Generative Attention Networks for Multi-Agent Behavioral Modeling

Guangyu Li,[1] **Bo Jiang,**[2] **Hao Zhu,**[3] **Zhengping Che,**[2] **Yan Liu**[1*]
[1]University of Southern California, Los Angeles, US
[2]AI Labs, Didi Chuxing, Beijing, China
[3]Peking University, Beijing, China
{guangyul, yanliu.cs}@usc.edu, hzhu1998@pku.edu.cn,
{scottjiangbo, chezhengping}@didiglobal.com

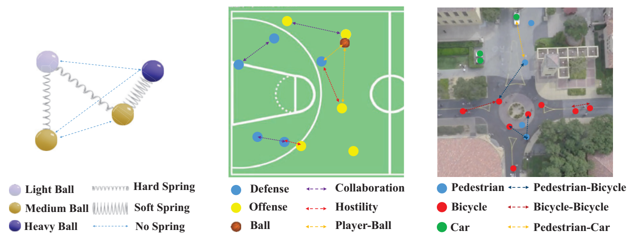
Abstract

Understanding and modeling behavior of multi-agent systems is a central step for artificial intelligence. Here we present a deep generative model which captures behavior generating process of multi-agent systems, supports accurate predictions and inference, infers how agents interact in a complex system, as well as identifies agent groups and interaction types. Built upon advances in deep generative models and a novel attention mechanism, our model can learn interactions in highly heterogeneous systems with linear complexity in the number of agents. We apply this model to three multi-agent systems in different domains and evaluate performance on a diverse set of tasks including behavior prediction, interaction analysis and system identification. Experimental results demonstrate its ability to model multi-agent systems, yielding improved performance over competitive baselines. We also show the model can successfully identify agent groups and interaction types in these systems. Our model offers new opportunities to predict complex multi-agent behaviors and takes a step forward in understanding interactions in multi-agent systems.

Introduction

Multi-agent systems are widespread in many real-world applications, including autonomous vehicles, multi-player games, etc. Approximating the behavior generating process of multi-agent systems from observations is essential for behavior prediction, system identification, simulation and general machine intelligence.

Unlike other machine learning domains such as computer vision and natural language processing, multi-agent systems are usually modeled in terms of agents and their interactions, given the explicit factorization within these systems. In some cases, interactions themselves play a key role in understanding multi-agent behavior. No matter playing football, or coordinating to take turns with other traffic participants at busy intersections, the behavior of each agent is heavily affected by other participants' behaviors. However, modeling behavior generating process and inferring interactions in multi-agent systems is challenging due to inherent properties in the following aspects.



(a) Spring-Ball (b) Basketball (c) Campus Traffic

Figure 1: Illustration of three multi-agent systems, including agent groups and interaction types.

Interaction Complexity To model interactions explicitly, one suffers the complexity of $O(N^d)$ where N is the number of agents and d is the typical interaction clique size, which makes it computationally unmanageable to model a large amount of agents or high order interactions.

Interaction Structure Although the interaction structure can be pre-defined in some simple systems, e.g., spring-systems in physics, most real-world multi-agent systems come with unknown and evolving interaction structure. Imagining in an open area in the university campus, the number of agents changes frequently, and the interaction between two pedestrians may disappear quickly after they passed by each other. Such unknown and evolving properties require a model to handle interaction adaptively.

Heterogeneity In a multi-agent system, agents usually come from groups with different behavior styles, and interactions among them may also have inherently different types. Considering in a basketball game, players are switching between offense and defense frequently. Interactions between players also come with two types, namely collaboration and hostility. To truly approximate behavior generating process, one should be able to identify such agent groups and interaction types from system observations.

Pioneering work in multi-agent behavior modeling has tackled some of the above challenges. Interactions within the crowd of pedestrians have been modeled implicitly with pooling mechanisms in (Alahi et al. 2016; Gupta et al. 2018;

[*]First two authors contributed equally.

