajectory label for lane changes falls in one of the three categories. Let $\mathcal{C}$ encode grid map representation of c_i such that $\mathcal{C}^t[X_i^t] = c_i$. Note that vehicle trajectories are not random compared to the human motion. Instead, they depend on behaviors of other vehicles in the road, which motivates us to classify trajectories based on different maneuvers.

We follow the formulation proposed in [40,15,34] where network takes previous states $\mathcal{S}^{1 \cdot\cdot t_{obs}}$ as input along with the scene context map $\mathcal{I}$, trajectory label map $\mathcal{C}$, location mask $\mathcal{M}$ and a noise map $\mathcal{Z}$ to predict the future trajectories ${}^R\hat{\mathbf{Y}}_i$ for every agent at its corresponding grid map location X_i^t in the scene. Note that we do not have a separate head for each agent. Instead, our network predicts a single future state map $\hat{\mathcal{S}}^t$ where each individual agent tries to match ${}^R\mathbf{Y}_i^t$ at t.

4.2 Method

We illustrate our pipeline in Figure 4. Our network architecture comprises of two major parts, a latent encoder and a conditional generator. We model the temporal information with the agents previous locations using ConvLSTMs. We further introduce a state pooling operation to feed agents state information at respective locations in consecutive timestep. While we provide trajectory specific labels to capture diverse predictions, we leverage conditional variational generative models (CVAE[33]) to model diversity in the data for each type of label.

Latent Encoder: It acts as a recognition module Q_ϕ for our CVAE framework and is only used during our training phase. Specifically, it takes in both the past and future trajectory information ${}^R\mathbf{X}_i$ and passes them through an embedding layer. The embedded vectors are then passed on to a LSTM network to output encoding at every timestep. The outputs across all the timesteps are concatenated together into a single vector along with the one hot trajectory label c_i to produce $V_{enc}(i)$. This vector is then passed on through a MLP to obtain μ and σ to output a distribution $Q_\phi(z_i|{}^R\mathbf{X}_i, c_i)$. Formally,

$$\begin{aligned} {}^o h_i^t &= LSTM(h_i^{t-1}, {}^R X_i^t) \\ V_{enc}(i) &= [{}^o h_i^1, ..., {}^o h_i^T, c_i] \\ \mu, \sigma &= MLP(V_{enc}(i)). \end{aligned} \tag{4}$$

Conditional Generator: We adapt a U-Net like architecture for the generator. At any timestep t, the inputs to the network conditional generator are the following, a scene context map $\mathcal{I}$ (HxWx3), a single representation of all agents current state $\mathcal{S}^t$ (HxWx2), location mask $\mathcal{M}^t$ (HxWx1), a one-hot trajectory specific label for each agent projected at agent specific locations in a grid from $\mathcal{C}^t$ (HxWx3) and a latent vector map $\mathcal{Z}^t$ (HxWx16) containing z_i obtained from $Q_\phi(z_i|{}^R\mathbf{X}_i, c_i)$ during training phase or sampled from prior distribution $P_v(z_i|{}^R\mathbf{X}_i, c_i)$ at test time. Formally the network input E^t is given by:

$$E^t = [\mathcal{I}, \mathcal{S}^t, \mathcal{M}^t, \mathcal{C}^t, \mathcal{Z}^t], \tag{5}$$

which is of size HxWx25 for any timestep t. Note that our representation is not entity centric i.e we do not have one target entity for which we want to predict trajectories but rather have a global one for all agents.

At each timestep from $1, ..., t_{obs}$, we pass the above inputs through the encoder module. This module is composed of strided convolutions, which encode information in small spatial dimensions, and passes them through the decoder. The decoder includes ConvLSTMs and transposed convolutions with skip connections from the encoder module, and outputs a HxW map. It is then passed on to another ConvLSTM layer with state pooling operations. The same network is shared during observation and prediction phase. A final 1x1 convolution layer is added to output a 2 channel map containing relative predicted coordinates ${}^{R}X_i^t$ for the agents in the next timestep.

We use the ground truth agent locations for the observed trajectory and unroll our ConvLSTM based on the predictions of our network. During the prediction phase $(t_{obs}, ..., T)$, the outputs are not directly fed back as inputs to the network rather the agent's state is updated to the next location in the scene based on the predictions. The relative predicted location ${}^{R}\hat{X}_i^{t-1}$ gets updated to absolute predicted location $\hat{X}_i^t$ to obtain a updated scene state map $\hat{\mathcal{S}}^t$ containing updated locations of all the agents in the scene. Note that using such representations for the scene is agnostic to number of agents and as the agents next state is predicted at its respective pixel location it is capable of handling dynamic entry and exit of agents from the scene.

State-Pooled ConvLSTMs: Simultaneous multi-agent predictions are realized through state-pooling in ConvLSTMs. Using standard ConvLSTMs for multi-agent trajectory predictions usually produces semantically aligned trajectories, but the trajectories occasionally contain erratic maneuvers. We solve this issue via state-pooling, which ensures the availability of previous state information when trying to predict the next location. We pool the previous state information from the final ConvLSTM layer for all the agents ${}^{sp}\mathbf{H}_i^{t-1}$ and initialize the next state with ${}^{sp}\mathbf{H}_i^{t-1}$ (for both hidden and cell state) at agents updated locations and zero vectors at all other locations for timestep t.

