Convolutional Spatial Fusion for Multi-Agent Trajectory Prediction

Tianyang Zhao
Peking University
Beijing, China
zhaotianyang@pku.edu.cn

Yifei Xu
University of California, Los Angeles
Los Angeles, CA
fei960922@ucla.edu

Mathew Monfort
MIT CSAIL
Cambridge, MA
mmonfort@mit.edu

Wongun Choi
ISEE
Cambridge, MA
wchoi@isee.ai

Chris Baker
ISEE
Pittsburgh, PA
chrisbaker@isee.ai

Yibiao Zhao
ISEE
Cambridge, MA
yz@isee.ai

Yizhou Wang
Peking University
Beijing, China
yizhou.wang@pku.edu.cn

Ying Nian Wu
University of California, Los Angeles
Los Angeles, CA
ywu@stat.ucla.edu

Abstract

Accurate prediction of other agents' trajectories is essential for mobile robot navigation. This is challenging because it requires the model to reason about agents' histories, social interaction among varying numbers and kinds of agents, and constraints from the scene context, and to account for the stochastic nature of human behaviors. We propose a spatial-centric approach to prediction by reasoning about these interactions and constraints jointly in a spatial grid. Specifically, the proposed model observes multiple agents' past trajectories and the scene, reasons about multiagent interactions via a novel convolutional spatial fusion module which retains the spatial structure of agents and the scene, and finally decodes recurrently to agents' future trajectories. Multiagent conditional generative adversarial training is used to learn multimodal, stochastic predictions. Experiments on both autonomous driving and pedestrian crowd datasets show that the model achieves state-of-the-art prediction accuracy.

1. Introduction

Accurate prediction of other agents' trajectories is essential for mobile robot navigation. Both low-speed vehicles and autonomous cars and trucks must predict the future motion of other vehicles, pedestrians, bicycles, etc. These motions depend on the scene context, and well as agents' goals and social interactions with other agents. Predictions must generalize to new situations, where the number and configuration of other agents are not fixed in advance. Deterministic trajectory prediction is insufficient due to the stochastic nature of human behaviors, so predicting a distribution over other agents' trajectories is needed.

Traditional trajectory prediction methods approach the problem of reasoning about social interaction by designing hand-crafted features [15]. Inverse optimal control methods also use hand-crafted cost features, but use learning to estimate linear weights to rationalize trajectories which are assumed to be generated by optimal control [19]. Recent data-driven approaches [1, 12, 30] capture interactive behavior by encouraging information sharing among agents via aggregation functions operating on agents' latent spaces. However, most of these models do not take scene context into consideration. Some recent approaches [21, 27] jointly reason about the scene context and social interaction, but these models also operate on agents' latent space and thus do not make use of spatial relationships among agents and between agents and scene contexts directly. These agent-centric models are also sensitive to the number of agents included in a scene, because information is generally aggregated within a predefined range of nearby agents.

As an alternative to these agent-centric approaches, we propose a novel spatial-centric approach to multiagent trajectory prediction. The model jointly predicts future trajectories of varying numbers of agents by modeling social interaction among them and constraints from the scene context spatially. This approach fuses information from multiple agents and the scene directly in a top-down view spatial

grid, and spatial locality is retained throughout the reasoning process.

Similar to recent data-driven models [1, 8, 9, 12, 13, 21, 27, 30, 31], we adopt an end-to-end deep learning approach. LSTM [17] encoders and decoders operate on individual agents to encode their past trajectories and to decode their future trajectories, and a spatial-centric, multiagent-scene fusion module based a convolutional architecture learns residuals [14] to agents' encoding vectors to model social interaction and context constraints spatially. Stochastic, multimodal predictions are achieved via multiagent conditional GAN [11, 23] training.

We conduct experiments on both autonomous driving datasets and a pedestrian crowd dataset. Experimental results are reported on the publicly available NGSIM US101 driving dataset [6], the publicly available Stanford Drone pedestrian crowd dataset [25], and a recently-collected autonomous highway driving dataset. Quantitative and qualitative ablative experiments are conducted to show the contribution of each part of the model, and quantitative comparisons with recent approaches show that the proposed approach achieves state-of-the-art accuracy in both autonomous driving and pedestrian trajectory prediction.

2. Related Work

Traditional methods for predicting or classifying trajectories model various kinds of interactions by hand-crafted rules or cost functions [3, 4, 5, 7, 15, 22, 32]. Recent data-driven methods based on recurrent networks [1, 8, 9, 12, 20, 21, 27, 28, 30] outperform traditional approaches. However, most of this work focuses either on modeling constraints from the scene context [28] or on modeling social interactions among multiple agents [1, 8, 9, 12, 30], while few models jointly consider both aspects [21, 27].

For neural-net-based approaches, modeling social interaction among varying numbers of agents is inherently difficult. This is because human interactive behavior is complex and stochastic, and also because neural nets prefer fixed shapes of parameters instead of varying ones. Thus, prediction models have to handle varying numbers of agents by parameter sharing among recurrent units for each agent, and by aggregation functions without parameters to fuse information from different agents. For instance, Social LSTM [1] uses max pooling over state vectors of nearby agents within a predefined distance range; consequently, influence from distant agents is ignored. Social GAN [12] explores max pooling of all the agents involved in a scene globally. However, this approach treats all the agents in a scene identically, regardless of their relative distance. Although these kinds of max pooling aggregation functions handle varying number of agents well, permutation invariant functions may discard information when input agents lose their uniqueness [27]. In contrast, Social Attention [30] and Sophie [27] address the heterogeneity of social interaction among different agents by attention mechanisms [2, 29] and spatial-temporal graphs [18]. Attention mechanisms encode which other agents are most important to focus on when predicting the trajectory of a given agent. However, the attention-based approach is very sensitive to the number of agents included, with $O(n^2)$ computational complexity to predict n agents.