Lee et al. 2017). However, local pooling mechanisms designed for pedestrian modeling in (Alahi et al. 2016) only deal with short-range interactions and provide limited interpretations. On the other hand, pairwise interactions are modeled explicitly with Interaction Networks (Battaglia et al. 2016), showing accurate results for physical systems of a small number of agents in which the interaction is pairwise in nature. Despite the tremendous progress in existing work, we are still in lack of a mechanism which can approximate underlying behavior generating process, capture interactions in a computationally efficient and interpretable fashion, as well as identify heterogeneity in multi-agent systems.

Here we present *Generative Attentional Multi-Agent Network* (GAMAN), a deep generative model with an *attention mechanism* for multi-agent behavior modeling which handles all challenges mentioned above. Build upon advances in deep generative model (Chung et al. 2015) (Krishnan, Shalit, and Sontag 2016), GAMAN targets on approximating the underlying generating distribution from which the multi-agent behavior was sampled, instead of cloning the behavior deterministically. In this way, GAMAN captures stochasticity in the latent dynamics that beyond short-term variation one can capture with observation-level noise. As a generative model, GAMAN also supports multi-agent behavior with multi-modal distributions by adopting mixture density in the generation process, as well as infer missing behavior when the system is only partially observable.

With the design of an *attention mechanism* and a graph representation, GAMAN captures agents' interactions with linear complexity in the number of agents, and handle the evolving interaction structure. Inspired by (Hoshen 2017), the interaction structure among agents is learned directly from observations rather than being pre-specified. Therefore GAMAN maintains a degree of interpretability as one can visualize the interaction structure from learned attention. With learned graph representation, GAMAN also supports system heterogeneity identification, by clustering agents and interactions embeddings. Moreover, GAMAN is amenable to dynamic variation in number of agents.

We conduct extensive quantitative and qualitative analysis of GAMAN on three inherently different multi-agent systems, namely: spring-ball system, basketball games and traffics in a university campus, illustrated in Figure 1. Experiments show that GAMAN supports accurate predictions as well as system identification in which hidden properties are abducted from observations, in all three domains.

In summary, our work makes following contributions:

- We introduce a deep generative model into multi-agent behavior modeling, which approximates underlying behavior generating distribution, supports heterogeneous systems, provides accurate predictions.
- We propose an *attention mechanism* working with deep generative model, which models interactions within multi-agent systems in a parsimonious way and provides interpretable interaction structure.
- We demonstrate the proposed model provides more accurate prediction and inference as well as more interpretability compared with competitive baselines in three different applications.

Related Work

Multi-agent behaviors have drawn significant research interests over the past decades. Pioneer work from Helbing and Molnar (Helbing and Molnar 1995) introduces social force model to capture pedestrian motion pattern. This work was later extended in (Mehran, Oyama, and Shah 2009) to learn the interaction forces in the human crowd, which relies on hand-crafted features and relative distances.With the emerge of neural networks, Recurrent Neural Networks (RNNs) or Long Short-Term Memory networks (LSTM) based models have been introduced to model multi-agent behavior, (Vemula, Muelling, and Oh 2018; Manh and Alaghband 2018; Alahi et al. 2016; Gupta et al. 2018; Lee et al. 2017; Xu, Piao, and Gao 2018). In this line of research, specially designed *pooling mechanism* is the key to capture interactions. Alahi et al. propose Social LSTM (Alahi et al. 2016) that predicts pedestrians trajectory using RNNs with a novel social pooling layer in the hidden state of nearby pedestrians. Later, Gupta et al. propose Social GAN (Gupta et al. 2018) showing a Multi-Layer Perceptron (MLP) followed by max pooling works better than social pooling method in a similar task and enjoys better computationally efficiency. Lee et al. (Lee et al. 2017) target on traffic behavior prediction at intersections and design DESIRE, an RNN Encoder-Decoder framework together with feature pooling layer. Although pooling layers methods show competitive performance in multi-agent behavior predictions, the lack of interoperability limits our understanding towards how multi-agents interact with each other.