Learning: We train both the recognition network $Q_\phi(z_i|{}^{R}\mathbf{X}_i, c_i)$ and the conditional generator $P_\theta(Y|E)$ concurrently. We obtain predicted trajectory ${}^{R}\hat{\mathbf{Y}}$ by pooling values from indexes that agents visited at every timestep. We use two loss functions in training our CVAE based ConvLSTM network:

- Reconstruction Loss: $\mathcal{L}_R = \frac{1}{N}\sum_i^N ||{}^{R}\mathbf{Y}_i - {}^{R}\hat{\mathbf{Y}}_i||$ that penalizes the predictions to enable them to reconstruct the ground truth accurately.
- KL Divergence Loss: $\mathcal{L}_{KLD} = D_{KL}(Q_\phi(z_i|{}^{R}\mathbf{X}_i, c_i)||P_v(z_i|{}^{R}\mathbf{X}_i, c_i))$. That regularizes the output distribution from Q_ϕ to match the sampling distribution P_v at test time.

Test phase: At inference time, we do not have access to trajectory specific labels c_i but rather query for a specific behavior by sampling these labels randomly. Along with c_i for each agent we also sample z_i from $P_v(z_i|{}^{R}\mathbf{X}_i, c_i)$. However, P_v can be relaxed to be independent of the input[33] implying the prior distribution to be $P_v(z_i)$. $P_v(z_i) := \mathcal{N}(0, 1)$ at test time.

5 Experiments

We evaluate our methods on publicly available Argoverse[7] Tracking(ArgoT) [3] and Forecasting(ArgoF) [4] dataset.We also introduce a simulated dataset based

[3] Generated 2044 scenes in total containing multiple trajectories for every scene

[4] Argoverse Forecasting for vehicle trajectory prediction is a large scale dataset containing 333,441 (5sec) trajectories captured from 320 hours of driving.

Table 2: Quantitative measurements on P-ArgoT. We report ADE, FDE (in meters) and NLL (N=5)

Model	1.0(sec)			2.0(sec)			3.0(sec)			4.0(sec)			5.0(sec)		
P-ArgoT ‖ ADE | FDE | NLL‖															
LSTM	0.53	0.87	-	1.03	2.01	-	1.62	3.41	-	2.31	5.09	-	3.09	6.98	-
CVAE	0.46	0.73	**2.16**	0.89	1.72	**3.31**	1.42	2.98	4.18	2.04	4.49	4.88	2.76	6.26	5.48
MATF Scene [40]	0.98	1.73	-	1.84	3.53	-	2.76	5.48	-	3.72	7.56	-	4.73	9.78	-
MATF GAN [40]	0.78	1.34	3.44	1.45	2.73	4.53	2.17	4.28	5.24	2.94	5.95	5.79	3.76	7.77	6.23
S-GAN[14]	**0.42**	0.72	2.21	0.85	1.68	3.49	1.36	2.83	4.36	1.93	4.08	5.03	2.54	5.46	5.57
SMART (c_{random})	0.73	0.64	3.72	0.84	0.98	3.92	0.94	1.35	4.31	1.15	1.73	4.70	1.38	2.16	5.06
SMART (c_{best})	0.58	**0.59**	3.21	**0.59**	**0.55**	3.39	**0.60**	**0.75**	**3.63**	**0.98**	**0.61**	**3.89**	**1.02**	**1.06**	**4.13**

Table 3: Left: Quantitative measurements on ArgoF validation set. (N=6). Right: Quantitative comparison of different datasets with introduced diversity metrics based on wasserstein distances.

Model	1.0(sec)			2.0(sec)			3.0(sec)		
Argo Forecasting Dataset (ArgoF) ‖ ADE | FDE | NLL‖									
LSTM	0.76	1.16	-	1.32	2.67	-	2.14	4.71	-
CVAE	1.22	2.27	9.14	2.56	5.32	11.7	4.14	8.94	13.2
MATF Scene [40]	1.56	2.71	-	2.90	5.54	-	4.35	8.17	-
MATF GAN [40]	1.48	2.54	13.5	2.72	5.17	13.6	4.08	8.13	14.0
S-GAN [14]	0.88	1.59	4.12	1.99	4.34	5.74	3.49	8.05	6.80
SMART (c_{random})	0.79	0.96	4.59	1.16	1.85	4.76	1.65	3.00	5.18
SMART (c_{best})	**0.71**	**0.83**	**3.56**	**1.03**	**1.55**	**4.05**	**1.44**	**2.47**	**4.61**

Datasets	Y Wasserstein		$\ddot{X}$ Wasserstein	
	Mean	Median	Mean	Median
KITTI	0.14	0.04	4.91	3.52
P-KITTI	**2.13**	**0.75**	**17.64**	**17.58**
ArgoT	0.49	**0.20**	5.98	2.97
P-ArgoT	**0.97**	0.12	**17.5**	**17.49**

on P-ArgoT and conduct experiments with it. Our simulated dataset utilizes 2000 scene instances from ArgoT to generate scenarios with multiple agents and trajectory durations of 7 seconds.

We use standard *evaluation metrics* suggested in previous approaches[7,40,34,14],e.g. Average Displacement Error(ADE), Final Displacement Error(FDE) and Negative Log Likelihood(NLL) with the ground truth.

We evaluate two versions of *SMART*, e.g. (SMART(c_{random})) and (SMART(c_{best})). For the former, we randomly sample our behavior specific trajectory labels for evaluation, while for the later we equally sample n trajectories over all the trajectory labels and report the best results across all. We comapare our proposed methods against the following *baselines*:

- LSTM: A sequence to sequence encoder-decoder network that regresses future locations based on the past trajectory [36].
- CVAE: A modified LSTM generator that predicts paths based on the input latent vector in the form of noise learned from the data distribution [33].
- S-GAN[14]: We implement and evaluate this method on all datasets.
- MATF GAN[40]: We implement it ourselves and evaluate it on all datasets.

