SMART: Simultaneous Multi-Agent Recurrent Trajectory Prediction Supplementary Material

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

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

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 [3] 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.

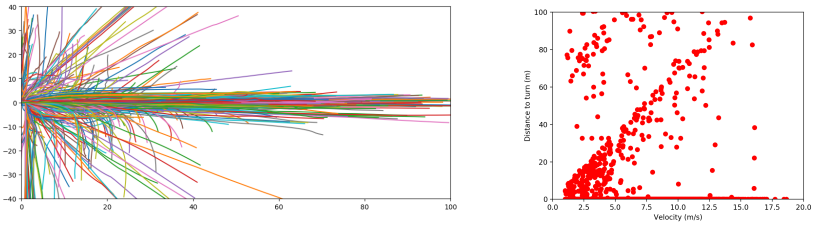

Fig. 1: **Left:** Shows several {*left,right and straight*} trajectories executed by vehicles in real world on Argoverse tracking (ArgoT) data sampled at 1Hz. **Right:** Velocity *vs* Distance to turn plot for tracklets in Argoverse tracking for the entire dataset. For plotting purposes we plot distance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

1.2 Low Level Controller:

Low-level controller simulates the desired behavior governed by vehicle dynamics module. It takes input from maneuver identification, IDM[3] and MOBIL[4], 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 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:

$$v_{lateral} = -kp_{lateral} * (x_{lateral} + \epsilon) \tag{1}$$

$$\psi_{req} = \arcsin(\frac{v_{lateral}}{v}) \tag{2}$$

$$\psi_{ref} = \psi_{future} + \psi_{req} \tag{3}$$

$$\dot{\psi} = kp_{heading} * \psi_{ref} \tag{4}$$

$$\phi = \arctan(\frac{L}{v}\dot{\psi}) \tag{5}$$

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 5 calculates the steering angle based on current velocity v, vehicle length L and heading rate $\dot{\psi}$.

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

1.3 Intelligent Driver Model:

Tab 1 shows values and sample space for parameters in Intelligent Driver Model[3]. We sample these parameters randomly to increase diversity of driving patterns.

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 [2] 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.

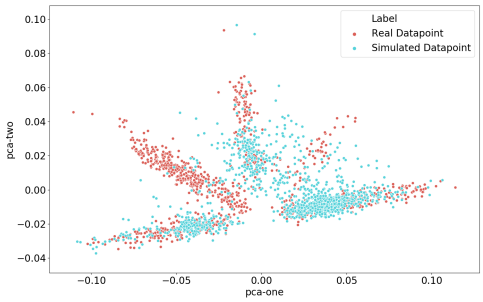

Fig. 2: Qualitative plot of PCA performed on 1000 random real and simulated trajectories from Argo Tracking

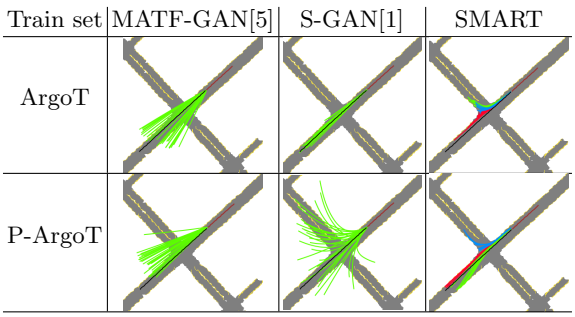

Fig. 3: Qualitative comparison of methods tested on Argo Tracking(ArgoT) with only training on ArgoT and training on P-ArgoT and fine tuning on ArgoT. Significant improvement in the diversity of prediction results can be observed in S-GAN[1] when the model is initialized using our simulated data.

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[1]. But it does not capture such diversity with the scene context 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 [P-ArgoT](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 [P-ArgoT](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.