Besides the pooling mechanism, Graph Neural Network (GNN) has also been introduced to handle the multi-agent system. For example, Battaglia et al. propose Interaction Networks (IN) to learn a physical simulation of objects with binary relations (Battaglia et al. 2016), Sukhbaatar et al. propose Communication Networks (CommNets) (Sukhbaatar, Fergus, and others 2016) for learning optimal communications between agents. Later, Hoshen introduces attention mechanism into GNN and develop VAIN (Hoshen 2017) and apply it to soccer data. Although VAIN scales efficiently with number of agents, it only predicts from a single frame and does not model past trajectories.

Another line of research models multi-agent systems in the imitation learning framework. The key idea is to learn policy representations from agent actions and system observations (Grover et al. 2018; Le et al. 2017; Liu et al. 2018). With the imitation learning framework, these models require explicitly defined action set, which targets more on continuous control and robotic applications.

Deep Generative Model Deep generative models have been proposed to reveal the underlying data generating process for sequential data (Rezende, Mohamed, and Wierstra 2014; Krishnan, Shalit, and Sontag 2016; Chung et al. 2015; Che et al. 2018), to name a few. Deep Markov Model (DMM), a nonlinear state-space model, is proposed in (Krishnan, Shalit, and Sontag 2016) by marrying the ideas of deep neural networks with Kalman filters. In (Chung et al. 2015), the authors introduced the variational recurrent neural network (VRNN) which glued an RNN with a VAE together to form a stochastic and sequential neural generative model.

Recently, there are several explorations of the deep generative model in multi-agent systems. Ivanovia et al. (Ivanovic et al. 2018) combine Conditional Variational Autoencoder (CVAE) (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014) and LSTM to generate behavior of basketball players. In another relevant work (Zhan et al. 2018), Zhan et al. also use VRNN-based generative models (Chung et al. 2015) and introduce a macro-goals mechanism (i.e., basketball players' intentions) and break down the multi-agent behavior generation to an evolving process with multiple pre-defined consequences. More recently, Yeh et al. integrate VRNN and GNN into GraphVRNN (Yeh et al. 2019), to model trajectory in sports. Most of the existing works focus on agent behavior forecasting and provide limited information regarding interactions types or agent groups.

Generative Attentional Multi-Agent Network

We consider a multi-agent system of K agents over time horizon T, which consists of behavior of every agent over T time steps. We establish notations as follow:

- Let $\mathcal{D}$ denote all behavior for a multi-agent system.
- Let $\mathbf{x}_{\leq T} = \{\mathbf{x}_t\}_{1 \leq t \leq T}$ denote a system's behaviors over time horizon T, where $\mathbf{x}_t = \{\mathbf{x}_t^k\}$ and $\mathbf{x}_t^k$ is the behavior of agent k at time step t.
- Let $\mathbf{z}_{\leq T} = \{\mathbf{z}_t\}_{1 \leq t \leq T}$ denote a system's latent states over time horizon T, where $\mathbf{z}_t = (\mathbf{e}_t, \mathbf{s}_t, \mathbf{a}_t) = (\{\mathbf{e}_t^k\}_{\text{agent } k}, \{\mathbf{s}_t^k\}_{\text{agent } k}, \{\mathbf{a}_t^k\}_{\text{agent } k})$ and $\mathbf{e}_t^k, \mathbf{s}_t^k, \mathbf{a}_t^k$ corresponds to the *agent vector*, *interaction vector*, *attention vector* of agent k at time step t respectively.
- Let $\{\theta_{\mathbf{x}}, \theta_{\mathbf{z}}, \theta_{\mathbf{s}}, \theta_a\}$ denote the parameter set for generation network π_θ and $\{\phi_{\mathbf{x}}, \phi_{\mathbf{z}}, \phi_{\mathbf{s}}, \phi_a\}$ denote the parameter set for inference network π_ϕ.