Quantitative Results: We first demonstrate that our proposed method, **SMART**, can beat the baseline methods. As shown in Tab. 2 and 3(left), our method can almost always outperform baselines with a large margin, especially in long-term scenarios. It is also worth noting that final ADE and FDE values of our method SMART(c_{best}) are at least 23% and 39% lower than that of others

Table 4: Results for methods tested on ArgoT. '[ ]' represents the training set. We report results on the basis of ADE, FDE and NLL (N=5).

Model	1.0(sec)			2.0(sec)			3.0(sec)			4.0(sec)			5.0(sec)		
ArgoT ‖ ADE | FDE | NLL ‖															
LSTM [ArgoT]	0.65	1.07	-	1.28	2.53	-	2.07	4.45	-	3.00	6.74	-	4.05	9.31	-
CVAE [ArgoT]	**0.45**	0.75	**1.99**	0.90	1.88	**3.13**	1.52	3.48	**4.07**	2.30	5.50	4.89	3.21	7.89	5.63
MATF Scene [ArgoT] [40]	1.24	2.20	-	2.39	4.67	-	3.66	7.49	-	5.03	10.4	-	6.45	13.4	-
MATF GAN [ArgoT] [40]	0.97	1.69	5.32	1.82	3.47	6.21	2.75	5.53	6.93	3.77	7.88	7.58	4.90	10.5	8.15
MATF GAN [P-ArgoT] [40]	1.03	1.79	6.32	1.93	3.68	7.51	2.89	5.77	8.43	3.94	8.13	9.19	5.07	10.7	9.77
S-GAN [ArgoT] [14]	0.77	1.35	4.29	1.47	2.79	5.48	2.25	4.47	6.24	3.11	6.38	6.81	4.06	8.54	7.27
S-GAN [P-ArgoT] [14]	0.94	1.63	4.84	1.76	3.31	6.00	2.66	5.24	6.74	3.66	7.37	7.30	4.74	9.68	7.75
SMART [ArgoT] (c_{random})	0.85	1.06	4.31	1.22	1.87	4.82	1.68	2.98	5.38	2.25	4.30	5.88	2.88	5.70	6.31
SMART [P-ArgoT] (c_{random})	0.68	0.80	4.12	0.97	1.51	4.29	1.37	2.45	4.71	1.85	3.58	5.13	2.39	4.85	5.51
SMART [ArgoT] (c_{best})	0.74	0.87	3.85	1.03	1.50	4.02	1.42	2.40	4.42	1.90	3.52	4.84	2.45	4.74	5.24
SMART [P-ArgoT] (c_{best})	0.61	**0.66**	3.91	**0.84**	**1.21**	3.81	**1.16**	**1.97**	4.08	**1.56**	**2.91**	**4.39**	**2.02**	**3.96**	**4.70**

Table 5: Results for methods tested on ArgoT without straight trajectories. '[ ]' represents the training set. We report results on the basis of ADE, FDE(N=5).

Model	1.0(sec)		2.0(sec)		3.0(sec)		4.0(sec)		5.0(sec)	
ArgoT ‖ ADE | FDE ‖										
MATF GAN [ArgoT] [40]	1.04	1.80	1.98	3.86	3.08	6.50	4.41	9.91	5.98	14.1
MATF GAN [P-ArgoT] [40]	0.94	1.63	1.79	3.50	2.81	6.00	4.07	9.35	5.61	13.6
S-GAN [ArgoT] [14]	0.94	1.67	1.86	3.64	2.91	5.91	4.12	8.48	5.47	11.4
S-GAN [P-ArgoT][14]	0.93	1.61	1.74	3.30	2.65	5.23	3.65	7.38	4.73	9.75

across all tables. We also observe that SMART(c_{best}) provides better results than SMART(c_{random}). This is due to the fact that SMART(c_{random}) randomly samples trajectory behavior labels thus ignores the data distribution. In contrast, SMART(c_{best}) is able to capture the diversity for particular label through CVAEs. Although other methods [14,40,22] are also able to generate diverse trajectories, they have to sample a significant number of trajectories to get a predictions (or driver intents) exhibiting different behaviors. We later show in Fig. 5 that our method is able to model data distribution more effectively, e.g. achieves comparable/better results with less samples.

In Fig. 5(left), we show variation of ADE/FDE values with increasing number of samples in ArgoF. We observe that our method performs significantly better even with lower number of samples compared to baselines, which again supports our claim that methods like [14] requires much more samples to even to get comparable performance with our method reported in Tab. 3(left). Fig.5(right)

Table 6: Average runtime in seconds to generate one prediction sample in scenes from Argoverse[7] dataset with increasing number of agents, benchmarked on RTX2080Ti, 11GB GPU.

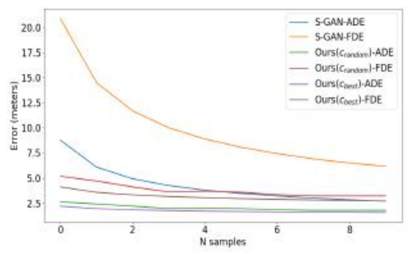
No. of Agents	1	2	3	4	5	6	7	8	9	10
SMART	.070	.070	.070	.070	.072	.069	.069	.075	.072	.069
S-GAN[14]	.024	.034	.044	.054	.062	.071	.082	.090	.102	.108

shows number of valid predictions produced across all datasets. Here, the validity is computed based on whether the predicted outputs lie within the road regions. Compared to baselines, our method is more likely to generate output predictions that satisfied context constraints.