The agent-centric approaches above either treat each agent identically or address which other agents are most important by attention mechanisms operating in agents' hidden state spaces. They generally do not make use of the spatial relationships among agents directly. Many of them do not take scene context into consideration, and for those that do, they do not retain the spatial structure of agents and the scene. As an alternative, the approach of Social Convolutional Pooling [8] partially retains this spatial structure by leveraging the strength of convolutional layers on spatial space. However, this approach does not reason about the scene context, and the trajectory of only one agent can be predicted with each forward pass – potentially too slow for real-time trajectory prediction of multiple agents.

Many data-driven approaches learn to predict deterministic futures of agents by minimizing reconstruction loss [1]. However, human behavior is inherently stochastic. Recent approaches address this by predicting a distribution over future trajectories either by combining Variational Auto-Encoders [10] and Inverse Optimal Control [21], or by conditional Generative Adversarial Nets [12, 27]. Other approaches predict future trajectories by conditioning on possible maneuver classes, to predict a set of possible trajectories instead of a deterministic one [8, 9]. Other work [20] uses generative adversarial imitation learning [16] to generate the dynamics of a bicycle model of vehicle motion.

3. Method

The trajectory prediction problem is formulated as follows. Given a scene containing a static context and multiple dynamic interacting agents, the top-down view representation of the static scene context is denoted as c, which can be either a segmented image containing all static objects, or a birds-eye view raw image. The number of agents n varies in different scenes. Past trajectories of the agents are denoted $\{x_1, x_2, .., x_n\}$, where x_i is the i-th agent's history over past time period T: $x_i = [x_{i1}, x_{i2}, .., x_{iT}]$, and x_{it} denotes the state of agent i at past time stamp t, *e.g.*, in the most simple case its coordinate at t. The objective is to predict the future trajectories of these agents jointly $\{y_1, y_2, .., y_n\}$ over future period T' given all the information above, where y_i is the future trajectory of the i-th agent: $y_i = [y_{iT+1}, y_{iT+2}, .., y_{iT+T'}]$.

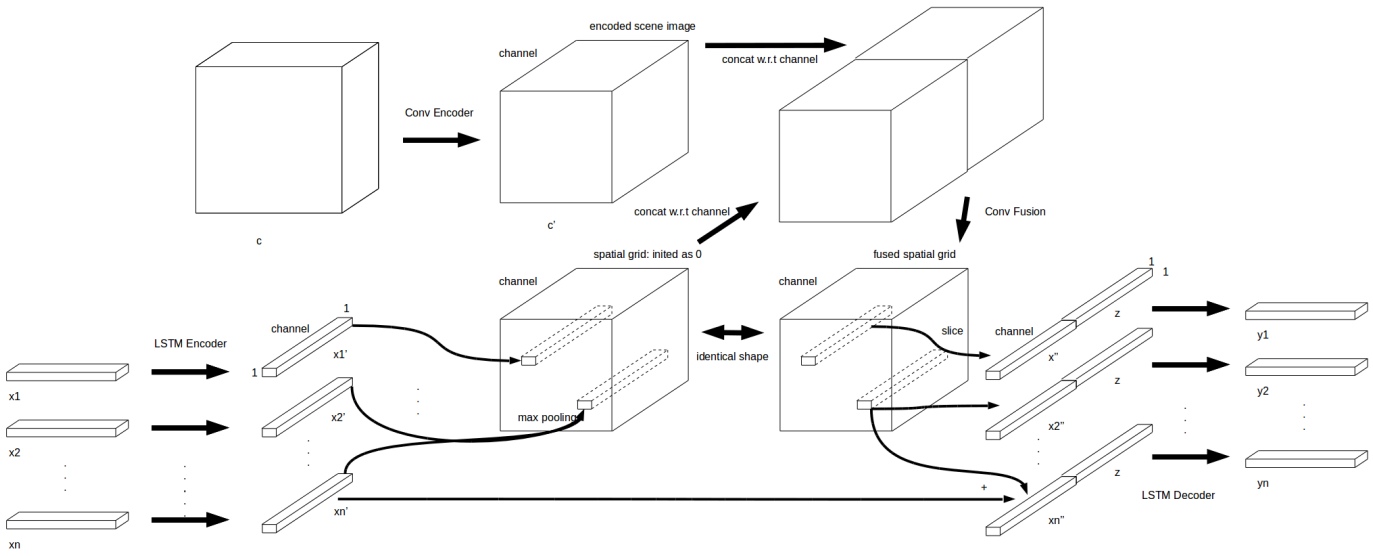

Figure 1. Overall illustration of the Multi-Agent Scene Spatial Fusion architecture. The inputs to the network are n agent trajectories $\{x_1, x_2, .., x_n\}$ over past time period T (bottom left), and a top-down view static scene context image c (top left). c and each x_i are then encoded independently through convolutional and recurrent encoding streams, respectively. Then, the top-down view encoded scene context c' and encoded agent vectors $\{x'_1, x'_2, .., x'_n\}$ from the two streams are fused spatially to form a spatially arranged grid of the features of both the scene and multiple agents. In this step, $\{x'_1, x'_2, .., x'_n\}$ are placed into the grid with respect to their scaled, discretized top-down view coordinates $\{x_{iT}, x_{2T}, .., x_{nT}\}$. Subsequently, the convolutional fusion module is applied on top of these features to reason about the interactions while retaining spatial locality. Fused vectors for each agent $\{x''_1, x''_2, .., x''_n\}$ are then sliced out from c'', which contain the interaction features for the corresponding agent. These fused vectors are then added to the original encoded vectors of the corresponding agents $\{x'_1 + x''_1, x'_2 + x''_2, .., x'_n + x''_n\}$ as residuals, and then get decoded independently to future trajectory predictions $\{\hat{y}_1, \hat{y}_2, .., \hat{y}_n\}$ (bottom right). The whole architecture is fully differentiable and is trained end-to-end.