The goal of GAMAN is to approximate the underlying behavior generating process and recover agents' interactions in a multi-agent system. To achieve this goal, GAMAN adopts latent state assumption for the system: at every time step t, there is a latent state $\mathbf{z}_t$ representing full information of the system, including each agent's state and their interactions. As the system evolving, new latent state $\mathbf{z}_{t+1}$ is generated from current latent state $\mathbf{z}_t$ and the behavior observation $\mathbf{x}_{t+1}$ is generated from new latent state $\mathbf{z}_{t+1}$. GAMAN represents this process by a generation network with attention mechanism parametrized by θ, from which we can sample the system's behavior observations $\mathbf{x}_{\leq T}$ conditioned on its latent states $\mathbf{z}_{\leq T}$.

$$\mathbf{x}_{\leq T} \sim p_\theta(\mathbf{x}_{\leq T} | \mathbf{z}_{\leq T}) \tag{1}$$

In order to learn the network, we maximize the marginal log likelihood of all behavior observations which is the summation of marginal log likelihood of every observation $\mathbf{x}_{<T}$.

$$\theta^* = \text{argmax}_\theta \sum_{\mathcal{D}} \mathcal{L}(\theta) = \text{argmax}_\theta \sum_{\mathcal{D}} \sum_{t=1}^{T} \log p_\theta(\mathbf{x}) \tag{2}$$

Given the generation network, the marginal log-likelihood of one behavior observation $p_\theta(\mathbf{x})$ could be calculated by integrating out all possible latent states $\mathbf{z}$. However, stochastic latent states $\mathbf{z}$ cannot be analytically integrated out. Following the same trick used in VAE (Kingma and Welling 2013), VRNN (Chung et al. 2015) and DMM (Krishnan, Shalit, and Sontag 2016), we resort to the well-known variational principle and design a inference network parameterized by ϕ , and maximize the *variational evidence lower bound* (ELBO) $\mathcal{F}(\theta, \phi) \leq \mathcal{L}(\theta)$ with respect to both θ and ϕ. The generation network and inference network are illustrated in Figure 2. In this section, we present the design of generation and inference network in details. Note that though GAMAN supports heterogeneous systems with multiple agent types, we only consider a single agent type in this section for the sake of clarity.

Generation Network

The generation network of GAMAN follows the **emission** and **transition** framework, which is designed by applying deep neural networks to continuous state space models.

Transition We design transition network from latent state $\mathbf{z}_{t-1} = \{\mathbf{e}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}\}$ to $\mathbf{z}_t = \{\mathbf{e}_t, \mathbf{s}_t, \mathbf{a}_t\}$ in GAMAN to learn the temporal dependencies in multi-agent systems. We also introduce *attention mechanism* into the transition network to capture and reveal agents' interactions. The transition network contains three sub-functions: agent function f_{θ_e}, interaction function f_{θ_s} and attention function f_{θ_a}. In GAMAN, we use gated recurrent units (GRU) (Chung et al. 2014) for agent function f_{θ_e}, multilayer perceptron (MLP) for interaction function f_{θ_s} and attention function f_{θ_a}.

At time step t, the *agent vector* of k-th agent $\mathbf{e}_t^k$ is first sampled from transition distribution π_{θ_e}

$$\mathbf{e}_t^k \sim \pi_{\theta_e}\big(\mathbf{e}_t^k | \mathbf{e}_{<t}^k, \{\mathbf{s}_{t-1}^j\}_{j \neq k}, \{\mathbf{a}_{t-1}^j\}_{j \neq k}; \theta_e\big) \tag{3}$$