We also provide analysis on time complexity of existing methods in Tab. 6. Without sacrificing the performance, our SMART always gives constant inference time with increasing number of agents.

To demonstrate the effectiveness/informativeness of our **simulated dataset**, we further conduct experiments and report numbers in Tab. 4. We test all methods on ArgoT test set. P-ArgoT in this table denotes that corresponding models are trained on P-ArgoT and fine-tuned on ArgoT training set. There are two main observations. Firstly, our methods that initialized with simulated data clearly achieve much better performance. Such significant performance boost indicates the benefits of augmenting a dataset with diverse trajectories. Secondly, such boost is missing in other methods. We argue that this might be attributed to the ability of the other methods in capturing the diversity in a wrong fashion (See Figure 3 in supplementary). For instance, S-GAN[14] is unaware of the scene context, hence when initialized with a model trained on diverse trajectories, the outputs are more spread thus leads to lower performance with fixed number of samples. Although MATF[40] includes scene context in its predictions, it has poor capability in producing multimodal outputs where most of the predictions are biased towards behavior of particular type (See Figure 3. [40]). To provide further analysis, and to show that training on simulated data improves diversity for other methods we report numbers evaluated on non-straight trajectories in Tab 5. As observed, other methods perform significantly better when when initialized with simulated model. This observation further demonstrates that our method is able to capture the diversity strongly coupled with the scene.

Qualitative Results We give some example predictions of our method in Fig. 6. In general, our predictions align well with scene context and obey traffic rules in most situations.



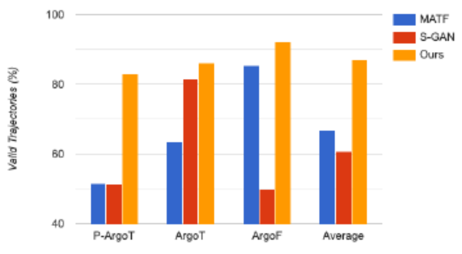


Fig. 5: **Left:**Quantitative results on ArgoF with increasing number of samples. Average and final displacements of our method is plotted against S-GAN[14]. **Right:** Percentage of samples(n=30) that produced trajectories inside the road.

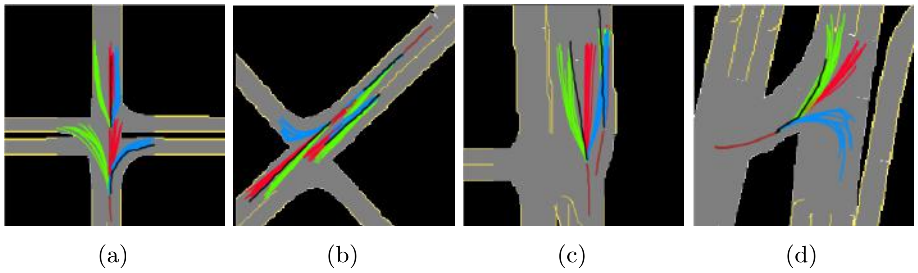


Fig. 6: Example predictions of SMART. The past trajectory and GT are visualized in brown and black lines. Red, blue and green lines are predictions sampled with different trajectory labels c_i given as input. From left, multi-agent prediction outputs from simulated dataset P-ArgoT, ArgoT, ArgoT and ArgoF datasets. (a),(b) and (c) show simultaneous multi-agent multimodal outputs. (d) shows a failure case where some of the predicted trajectories are aligned in opposite to the direction of road. However, we argue that such traffic rules might be hard to obtain with only top view map information.

Wasserstein diversity metric To quantify the diversity of the simulated dataset, we introduce a novel diversity metric based on Wasserstein distances and showcase our results on both real and simulated data. Firstly, we normalize the trajectories such that it starts at the origin and ends at some x coordinate. We use a trajectory with zero acceleration ($\ddot{x} = 0$) and zero deviation from the x axis ($y = 0$) as a reference trajectory for comparison. We define two metrics y (deviation from x axis) and $\ddot{x}$ (deviation from zero acceleration) Wasserstein. A higher Wasserstein metric indicates a higher deviation from the reference trajectory. Tab. 3(right) shows the Wasserstein metric between real and simulated data for two different datasets. Tracklets in KITTI [13] and Argoverse [7] generally move in straight directions with very minimal turns indicating a very low diversity. In contrast, our simulated trajectories are more diverse with agents executing turns whenever possible, going hand in hand with the higher diversity in Tab. 3(right).

6 Conclusion

In this paper, we have addressed data diversity and model complexity issues in multiple-agent trajectory prediction. We first introduced a new simulated dataset that includes diverse yet realistic trajectories for multiple agents. Further, we propose SMART, a method that simultaneously generates trajectories for all agents with a single forward pass and provides multimodal, context-aware SOTA predictions. Our experiments on both real and simulated dataset show superiority of SMART over existing methods in terms of both accuracy and efficiency. In addition, we demonstrate that our simulated dataset is diverse and general, thus, is useful to train or test prediction models.