3.1. Single-Agent Trajectory Encoder-Decoder

A single agent trajectory Encoder-Decoder is built as a baseline model. The model learns to regress the future trajectory y_i of a given agent i only from its own past trajectory x_i without information about other agents or the static scene context c. In this model, x_i is encoded by an LSTM encoder, and the encoding x'_i is then decoded to $\hat{y}_i$ as a prediction of ground truth y_i. Reconstruction loss between y_i and $\hat{y}_i$ is optimized. The network learns a mean deterministic policy. The trained network is used to initialize the parameters for Section 3.2.

3.2. Multi-Agent Scene Spatial Fusion

In this section, we propose a novel spatial-centric fusion approach for jointly modeling social interaction among multiple agents and constraints from the scene context. Figure 1 gives an overall illustration of the proposed Multi-Agent Scene Spatial Fusion architecture.

There are two paralleled encoding streams in the model. One encodes the static scene context image c using CNNs, and another encodes past trajectories of each individual agent x_i independently using the LSTM Encoders described in Section 3.1. Each LSTM Encoder shares the same set of parameters, so the architecture is insensitive to varying numbers of agents in different scenes. The output of the scene context encoder CNNs is a scaled feature map c' retaining the spatial structure of the top-down scene context image, and the outputs of the LSTM encoders are 1-d agent state vectors $\{x'_1, x'_2, .., x'_n\}$ without temporal structure.

Next, the two encoding streams are fused spatially into a top-down view discrete spatial grid in the following way: agent encodings $\{x'_1, x'_2, .., x'_n\}$ are placed into one discrete top-down view spatial grid, which is initialized as 0 and is of the same shape (width and height) as the encoded scene image c'. The dimension axis of the encodings fits into the channel axis of the grid as shown in Figure 1. The agent encodings are placed into the spatial grid with respect to their positions at the last time stamp of their past trajectories x_{iT}, respectively. This grid is then concatenated with the encoded scene image in the channel dimension to get a combined spatial grid. If multiple agents are placed into the same cell in the spatial grid due to discretization, element-wise max pooling is performed. This spatial fusing method ensures that the architecture is not sensitive to varying numbers of agents in different scenes.

This combined spatial grid is fed into fully convolutional

layers, which learn to represent interactions among multiple agents and between agents and the scene context, while retaining spatial locality. Specifically, these layers operate at multiple spatial resolution scale levels by adopting U-Net-like architectures [26] to model interaction at different spatial scales. The output feature map of this fusion model c'' shares exactly the same shape with c' in width and height to retain the spatial structure.

Subsequently, agent-specific representations with fused interaction features for each agent $\{x_1'', x_2'', .., x_n''\}$ are sliced out according to their coordinates $\{x_{iT}, x_{2T}, .., x_{nT}\}$ from the fused spatial grid output c''. These agent-specific representations are then added as a residual [14] to the original encoded agent vectors to form fused agent vectors $\{x_1' + x_1'', x_2' + x_2'', .., x_n' + x_n''\}$, which encode all the information from the past trajectories of the agents themselves, the static scene context, and the interaction features among multiple agents. Compared with Social LSTM [1] and Social GAN [12], where multiple agents get identical social embeddings, our approach allows each agent to get a different social and context embedding focused on itself. Furthermore, the fusion model gets these embeddings for multiple agents using shared feature extractors instead of operating n times for n agents. This is similar to Faster R-CNN's [24] feature extractor approach for feeding the feature map into the Region Proposal Network.

Finally, for each agent in the scene, its fused vector $x_i' + x_i''$ is decoded to future trajectory prediction $\hat{y}_i$ by the LSTM Decoders described in Section 3.1. Similar to the encoders for each agent, parameters are shared to guarantee that the network can generalize well when the number of agents in the scene varies.

The whole architecture is fully differentiable and is trained end-to-end to minimize reconstruction loss between $\{\hat{y}_1, \hat{y}_2, .., \hat{y}_n\}$ and $\{y_1, y_2, .., y_n\}$.

3.3. Multi-Agent Spatial GAN

To learn multimodal stochastic predictions, we use multiagent conditional generative adversarial training [11, 23] based upon the Multi-Agent Scene Spatial Fusion model described above in Section 3.2.

GANs consist of two networks, a generator G and a discriminator D competing against each other. G learns the distribution of the data and generates samples, while D learns to distinguish the feasibility or infeasibility of the generated samples. These networks are simultaneously trained in a two player min-max game framework.

In our setting, we use a conditional G to generate future trajectories of multiple agents jointly, conditioning on all the agents' past trajectories, the static scene context, and random noise input to create stochastic outputs. Simultaneously, we use D to distinguish whether the generated paths are real (ground truth) or fake (generated). Both G and D share exactly the same architecture in their encoding parts with the deterministic model presented in Section 3.2, to reason about static scene context and interaction among multiple agents spatially. Both G and D are initialized with parameters from the trained deterministic model introduced in Section 3.2. Detailed architectures and losses are introduced below.

Generator (G) G observes past trajectories of all the agents in a given scene $\{x_1, x_2, .., x_n\}$, and the static scene context c. It jointly outputs the predicted future trajectories $\{\hat{y}_1, \hat{y}_2, .., \hat{y}_n\}$ by decoding the fused agent vectors from the multiagent scene spatial fusion model, concatenated with Gaussian white noise vector z. Its architecture is exactly the same as presented in Section 3.2, except that in the deterministic model, the fused encoding for a given agent $x_i' + x_i''$ is concatenated with $z = 0$ vector to decode into its future trajectory; while in G, z is sampled from a Gaussian distribution.

Discriminator (D) D observes either all generated or all ground truth past and future trajectories of the agents $\{x_1, x_2, .., x_n, \hat{y}_2, .., \hat{y}_n\}$ or $\{x_1, x_2, .., x_n, y_1, y_2, .., y_n\}$ in a given static scene context c. It outputs real or fake labels for the future trajectory of each agent in the scene $\{\hat{l}_1, \hat{l}_2, .., \hat{l}_n\}$. It shares nearly the same architecture as presented in Section 3.2, except for the following differences: (1) Its agents' LSTM encoders take in past and future trajectories as input instead of just past trajectories; (2) As a classifier, it does not use the LSTM decoders to decode x_i', x_i'' to future trajectories. Instead, the agent fused encodings x_i', x_i'' are fed into fully connected layers to be classified as a real or a fake trajectory of agent i: l_i.