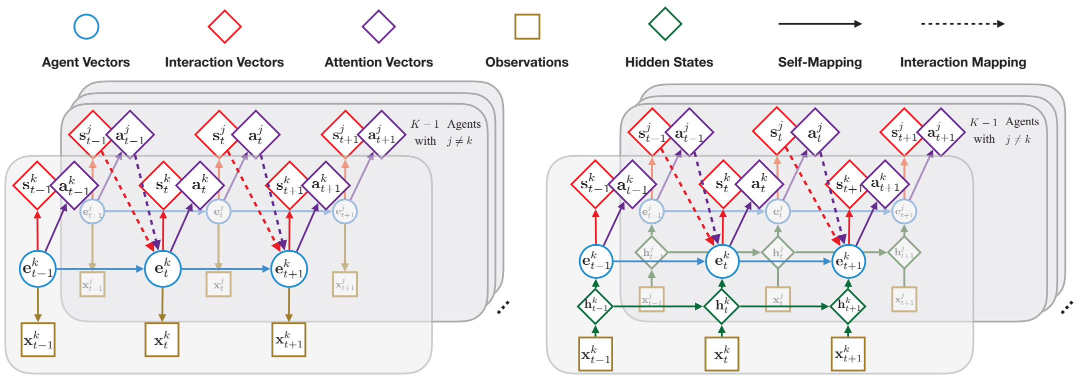
In GAMAN, we model the transition distribution as a multivariate Gaussian distribution $\pi_{\theta_e} = \mathcal{N}({\mu^{(\theta)}}_t^k, {\mathbf{\Sigma}^{(\theta)}}_t^k; \theta_e)$. We parametrize π_{θ_e} with the agent function f_{θ_e}, which takes previous *agent vectors* of the k-th agent $\mathbf{e}_{<t}^k$ and all other agents' *interaction vectors* $\{\mathbf{s}_t^j\}_{j \neq k}$ weighted by the weight $w_{k,j} = Softmax_k(-||\mathbf{a}_k^t - \mathbf{a}_j^t||^2)$.

$$\begin{aligned}({\mu^{(\theta)}}_t^k, {\mathbf{\Sigma}^{(\theta)}}_t^k) &= f_{\theta_e}\left(\mathbf{e}_{<t}^k, \sum_{j \neq k} w_{k,j} \mathbf{s}_{t-1}^j\right) \\ &= f_{\theta_e}\left(\mathbf{e}_{<t}^k, \frac{\sum_{j \neq k} e^{-||\mathbf{a}_{t-1}^k - \mathbf{a}_{t-1}^j||^2} \mathbf{s}_{t-1}^j}{\sum_{j \neq k} e^{-||\mathbf{a}_{t-1}^k - \mathbf{a}_{t-1}^j||^2}}\right)\end{aligned} \tag{4}$$

Then the *interaction vector* $\mathbf{s}_t^k$ and *attention vector* $\mathbf{a}_t^k$ of the k-th agent are updated through interaction function f_{θ_s} and attention function f_{θ_a} respectively.

$$\mathbf{s}_t^k = f_{\theta_s}(\mathbf{e}_t^k) \qquad \mathbf{a}_t^k = f_{\theta_a}(\mathbf{e}_t^k) \tag{5}$$

Combining equations 3, 4 and 5, GAMAN generates the latent state $\mathbf{z}_t = \{\mathbf{e}_t, \mathbf{s}_t, \mathbf{a}_t\}$ from previous one $\mathbf{z}_{t-1} = \{\mathbf{e}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}\}$.



(a) Generation Network for the k-th agent (b) Inference Network for the k-th agent

Figure 2: Generation and inference network of GAMAN for the k-th agent in a K-agent system. Note that we only consider *forward* setting for the sake of clarity.