References

1. Waymo open dataset: An autonomous driving dataset (2019)
2. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
3. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
4. Amirian, J., Hayet, J.B., Pettre, J.: Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2019)
5. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)
6. Chandra, R., Bhattacharya, U., Bera, A., Manocha, D.: Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8483–8492 (2019)
7. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., Hays, J.: Argoverse: 3d tracking and forecasting with rich maps. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
8. Colyar, J., Halkias, J.: Us highway 101 dataset. Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030 (2007)
9. Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. CoRR **abs/1805.06771** (2018), `http://arxiv.org/abs/1805.06771`
10. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
11. Fang, J., Yan, F., Zhao, T., Zhang, F., Zhou, D., Yang, R., Ma, Y., Wang, L.: Simulating LIDAR point cloud for autonomous driving using real-world scenes and traffic flows. CoRR **abs/1811.07112** (2018), `http://arxiv.org/abs/1811.07112`
12. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. CoRR **abs/1605.06457** (2016), `http://arxiv.org/abs/1605.06457`
13. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)
14. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social gan: Socially acceptable trajectories with generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2255–2264 (2018)
15. Hong, J., Sapp, B., Philbin, J.: Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
16. Ivanovic, B., Pavone, M.: The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs (2018)
17. Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S.N., Vasudevan, R.: Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? CoRR **abs/1610.01983** (2016), `http://arxiv.org/abs/1610.01983`

18. Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., Omari, S., Shah, S., Kulkarni, A., Kazakova, A., Tao, C., Platinsky, L., Jiang, W., Shet, V.: Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/ (2019)
19. Kitani, K.M., Ziebart, B.D., Bagnell, J.A., Hebert, M.: Activity forecasting. In: European Conference on Computer Vision. pp. 201–214. Springer (2012)
20. Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I.D., Rezatofighi, S.H., Savarese, S.: Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. CoRR **abs/1907.03395** (2019), `http://arxiv.org/abs/1907.03395`
21. Kosaraju, V., Sadeghian, A., Martn-Martn, R., Reid, I., Rezatofighi, S.H., Savarese, S.: Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks (2019)
22. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H.S., Chandraker, M.K.: Desire: Distant future prediction in dynamic scenes with interacting agents. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2165–2174 (2017)
23. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: Desire: Distant future prediction in dynamic scenes with interacting agents. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 336–345 (2017)
24. Leurent, E.: An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env` (2018)
25. Li, Y.: Which way are you going? imitative decision learning for path forecasting in dynamic scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 294–303 (2019)
26. Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W., Manocha, D.: Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. arXiv preprint arXiv:1811.02146 (2018)
27. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org` (2017)
28. Rhinehart, N., Kitani, K.M., Vernaza, P.: R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In: The European Conference on Computer Vision (ECCV) (September 2018)
29. Rhinehart, N., Kitani, K.M., Vernaza, P.: r2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. pp. 794–811. Springer International Publishing, Cham (2018)
30. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.: The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes (2016)
31. Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Savarese, S.: Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. CoRR **abs/1806.01482** (2018), `http://arxiv.org/abs/1806.01482`
32. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. CoRR **abs/1705.05065** (2017), `http://arxiv.org/abs/1705.05065`
33. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: Advances in neural information processing systems. pp. 3483–3491 (2015)

34. Srikanth, S., Ansari, J.A., Ram, R.K., Sharma, S., Murthy, J.K., Krishna, K.M.: Infer: Intermediate representations for future prediction. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Nov 2019). https://doi.org/10.1109/iros40897.2019.8968553, `http://dx.doi.org/10.1109/IROS40897.2019.8968553`
35. Sriram, N.N., Kumar, G., Singh, A., Karthik, M.S., Saurav, S., Bhowrnick, B., Krishna, K.M.: A hierarchical network for diverse trajectory proposals. In: 2019 IEEE Intelligent Vehicles Symposium (IV). pp. 689–694 (June 2019). https://doi.org/10.1109/IVS.2019.8813986
36. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. p. 31043112. NIPS14, MIT Press, Cambridge, MA, USA (2014)
37. Treiber, Hennecke, Helbing: Congested traffic states in empirical observations and microscopic simulations. Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics **62 2 Pt A**, 1805–24 (2000)
38. Treiber, M., Kesting, A.: Modeling lane-changing decisions with mobil. In: Appert-Rolland, C., Chevoir, F., Gondret, P., Lassarre, S., Lebacque, J.P., Schreckenberg, M. (eds.) Traffic and Granular Flow '07. pp. 211–221. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
39. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems. pp. 802–810 (2015)
40. Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., Wu, Y.N.: Multi-agent tensor fusion for contextual trajectory prediction. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

Supplementary Material

Abstract. In this supplementary, we provide more details for our proposed simulator in Sec. 1 and for the proposed method for trajectory prediction, SMART in Sec. 2. We further provide quantitative analysis and qualitative comparisons with a few prior works in Sec. 3. The accompanying video includes an overview of the method and qualitative visualizations of our predictions.

1 Further Details for Simulator

In this section, we provide more details about reference velocity sampling, low-level controller and also specify the range of values for IDM [37] parameters used in our simulation.

1.1 Velocity Sampling:

Figure 1 shows several real trajectories (x and y axes represent distance travelled in meters in longitudinal and lateral directions) and a plot between distance before turn and average velocity for real trajectory samples. We calculate distance before turn as the distance travelled by the vehicle before it starts executing a turn maneuver and average velocity is the mean velocity through the course of the trajectory before turn. Interestingly, we found distance before taking turns is highly correlated to average velocity. Specifically, we can see a trend of decrease(or slowing down) in average velocity while approaching an intersection with an intention of making a turn maneuver. To this end, we identify distance to intersections as useful feature and demonstrate that it helps in mimicking the real data. We label every velocity profile from the real data with a value of distance before turn or average velocity, for turn and straight maneuvers respectively. Here, by velocity profile we mean a series of vehicle velocities at every timestep for the simulation period. We identify the nearest neighbor velocity profile in the real data using the above mentioned features with the values from the simulated vehicle. We use the identified velocity profile as reference for the simulated vehicle to achieve at every timestep. We also add gaussian noise with zero mean and unit variance to add diversity in the sampled velocity profiles.