Losses To train the Multi-Agent Spatial GAN, we use the following losses:

$$W^* = \arg\min_W \mathbb{E}_{scene}[\mathcal{L}_{GAN}(\{\hat{l}_i, l_i\}_{i \in scene}) + \lambda \sum_{i \in scene} L_{L2/L1}(\hat{y}_i, y_i)], \tag{1}$$

where W is the set of weights of the model and λ is a regularizer between adversarial loss and reconstruction loss. The adversarial loss L_{GAN} is:

$$\mathcal{L}_{GAN}(\{\hat{l}_i, l_i\}_{i \in scene}) = \min_G \max_D \sum_{i \in scene} \mathbb{E}_{traj_i \sim p(y_i)}[l_i \log \hat{l}_i] + \mathbb{E}_{traj_i \sim p(\hat{y}_i)}[(1 - l_i) \log(1 - \hat{l}_i)]. \tag{2}$$

4. Experiments

In the Experiments and Results sections, we evaluate our model on both driving datasets [6] and a pedestrian crowd dataset [25]. We construct different variants of our models for ablative studies, and compare with state-of-the-art alternative methods [1, 7, 8, 12, 15, 20] quantitatively. Qualitative results are also presented for further analysis.

4.1. Datasets

We use the publicly available NGSIM US101 dataset [6], the publicly available Stanford Drone Dataset [25], and a recently collected highway driving dataset for training and evaluation.

NGSIM. NGSIM US101 dataset [6] is a driving dataset consisting of trajectories of real freeway traffic over a time span of 45 minutes. It is recorded by fixed birds-eye view cameras placed over a 640-meter span of US101. Trajectories of all the vehicles traveling through the area within this 45 minutes are annotated. The dataset consists of various traffic conditions (mild, moderate and congested), and contains around 6k vehicles in total.

Autonomous Driving. NGSIM contains rich vehicle-vehicle interactions. However, the recorded highway span is quite straight, so minimal agent-scene interaction is observed. As an alternative, we analyze a large-scale dataset gathered by an autonomous vehicle during highway driving, including a several-mile stretch of highway, with curved lanes, highway exits and entrances. Ablative studies are conducted for this dataset to show our model's ability to model agent-scene and agent-agent interactions, respectively.

Stanford Drone. Stanford Drone dataset is a large-scale pedestrian crowd dataset [25] consisting of a top-down view of 20 unique scenes in which pedestrians, bicyclists, skateboarders, carts, cars and buses navigate on a university campus. Raw, static scene context images are provided from a top-down view, and coordinates of multiple agents' trajectories are provided in pixels. These scenes contain rich human-human interactions, often in the form of high density crowds, and diverse physical landmarks such as buildings and roundabouts that must be avoided. We crop the videos into around 16k scenes, and we use the standard test set for quantitative evaluation. Also note that some scenes from the standard training set are not used for our training process, but leaved for qualitative evaluation instead.

4.2. Baseline Models

We compare quantitatively with the following published baseline approaches, and we also construct a set of variants of our model for ablative studies.

Baselines:

CV: A simple constant velocity Kalman filter.

C-VGMM + VIM [7]: A recent vehicle interaction approach based on variational Gaussian mixture models with Markov random fields.

GAIL-GRU [20]: State-of-the-art vehicle trajectory prediction model. It uses GRU-based GAIL [16] to generate the dynamics of a bicycle model of vehicle motion. In this approach, trajectories are generated one vehicle at a time, and the model assumes access to the ground truth trajectories of nearby vehicles over the prediction horizon, which other approaches, including our, do not.

Social Conv [8]: State-of-the-art vehicle trajectory prediction approach based on maneuver modeling. This model uses extra supervision of maneuver labels for a two-step training procedure to model the latent distribution space for future trajectories. They also evaluate on NGSIM [6], so we design our experimental setting identically and directly compare with their reported quantitative results.

Social Force [15]: Pioneering traditional feature-based pedestrian trajectory prediction model.

Social LSTM [1]: State-of-the-art deterministic pedestrian trajectory prediction model.

Social GAN [12]: State-of-the-art stochastic pedestrian trajectory prediction model. They report the error of the best sample among N generated samples for stochastic prediction evaluation. We also report this for fair comparison with our stochastic models.

Desire [21]: State-of-the-art stochastic pedestrian trajectory prediction model. They use Variational Auto-Encoders [10] and Inverse Optimal Control to generate and rank trajectories.

Sophie [27]: A complex, state-of-the-art stochastic attention-based prediction model. They use the same evaluation metrics as Social GAN [12], and they evaluate on the same pedestrian crowd dataset [25] as we do here. We design our experimental setting identically, and compare with their reported quantitative results.

Variants:

LSTM: A simple LSTM Encoder-Decoder as introduced in Section 4.1.

Single Agent Scene: The model shares exactly the same architecture as introduced in Section 3.2, except that it only takes in one agent history x_i with scene representation c and outputs only $\hat{y}_i$ each time, so the model reasons about scene-agent interaction, but is completely unaware of multi-agent interaction.

Multi Agent: The model has the same details as the described model in Section 3.2, except that scene representation c is not provided as input. The model only reasons about multi-agent interaction absent from scene context information.

Multi Agent Scene: The introduced deterministic model in Section 3.2.

Individual D: The stochastic model which shares the same architecture with the GAN introduced in Section 3.3, except that its D only takes in each agent's individual encoding x_i' instead of $x_i' + x_i''$ for classification.

Joint D: The stochastic model introduced in Section 3.3. Similar to Social GAN [12], we sample N times and report the best trajectory in the L2 sense for fair comparison with stochastic models, with $N = 3$ in Section 5.1, and $N = 20$ as adopted by [12] in Section 5.2.