The motivation behind such *attention mechanism* comes as follows. In a multi-agent system, an agent's temporal dynamic is affected by itself and other agents' interactions. Ideally, the transition distribution should be parametrized by the combination of the agent encoding vector $\mathbf{e}_k^t$ and all cliques of interaction (could be 2nd or higher-order). However, these interaction cliques are not known a-priori in most scenarios and usually require $O(N^d)$ encoding evaluations where N is the number of agents and d is the typical interaction clique size. To address these limitations, we design the *attention mechanism* to approximate other agents' interactions by a weighted average of each agent's interaction encoding vector $\{\mathbf{s}_t^j\}_{j\neq k}$ as in Equation 4. With such linear approximation, GAMAN could incorporate interactions in a multi-agent system efficiently as well as provide a measure of the interaction between agents through the weight $w_{k,j}$. Although the $Softmax$ weight operation scales quadratically with the number of agents, it cost significantly less computation than evaluating interaction encoding $\mathbf{s}$ which scales linearly in a number of agents. Therefore the overall computation still scales linearly in the number of agents. Furthermore, $Softmax$ operation usually yields sparse results, in which the interaction can be further approximated by K-nearest neighbor measured in attention, boosting the computational efficiency. Beyond efficiency advantage, the proposed *attention mechanism* is also robust to variation in the number of agents. Because no matter how the amount of agents varies, the $Softmax$ weighted average will adaptively distribute the interaction within the system.

Emission Multi-agent system's behavior $\mathbf{x}$ is generated from latent states $\mathbf{z}$ in the emission process.

At time step t, behavior of the k-th agent $\mathbf{x}_t^k$ is sampled from emission distribution π_{θ_x}

$$\mathbf{x}_t^k \sim \pi_{\theta_x}(\mathbf{x}_t^k|\mathbf{e}_t^k;\theta_x) \tag{6}$$

The choice of emission distribution π_{θ_x} is flexible and depends on applications. Multinomial distribution is used for categorical data while Gaussian distribution is used for continuous data. For multi-agent systems, mixture distribution is adopted given the multi-modal nature of agent's behavior.

The emission distribution π_{θ_x} is parametrized by a multi-layer perception (MLP) mapping f_{θ_x}, which takes the *agent vector* of the k-th agent $\mathbf{e}_t^k$ as input. For example, consider a Gaussian distribution (each with mean ${\mu^{(\mathbf{x})}}_t^k$, covariance ${\mathbf{\Sigma}^{(\mathbf{x})}}_t^k$ and weight w_t^k) as the emission distribution π_{θ_x},

$$\{{\mu^{(\mathbf{x})}}_t^k, {\mathbf{\Sigma}^{(\mathbf{x})}}_t^k, w_t^k\} = f_{\theta_x}(\mathbf{e}_t^k) \tag{7}$$

To summarize, the overall generation process is described in Algorithm 1. The parameter set of generation network is $\theta = \{\theta_x, \theta_e, \theta_s, \theta_a\}$. The joint probability of behavior and latent states can be factorized as following Equation 8.

$$\begin{aligned} p_\theta(\mathbf{x}_{\le T}, \mathbf{z}_{\le T}|\mathbf{z}_0) &= p_{\theta_x}(\mathbf{x}_{\le T}|\mathbf{e}_{\le T}) \cdot p_{\theta_e}\left(\mathbf{e}_{\le T}|\mathbf{z}_0;\theta_s,\theta_a\right) \\ &= \prod_{t=1}^{T}\left[p_{\theta_x}(\mathbf{x}_t|\mathbf{e}_t)\cdot p_{\theta_e}(\mathbf{e}_t|\mathbf{e}_{t-1},\mathbf{s}_{t-1},\mathbf{a}_{t-1};\theta_s,\theta_a)\right] \\ &= \prod_{t=1}^{T}\prod_{k=1}^{K}\Big[p_{\theta_x}(\mathbf{x}_t^k|\mathbf{e}_t^k)\cdot p_{\theta_e}\Big(\mathbf{e}_t^k|\mathbf{e}_{t-1}^k,\{\mathbf{s}_{t-1}^j\}_{j\neq k}, \\ &\qquad \{\mathbf{a}_{t-1}^j\}_{j\neq k};\theta_s,\theta_a\Big)\Big] \end{aligned} \tag{8}$$