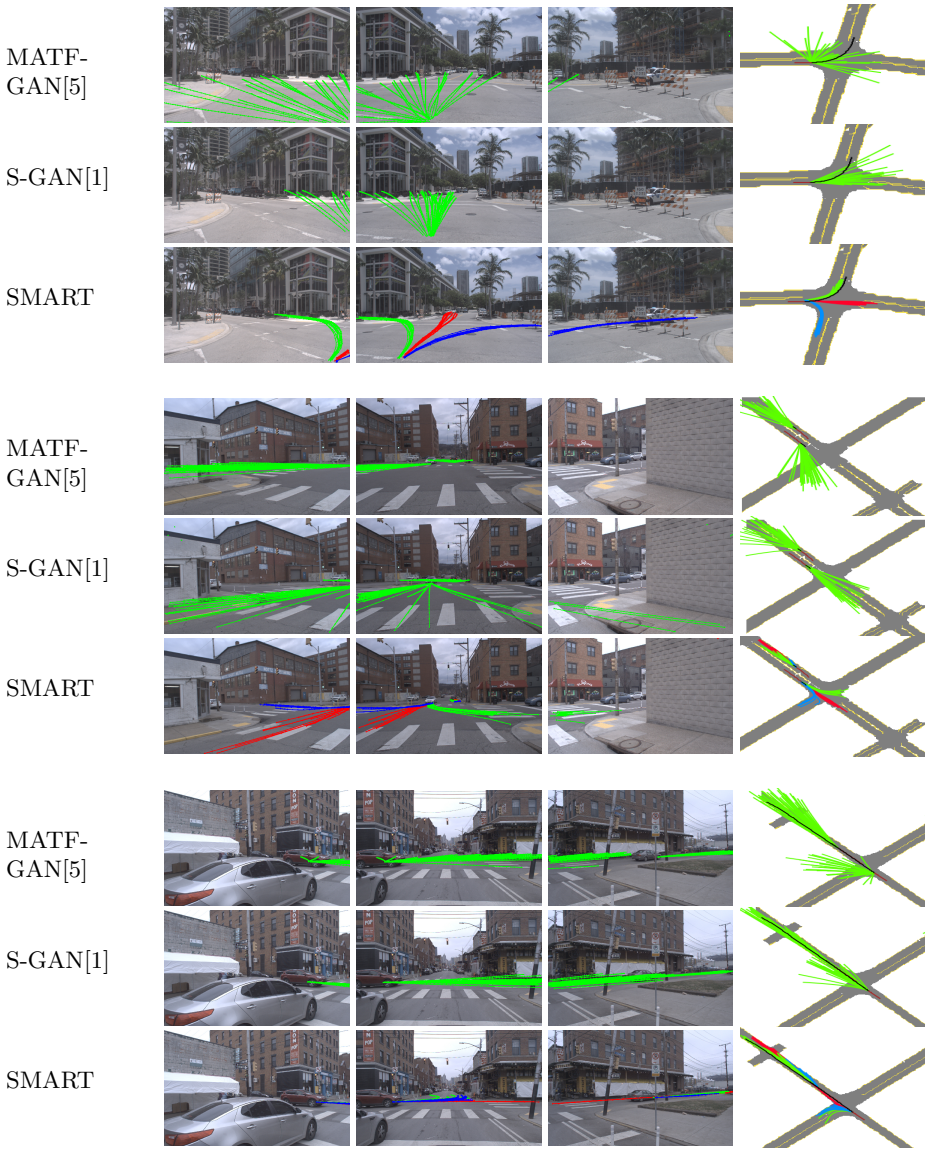

Fig. 4: Qualitative comparison of SMART with other baseline methods. The figure depicts predicted outputs plotted on left, center, right camera images and top view images of the scene. Note that for SMART, red, blue and green trajectories show outputs with different trajectory labels that captures different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

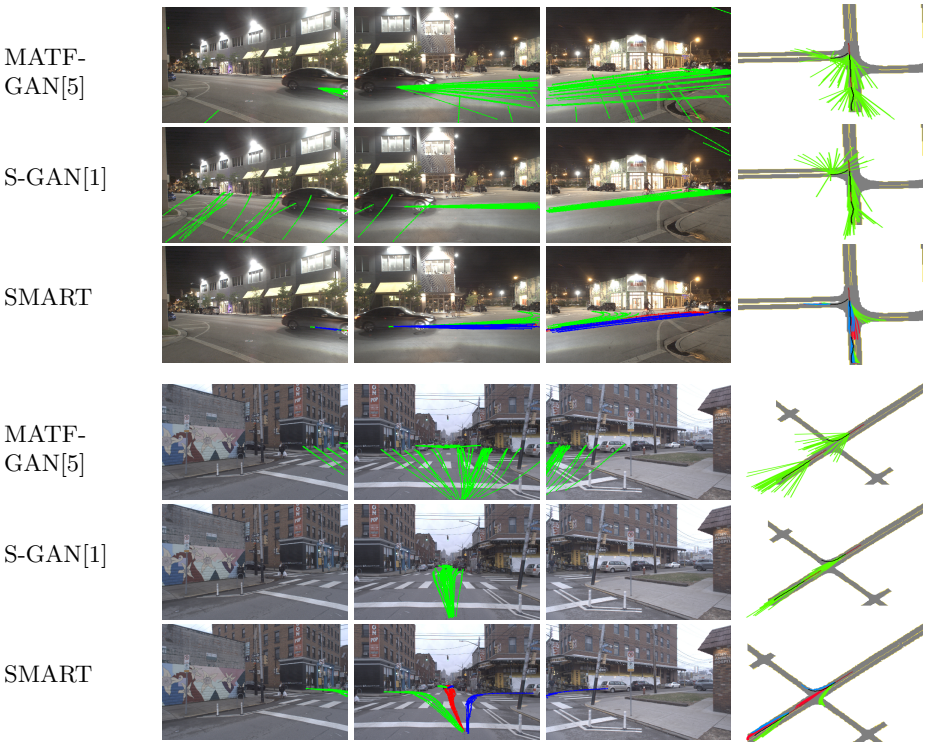

Fig. 5: Qualitative comparison of SMART with other baseline methods. The figure depicts predicted outputs plotted on left, center, right camera images and top view images of the scene. Note that for SMART, red, blue and green trajectories show outputs with different trajectory labels that captures different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

References

1. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social GAN: Socially acceptable trajectories with generative adversarial networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2255–2264 (June 2018). https://doi.org/10.1109/CVPR.2018.00240
2. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
3. 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)
4. 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)
5. 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)