See Supplementary Materials for implementation details.

5. Results

5.1. Driving Datasets

We conducted experiments on the publicly available NGSIM US101 [6] dataset. We adopt the same experimental setting as reported in [8]: We split the trajectories into segments of 8s, and all agents appearing in the 640-meter span are considered in the reasoning and prediction process. We use 3s of trajectory history and a 5s prediction horizon. LSTMs operate at 0.2s per timestep. As in [8], we report the RMSE in meters with respect to each timestep t within the prediction horizon:

$$RMSE(t) = \sqrt{\frac{1}{n}\sum_{i=1,2,..,n}((\hat{x}_{it}-x_{it})^2+(\hat{y}_{it}-y_{it})^2)}, \quad (3)$$

where n is the total number of agents in the validation set, x_{it} denotes the x coordinate of the i-th car in the dataset at future timestamp t, and y_{it} the y coordinate respectively.

RMSE	1s	2s	3s	4s	5s
CV [8]	0.73	1.78	3.13	4.78	6.68
Our LSTM	0.66	1.62	2.94	4.63	6.63
C-VGMM + VIM [7]	0.66	1.56	2.75	4.24	5.99
GAIL-GRU [20]	0.69	1.51	2.55	3.65	4.71
Our Multi Agent	0.67	1.51	2.51	3.71	5.12
Social Conv [9]	**0.61**	**1.27**	2.09	3.10	4.37
Our Joint D	0.66	1.34	**2.08**	**2.97**	**4.13**

Table 1. Quantitative results on NGSIM [6] dataset. RMSEs in meters with respect to each future timestep in the prediction horizon are reported.

Quantitative results are shown in Table 1. This result shows that our deterministic model *Multi Agent* achieves the state-of-the-art accuracy as other deterministic models *C-VGMM + VIM, GAIL-GRU*. Note that *GAIL-GRU* has access to the future gound-truth of other agents while predicting a given agents, but we do not have access to it. The result also shows compared with approaches that capture distribution or maneuvers of future trajectories *Social Conv*, our stochastic model *Joint D* also performs at the state-of-the-art level. Note that *Social Conv* have access to auxiliary supervision of maneuver labels, while we do not encourage this. Lanes in NGSIM dataset are quite straight, so little agent-scene interaction is included, and our *Multi Agent Scene* model does not outperforms *Multi Agent*.

We also explore our Autonomous Driving dataset, where more curved lanes and more complex static scene contexts are included. We report the MAE in meters with respect to each timestep t within the prediction horizon:

$$MAE(t) = \frac{1}{n}\sum_{i=1,2,..,n}\sqrt{(\hat{x}_{it}-x_{it})^2+(\hat{y}_{it}-y_{it})^2} \quad (4)$$

Quantitative ablative results are shown in Figure 3, and some interesting qualitative examples are shown in Figure 2. Quantitative results show that both *Single Agent Scene* and *Multi Agent* outperforms *LSTM* baseline, and that *Multi Agent* consistently outperforms *Single Agent Scene* and *Multi Agent*, and comparison between *Multi Agent* and *Single Agent Scene* shows that the former one performs better in the short term while the latter one performs better in the longer term. Qualitative results give reason to these observations.

Left 5 figures in Figure 2 shows a typical agent-scene interaction scenario. In this scene, the ground truth future of the two cars are on a exit of a highway driving along a curved lane. Without multi-agent or scene information, the predicted results from *LSTM* learns a strategy of going straight out of the lane; while all the other 3 models follow lane in different accuracy levels. With scene context information, *Single Agent Scene* and *Multi Agent Scene* learn to follow the lane as expected. However, we notice that *Multi Agent* can also follow the lane in a shorter future although finally fails in distant future. This may because the following car can follow the leading car's past trajectory which is an approximation to the lane shape, and because the followed car can reason the lane shape by the relative position of multiple cars and follow an 'imaginery' lane. This lane may be correct in the near future, but may diverge from reality in the distant future, where curvature of the lane changes. This explains why *Single Agent Scene* outperforms *Multi Agent* in the distant future.

While, the rightmost figure shows a typical agent-agent interaction scenario. The leading car in the middle is traveling slower than the following car, and the *Multi Agent* model successfully predicts a taking-over of the leading car, and results in its changing lane behavior. This kind of changing lane behavior promote the prediction accuracy from the start of future time span, and this explains why *Multi Agent* outperforms *Single Agent scene* in the near future.

Overall from these ablative studies, we conclude that our spatial fusion model successfully models agent-agent and agent-scene interaction. More specifically, the scene fusion model learns constraints from scene context, and the multi-agent model learns multi-agent interaction. These interaction have different types, including following behaviors, spatial reasoning behaviors, and competing behaviors.