1.2 Low Level Controller:

Low-level controller simulates the desired behavior governed by vehicle dynamics module. It takes input from maneuver identification, IDM[37] and MOBIL[38], and produces state changes for the simulated vehicle. It consists of longitudinal and lateral proportional controllers that give out required velocity commands. The lane centerline is used as the reference trajectory for the simulated vehicle to follow. The velocity obtained from the lateral controller is converted to appropriate

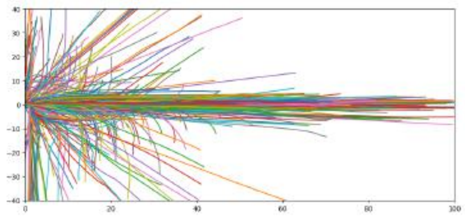
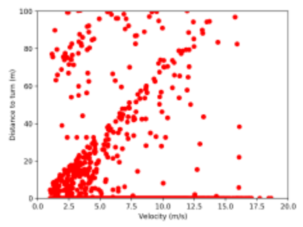



Fig. 1: **Left:** Shows several {*left,right and straight*} trajectories executed by vehicles in real world on Argoverse tracking (ArgoT) data sampled at 1Hz. **Right:** Velocity *vs* Distance to turn plot for tracklets in Argoverse tracking for the entire dataset. For plotting purposes we plot distance to turn to be zero for straight maneuvers. Here each dot represents a velocity profile for 7 seconds.

steering commands that helps in tracking the reference trajectory. Let v be the current velocity of the vehicle, $x_{lateral}$ be the lateral position from the lane and $v_{lateral}$ be the lateral velocity then steering angle ϕ is obtained through the following set of equations:

$$v_{lateral} = -kp_{lateral} * (x_{lateral} + \epsilon) \tag{6}$$

$$\psi_{req} = \arcsin(\frac{v_{lateral}}{v}) \tag{7}$$

$$\psi_{ref} = \psi_{future} + \psi_{req} \tag{8}$$

$$\dot{\psi} = kp_{heading} * \psi_{ref} \tag{9}$$

$$\phi = \arctan(\frac{L}{v}\dot{\psi}) \tag{10}$$

where $kp_{lateral}$ and $kp_{heading}$ are controller parameters, L represents length of the vehicle and ϵ acts as an offset noise in tracking the lane. ψ_{req} is the heading that needs to be compensated for aligning with the lane center, while ψ_{future} is the required heading that needs to be achieved for future timesteps. A heading controller provides a heading rate $\dot{\psi}$ for the given reference heading ψ_{ref}. Equation 10 calculates the steering angle based on current velocity v, vehicle length L and heading rate $\dot{\psi}$.

1.3 Intelligent Driver Model:

Tab 1 shows values and sample space for parameters in Intelligent Driver Model[37]. We sample these parameters randomly to increase diversity of driving patterns.

Table 1: Parameters for Intelligent Driver Model.

Parameter	Values	Units
Desired velocity v_o	Reference profile	m/s
Free acceleration exponent δ	4.0	-
Safety time gap T	$range(0.5, 2.5)$	s
Desired safety distance s_o	$range(0.5, 4.0)$	m
Comfortable acceleration a	1.5 ± 0.5	m/s^2
Comfortable deceleration b	2.0 ± 0.5	m/s^2

2 Further Details for SMART

2.1 Network Architecture:

In this section, we provide more details of our network architecture.

Latent Encoder: It takes concatenated past and future trajectories and a corresponding trajectory label as input. The embedding layer and LSTM contain 16 dimensions. The fully connected layer has 512 and 128 units that produce 16 dimensional μ and σ as outputs.

Convolutional Encoder: Our encoder receives input of HxWx25 dimension. It consists of 6 convolutional layers with first layer being 3D convolution and followed by 2D ones. The number of filters are 16,16,32,32,64 and 64 from the very first to the sixth layer, respectively, with alternativing stride 1 and stride 2. We set the kernel size to 1x1x4 in the first layer and that of the remaining layers to 3x3.

ConvLSTM Decoder It consists of alternating ConvLSTMs and 2D transposed convolutional layers. The size of hidden state or output filters in every pair of ConvLSTMs and transposed convolutions are 64,32 and 16, respectively. Convolutions share the same kernel size 3x3. And the transposed convolutions have a stride of 2. We add skip connections from encoder layers 2 and 4 to corresponding second and third ConvLSTM layers in the decoder.

Finally, a ConvLSTMs with state-pooling operation is further put to the end of decoder. It has a hidden state of 16 channels with 1x1 kernel size. We also concatenate features from first 3D convolution before feeding it to ConvLSTM layer. In addition, we include one last convolution layer that generates 2 channels with 1x1 kernel size as the output layer.

2.2 Learning Details

The models are trained using Adam optimizer with a learning rate of 0.008 and a batch size of 6. The model is trained on ArgoF for 10 epochs and 400 epochs on both ArgoT and P-ArgoT. We train the models at the trajectory frequency of 5hz and interpolate the results at the desired frequency. In order to avoid exploding gradients, we apply gradient clipping with L2 norm of 1.0. Further, during the training procedure, we augment the data by randomly rotating the scene and trajectories to reduce over-fitting. All models are implemented using Tensorflow 2.0 and trained with a NVIDIA RTX 2080Ti GPU.