Inference Network

The goal of the inference network is to obtain an objective which can be optimized easily, making the model parameter learning amenable. Instead of directly maximizing marginal log-likelihood $\mathcal{L}(\theta)$ w.r.t. θ, we build an inference network with a tractable distribution π_ϕ and maximize the *variational evidence lower bound* (ELBO) $\mathcal{F}(\theta,\phi) \le \mathcal{L}(\theta)$ w.r.t both θ and ϕ. Here ϕ is the parameter set of the inference network which will be formally defined at the end of this section. The overall ELBO is summed over each time step of the behavior demonstration.

$$\begin{aligned} \mathcal{F}(\theta,\phi) =& \mathbf{E}_{q_\phi(\mathbf{z}_{\le T}|\mathbf{x}_{\le T})}\Big[\sum_{t=1}^{T}\log p_\theta(\mathbf{x}_t|\mathbf{z}_{\le T}) \\ & - D_{KL}(q_\phi(\mathbf{z}_{\le T}|\mathbf{x}_{\le T},\mathbf{z}_0)||p_\theta(\mathbf{z}_{\le T}|\mathbf{z}_0))\Big] \end{aligned} \tag{9}$$

Algorithm 1 Generation Network of GAMAN in a K-agent system with mixture of Gaussian emission distribution.

```
1: for $k = 1, \dots, K$ do
2:     Initialize $\mathbf{e}_0^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3:     Initialize $\mathbf{s}_0^k = f_{\theta_s}(\mathbf{e}_0^k) \quad \mathbf{a}_0^k = f_{\theta_a}(\mathbf{e}_0^k)$
4: end for
5: for $t = 1, \dots, T$ do
6:     for $k = 1, \dots, K$ do
7:         $({\mu^{(\theta)}}_t^k, {\mathbf{\Sigma}^{(\theta)}}_t^k) = f_{\theta_e}\left(\mathbf{e}_{<t}^k, \sum_{j \neq k} w_{k,j}\mathbf{s}_{t-1}^j\right)$
8:         Sample $\mathbf{e}_t^k \sim \mathcal{N}({\mu^{(\theta)}}_t^k, {\mathbf{\Sigma}^{(\theta)}}_t^k; \theta_e)$
9:         Compute $\mathbf{s}_t^k = f_{\theta_s}(\mathbf{e}_t^k) \quad \mathbf{a}_t^k = f_{\theta_a}(\mathbf{e}_t^k)$
10:    end for
11:    for $k = 1, \dots, K$ do
12:        Compute $\{{\mu^{(\mathbf{x})}}_t^k, {\mathbf{\Sigma}^{(\mathbf{x})}}_t^k, w_t^k, \}_m = f_{\theta_x}(\mathbf{e}_t^k)$
13:        Sample $\mathbf{x}_t^k \sim \sum_{i=i}^m \{w_t^k\}_i \mathcal{N}(\{{\mu^{(\mathbf{x})}}_t^k, {\mathbf{\Sigma}^{(\mathbf{x})}}_t^k\}_i)$
14:    end for
15: end for
```

where the expectation is taken w.r.t $q_\phi(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$, the approximated posterior of latent states $\mathbf{z}$ provided by the inference network. To get a tight bound and an accurate estimate from GAMAN, we design the inference network as follows. First, we keep the same temporal dependency of latent states $\mathbf{z}$ in the inference network, leading to the factorization:

$$q_\phi(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{<t}, \mathbf{x}_{\leq T}) \tag{10}$$

Further, the inference network inherits both *attention mechanism* and corresponding conditional independence from the generation network. Thus the right-hand side of Equation 10 can be further factorized as:

$$\begin{aligned} q_\phi(\mathbf{z}_t|\mathbf{z}_{<t}, \mathbf{x}_{\leq T}) &= q_\phi(\mathbf{e}_t, \mathbf{s}_t, \mathbf{a}_t|\mathbf{e}_{<t}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{x}_{\leq T}) \\ &= q_\phi(\mathbf{e}_t|\mathbf{e}_{<t}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{x}_{\leq T}) \\ &= \prod_{k=1}^{K} q_\phi(\mathbf{e}_t^k|\mathbf{e}_{<t}^k, \{\mathbf{s}_{t-1}^j\}_{j\neq k}, \{\mathbf{a}_{t-1}^j\}_{j\neq k}, \mathbf{x}_{\leq T}^k) \end{aligned} \tag{11}$$