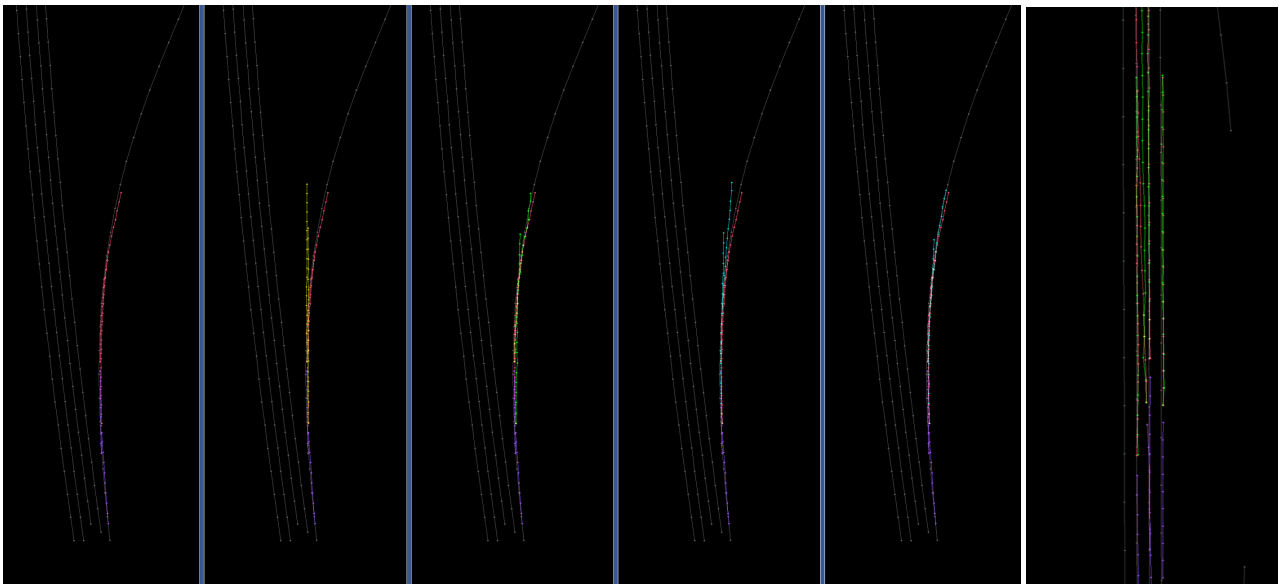

Figure 2. Qualitative results from Autonomous Driving dataset. The left 5 figures are extracted from one scene for ablative study. In these 5 figures, the grey lines indicate lane centers, purple past trajectories, red ground truth future trajectories. From left to right shows predicted results of ground truth (red), *LSTM* (yellow), *Single Agent Scene* (green), *Multi Agent* (blue), and *Multi Agent Scene* (blue). The right most figure shows an interesting prediction result of the *Multi Agent* model. In this case, the following car in the middle is taking over the leading car. All results are from validation set.

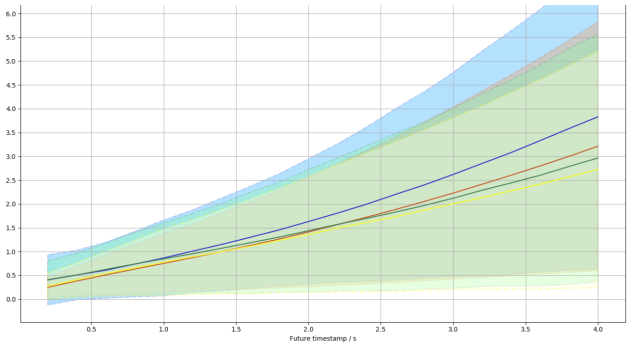

Figure 3. Quantitative results on Autonomous Driving dataset. MAEs in meters with respect to each future timestep in the prediction horizon are reported. Blue line is the evaluation result of *LSTM*, red for *Multi Agent*, green for *Single Agent Scene*, and yellow for *Multi Agent Scene*.

5.2. Pedestrian Dataset

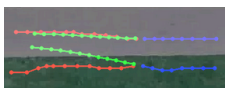

Figure 4. A collision avoidance example predicted by *Multi Agent* in Stanford Drone [25] dataset. Refer to Figure 5 for legends.

We also explore a pedestrian dataset, the publicly available Stanford Drone [25] dataset. We adopt the same experimental setting as reported in [27]: We split the trajectories into segments of 8s, and all agents appearing in the scene are considered in the reasoning and prediction process. We use 3.2s of trajectory history and a 4.8s prediction horizon. LSTMs operate at 0.4s per timestep. As in [25], we report the ADE and FDE in pixels with respect to each timestep t within the prediction horizon:

$$
\begin{aligned}
ADE(i) &= \frac{1}{t} \sum_{j=1,2,..,T'} \sqrt{(\hat{x}_{ij} - x_{ij})^2 + (\hat{y}_{ij} - y_{ij})^2} \\
ADE &= \frac{1}{n} \sum_{i=1,2,..,n} ADE(i) \\
FDE(i) &= \sqrt{(\hat{x}_{iT'} - x_{iT'})^2 + (\hat{y}_{iT'} - y_{iT'})^2} \\
FDE &= \frac{1}{n} \sum_{i=1,2,..,n} FDE(i)
\end{aligned}
\tag{5}
$$

where n is the total number of agents in the validation set, x_{ij} denotes the x coordinate of the i-th car in the dataset at future timestamp j, and y_{ij} the y coordinate respectively, T' denotes final future timestamp. Quantitative results show below in Table 2. Note that the 5 models on the top are deterministic ones, while the bottom 5 capture distribution. The proposed $MultiAgentScene$ outperforms other deterministic model in ADE, and the propsed $JointD$ performs at the state-of-the-art level.