2.3 Future work

SMART methods capabilities can be extended by incorporating traffic rules to reduce the number of invalid trajectories that span in the wrong direction (Figure 6d main paper). Furthermore, explicitly modelling interactions among multiple agents improve predictions and reduce invalid trajectory collisions with other agents in the scene.

3 Further Results

We provide more qualitative and quantitative analysis in this section.

Realism of Simulated Data To evaluate the realism of our simulated dataset, we perform PCA on a random set of real and simulated trajectories, followed by a Gaussian KDE on the PCA-transformed real trajectories. The calculated log likelihood for both real and simulated data on real fitted KDE for 1000 random datapoints are 2.25 and 2.19. This indicates that the simulated distribution falls very close to the real one. A qualitative plot of real and simulated trajectories after PCA transformation is shown in Figure 2.

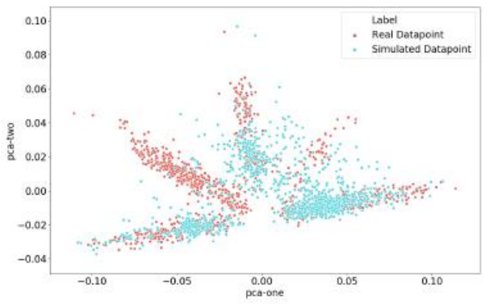


Fig. 2: Qualitative plot of PCA performed on 1000 random real and simulated trajectories from Argo Tracking

Visualization for diversity Figure 3 shows comparison of prediction outputs for methods trained only on real data and methods trained on simulated data and fine-tuned on real data. Clear improvement in the diversity can be observed with S-GAN[14]. But it does not capture such diversity with the scene context

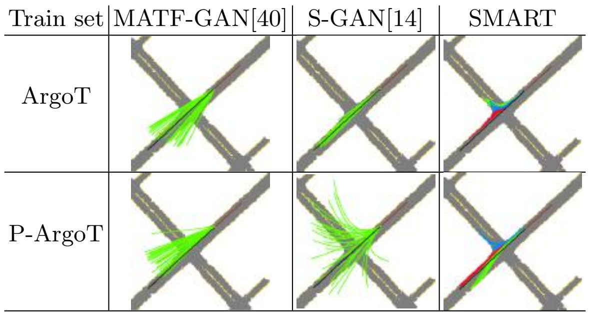


Fig. 3: Qualitative comparison of methods tested on Argo Tracking(ArgoT) with only training on ArgoT and training on P-ArgoT and fine tuning on ArgoT. Significant improvement in the diversity of prediction results can be observed in S-GAN[14] when the model is initialized using our simulated data.

attributing to higher deviation with the ground truth for the same number of samples while our method captures diversity coupled with the semantics in the right way enabling it to provide outputs towards different modes.

Results on KITTI We also conduct experiments to showcase the generalization of our simulated dataset. Specifically, we train models either on simulated or real ArgoT dataset and test them on KITTI. Our results in Tab. 2 show that models trained with our simulated but diverse dataset actually generate better results than that trained with real dataset. Such observation further shows that our simulated dataset generalizes better in general and has not been overfitted to particular dataset.

Table 2: Quantitative results on KITTI dataset. '[ ]' denotes the training set.

Model	1.0(sec)			2.0(sec)			3.0(sec)			4.0(sec)			5.0(sec)		
	KITTI Dataset ‖ ADE | FDE | NLL ‖														
SMART [ArgoT] (c_{random})	1.30	2.06	8.45	2.47	4.76	8.50	3.88	7.97	8.75	5.19	9.11	8.93	6.14	11.2	8.98
SMART [PArgoT] (c_{random})	1.14	1.77	5.33	2.12	3.94	6.03	3.29	6.55	6.61	4.53	8.69	7.13	5.88	9.92	7.58
SMART [ArgoT] (c_{best})	1.22	1.92	7.10	2.30	4.38	7.42	3.58	7.28	7.87	4.75	8.10	8.174	5.65	9.88	8.28
SMART [PArgoT] (c_{best})	**1.03**	**1.56**	**5.29**	**1.89**	**3.50**	**5.95**	**2.94**	**5.77**	**6.56**	**4.06**	**7.60**	**7.08**	**5.32**	**8.71**	**7.49**

Further Qualitative Results Figures 4 and 5 show qualitative comparisons of the proposed SMART method with other baselines.