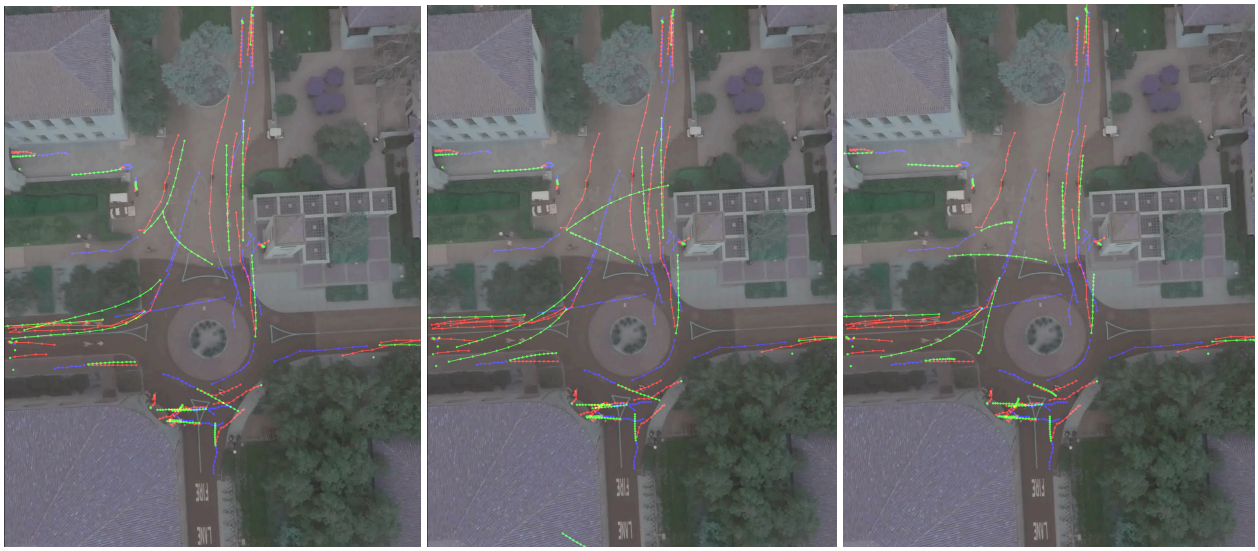

Figure 5. Ablative qualitative results on Stanford Drone [25] dataset. From left to right are results from *Multi Agent Scene*, *Multi Agent*, and *LSTM*. Blue lines show past trajectories, red ground truth, and green predicted. All results come from the qualitative validation dataset.

Method	ADE	FDE (4.8s)
Our LSTM	37.35	77.13
Social Force [15]	36.38	58.14
Social LSTM [1]	31.19	56.97
Our Multi Agent	30.75	65.90
Our Multi Agent Scene	28.49	61.07
Social GAN [12]	27.25	41.44
Our Individual D	25.81	39.72
Our joint D	23.05	34.65
Desire [21]	19.25	34.05
Sophie [27]	**16.27**	**29.38**

Table 2. Quantitative results on Stanford Drone [25] dataset. Average and Final Displacement Errors are reported.

Figure 5 shows ablative qualitative results. For example, notice that two agents traveling from top are making right turns to the left, which, given their past trajectories, scene contexts and other agents' past, can be inferred. This intention is successfully captured by *Multi Agent Scene*, and *Multi Agent* model also inferred partially from the information of other agents. However, without scene information, the predicted trajectories violate scene constraints: they are traveling in the left of the road instead of in the right. In the contrast, although one of the predicted agents *LSTM* makes a right turn following the agent's own past curvature, most of the predicted behavior make little sense.

6. Discussion

We proposed an architecture for trajectory prediction which models scene context constraints and social interaction in a spatial-centric representation, rather than the agent-centric approach more commonly used in the literature. Our motivation was that scene context constraints and social interaction patterns are invariant to the absolute coordinates where they take place; these patterns only depend on the relative positions among agents and scenes. Convolutional layers are suited to modeling these kinds of position-invariant spatial interactions by sharing parameters across space, while recent approaches like Social Pooling [1, 12] or Attention mechanisms [30] cannot explicitly reason about spatial relationships among agents and cannot reason about these relationships at multiple spatial scales. Our spatial-centric fusion module with multi-grid convolutional layers models this ideally. Our spatial fusion approach, to our best knowledge, is the first approach which fuses information from a static scene context with multiple dynamic agent states and retains the their spatial structure throughout the reasoning process for trajectory prediction.

We applied our model to two different trajectory prediction tasks to demonstrate its flexibility and capacity to learn different types of behaviors, agents, and scenarios from data. In the vehicle prediction domain, our model achieved state-of-the-art results at long-range prediction of vehicle trajectories in the NGSIM dataset. Our adversarially trained stochastic prediction model performed best relative to the maneuver-based approach of [8], suggesting that the representation of the distribution over maneuvers was necessary – whether implicit or explicit. Our ablative studies on our private highway driving dataset showed that representations of both the scene and multiagent interactions were necessary for accurate trajectory prediction for highway driving, which typically involves more complex scene context than

NGSIM (greater lane curvature, more entrances and exits, etc.).

Our application to a state-of-the-art pedestrian dataset [25] demonstrated comparable performance with previously published results. Although some recent models achieved greater accuracy than ours [27, 21], all used dramatically different architectures; it is interesting to find that a novel spatial-centric architecture can also achieve a high standard of performance. Future work should examine the factors that influence performance, and the advantages and disadvantages of different architectures.

In future work, we plan to integrate explicit maneuver representations using an auxiliary maneuver classification task. We hope that this will close the gap in performance between our deterministic model, and the model in [8], and also increase the interpretability of our predictions.

Social trajectory prediction is a complex task, which depends on the ability to extract structure from the history of agents' joint motions. Our goal here has been to explore the contribution of spatial-centric representations to trajectory prediction. Though our ultimate goal is accurate trajectory predictions, our hope is that the process of engineering better models will continue to yield further insights into the structure of human interaction.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 4, 5, 8

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*. 2015. 2

[3] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr. A combined model and learning based framework for interaction-aware maneuver prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2016. 2

[4] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. *Computer VisionECCV*, 2012. 2

[5] W. Choi and S. Savarese. Understanding collective activities of people from videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1242–1257, 2014. 2

[6] J. Colyar and J. Halkias. Us highway dataset. Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030. 2, 4, 5, 6

[7] N. Deo, A. Rangesh, and M. M. Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *arXiv:1801.06523*, 2018. 2, 4, 5, 6

[8] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *IEEE Computer Vision and Pattern Recognition Workshop on Joint Detection, Tracking, and Prediction in the Wild*, 2018. 2, 4, 5, 6, 8, 9

[9] N. Deo and M. M. Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *IEEE Intelligent Vehicles Symposium (IV)*, 2018. 2, 6

[10] K. Diederik and W. Max. Auto-encoding variational bayes. In *ICLR*. 2014. 2, 5

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. 2014. 2, 4

[12] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 4, 5, 8

[13] I. Hasan, F. Setti, T. Tsesmelis, A. Del Bue, F. Galasso, and M. Cristani. MX-LSTM: mixing tracklets and vislets to jointly forecast trajectories and head poses. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2018. 2