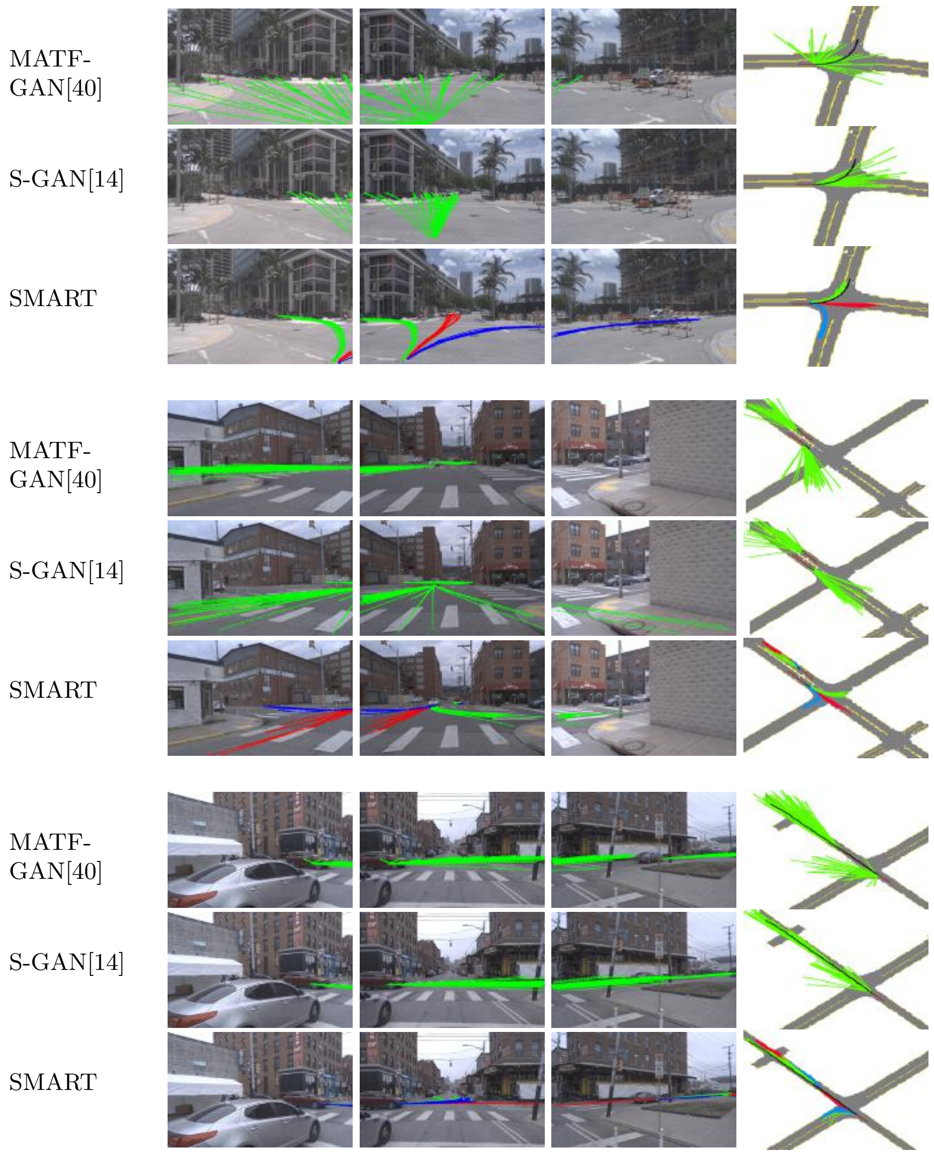


Fig. 4: Qualitative comparison of SMART with other baseline methods. The figure depicts predicted outputs plotted on left, center, right camera images and top view images of the scene. Note that for SMART, red, blue and green trajectories show outputs with different trajectory labels that captures different modes in the output.

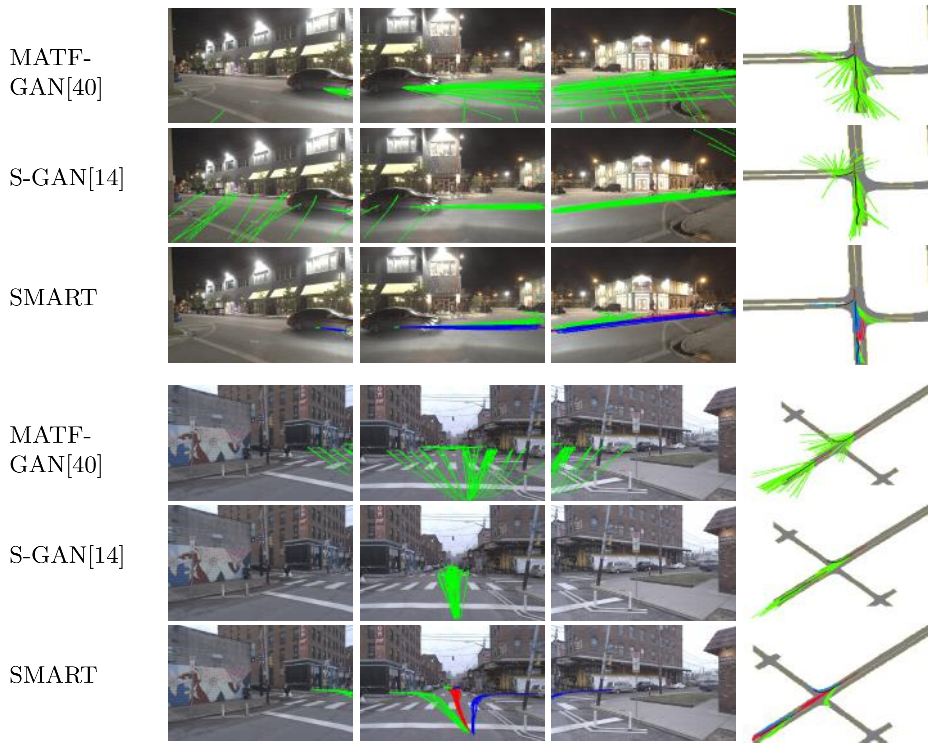


Fig. 5: Qualitative comparison of SMART with other baseline methods. The figure depicts predicted outputs plotted on left, center, right camera images and top view images of the scene. Note that for SMART, red, blue and green trajectories show outputs with different trajectory labels that captures different modes in the output.