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 2, 4, 10

[15] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 1, 2, 4, 5, 8

[16] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016. 2

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, pages 1735–1780, 1997. 2

[18] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018*, pages 5308–5317, 2016. 2

[19] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012. 1

[20] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating driver behavior with generative adversarial networks. *Intelligent Vehicles Symposium (IV)*, 2017. 2, 4, 5, 6

[21] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2165–2174, 2017. 1, 2, 5, 8, 9

[22] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. *IEEE Transactions on CVPR*, 2009. 2

[23] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arxiv:1411.1784*, 2014. 2, 4

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. 4

[25] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory prediction in crowded scenes. *European Conference on Computer Vision (ECCV)*, 2016. 2, 4, 5, 7, 8, 9

[26] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 4, 10

[27] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, and S. Savarese. Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. *arXiv, CoRR*, abs/1806.01482, 2018. 1, 2, 5, 7, 8, 9

[28] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese. Car-net: Clairvoyant attentive recurrent network. *arXiv:1711.10061*, 2017. 2

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. 2

[30] A. Vemula, K. Muelling, and J. Oh. Social attention: Modeling attention in human crowds. In *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018*, May 2018. 1, 2, 8

[31] Y. Xu, Z. Piao, and S. Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2018. 2

[32] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? *IEEE Transactions on CVPR*, 2011. 2

7. Supplementary Material: Implementation Details

This section presents learning details for the Convolutional Spatial Fusion architecture.

For the image encoder, we use shallow CNNs for the driving datasets, considering that static scene context images from these datasets have already been annotated semantically; while we use a U-Net-like [26] architecture which includes a pretrained ResNet [14] as image encoder for Stanford Drone dataset, considering that raw images are fed. Architecture of the shallow CNNs is presented as follows. Input images are resized into $3 \times 60 \times 60$ (channel $\times$ width $\times$ height) and then go through 3 Convolutional layers with kernel sizes of $3, 4, 5$, filters $16, 16, 32$, paddings $1, 2, 2$ respectively, all with ReLU activations and Batch Normalization. Max Pooling of kernel size 2, stride 2 is operated after the second Convolutional layer. Architecture for the ResNet encoder is presented as follows. A pretrained ResNet18 on ImageNet is included. Output feature maps from the 2nd Convolutional module is concatenated with up-scaled output feature maps from the 3-rd and the 4-th Convolutional modules, where the up-scaling layers are Transpose Convolutional layers of filters $128, 128$, kernel sizes $3, 7$, strides $2, 4$, paddings $1, 3$, output paddings $1, 3$ respectively, all with ReLU activations and Batch Normalization. The concatenated output is then upsampled to shape $384 \times 30 \times 30$, and go through a 1×1 Convolutional bottleneck to reduce its channels to 32. The output feature map of this scene encoding module, either the shallow encoder or the ResNet one, is of shape $32 \times 30 \times 30$.

For the agent encoder, one-layer LSTMs of hidden state dimensions 32 are used. The input x, y coordinates are processed to be $\Delta x, \Delta y$, and are then embedded linearly to dimension 32 as inputs to the LSTMs. Dropout ratio in LTSMs are set to be 0.3 for training. The state tuples of the LSTMs are initialized as 0. Individual encoding of a given agent is its final state vector h of shape 32 of the LSTM.

The Multi-Agent Scene Spatial Fusion module operates at a resolution level of 30×30 with a U-Net-like architecture. The encoded agent grid and the encoded context grid are concatenated in their channel dimension to form a spatial grid of shape $64 \times 30 \times 30$. This input grid goes into 3 Convolutional layers sequentially of filters $32, 32, 32$, kernel sizes $3, 3, 4$, strides $1, 1, 1$, paddings $1, 1, 1$. After the first and the second layer, Max Pooling of kernel sizes $2, 2$ and strides $2, 2$ are performed respectively. The final output of this fusion module is the sum of 3 features maps: the output of the 1st Convolutional layer is added together with up-scaled feature maps of the 2nd and the 3rd feature maps. The up-scalings are performed with a Transpose Convolutional layer of kernel size 4, stride 2, padding 1, and an Upsampling layer of scale factor 5 respectively. All Convolutional and Transpose Convolutional layers operate ReLU activations and Batch Normalization. The output fused feature map is of shape $32 \times 30 \times 30$.

For the agent decoder, one-layer LSTMs of hidden state dimension 48 are used. Its state tuples of the LSTMs are initialized in the following way: h is initialized as 0, and c is initialized as $x_i' + x_i''$ (See Section 3.2) concatenated with a 16-dim z. z is 0 for all deterministic models and is sampled from Gaussian distribution for all stochastic models. The decoder decodes LSTM outputs h_t to $\Delta\hat{x}_t, \Delta\hat{y}_t$ linearly. These predicted $\Delta x_t, \Delta y_t$ are embedded linearly to dimension 48 as inputs to the LSTMs on future timestamps iteratively. $\Delta\hat{x}_t, \Delta\hat{y}_t$ are processed to form $\hat{x}_t, \hat{y}_t$ at last for calculating losses. Dropout ratio in LTSMs are set to be 0.3 for training.

The classifier of the discriminator is an MLP with a hidden layer size 512 and Leaky ReLU activation. Sigmoid activation is operated at last for $0 - 1$ classification.

We schedule the learning process as follows: only train *LSTM*, then train *Multi Agent* or *Single Agent Scene* from pretrained parameters of *LSTM* to learn residuals, then *Multi Agent Scene*, and finally *Individual D* or *Joint D*. Although all these architectures can be trained from scratch directly, this scheduled training process accelerates convergence. While training *Multi Agent Scene*, the context scene

image is randomly dropout of $p = 0.5$ to encourage the module not to be dependent too much on either scene or multi-agent information too much. We iteratively train the network with a batch size of 32 (note that 32 scenes per batch indicates much more than 32 agents given our multi-agent scenario) using Adam with an initial learning rate of 0.001. We use PyTorch for implementation.