while non-stochastic *interaction vector* $\mathbf{s}_t^k$ and *attention vector* $\mathbf{a}_t^k$ are mapped directly from *agent vector* $\mathbf{e}_t^k$

$$\mathbf{s}_t^k = f_{\theta_s}(\mathbf{e}_t^k) \qquad \mathbf{a}_t^k = f_{\theta_a}(\mathbf{e}_t^k) \tag{12}$$

where function f_{θ_s} and f_{θ_a} are inherited from generation network. Further, we model the right hand side of Equation 11 $q_\phi(\mathbf{e}_t^k|\mathbf{e}_{<t}^k, \{\mathbf{s}_{t-1}^j\}_{j\neq k}, \{\mathbf{a}_{t-1}^j\}_{j\neq k}, \mathbf{x}_{\leq T}^k)$ as Gaussian distribution $\pi_{\phi_e}^k = \mathcal{N}({\mu^{(\phi)}}_t^k, {\mu^{(\phi)}}_t^k; \phi_e)$ parametrized by gated recurrent units (GRU) $g_{\phi_\mathbf{e}}$

$$({\mu^{(\phi)}}_t^k, {\mathbf{\Sigma}^{(\phi)}}_t^k) = g_{\phi_e}\left(\mathbf{h}_t^k, \mathbf{e}_{<t}^k, \frac{\sum_{j\neq k} e^{||\mathbf{a}_{t-1}^k - \mathbf{a}_{t-1}^j||^2}\mathbf{s}_{t-1}^j}{\sum_{j\neq k} e^{||\mathbf{a}_{t-1}^k - \mathbf{a}_{t-1}^j||^2}}\right) \tag{13}$$

where $\mathbf{h}_t^k$ is the encoding of the k-th agent's behavior $\mathbf{x}^k$. Inspired by (Krishnan, Shalit, and Sontag 2016) , we construct this *behavior encoding* with *forward*. In *forward* setting, we only use a forward RNN $g^{forward}$ to pass the information up to time step t into *behavior encoding* $\mathbf{h}_t^k$ i.e. $\mathbf{h}_t^k = g^{forward}(\mathbf{x}_{\leq t}^k)$. The *forward* setting only use the information in the past, so it is suitable for prediction task at future time step $t' > T$. On the other hand, *bi-direction* setting uses a bi-directional RNN to capture information from both history and future, i.e. $\mathbf{h}_t^k = \left[g^{forward}(\mathbf{x}_{\leq t}^k), g^{backward}(\mathbf{x}_{t+1:T}^k)\right]$, which supports system identification and inferring unobserved part from partially observed behaviors. To summarize, we use ϕ_h and ϕ_e to denote parameters related to *behavior encoding* $\mathbf{h}$ and *agent vector* $\mathbf{e}$ respectively and use $\phi = \{\phi_h, \phi_e\}$ to represent the parameter set of the inference network.

Algorithm 2 Learning GAMAN with stochastic backpropagation and Adam optimizer (Kingma and Ba 2014).

```
Require: A set of behavior demonstrations $\mathcal{D}$; Initial $(\theta, \phi)$
1: while not converged do
2:     Choose a random minibatch of behaviors $\mathcal{X} \subset \mathcal{D}$
3:     for each sample $\mathbf{x}_{1:T} \in \mathcal{X}$ do
4:         for $k = 1, \dots, K$ do
5:             Compute $\mathbf{h}_{1:T}^k$ by function $\phi_h$ on input $\mathbf{x}_{1:T}^k$
6:             Sample $\widehat{\mathbf{e}_0^k} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
7:             Sample $\mathbf{s}_0^k = f_{\theta_s}(\mathbf{e}_0^k) \quad \mathbf{a}_0^k = f_{\theta_a}(\mathbf{e}_0^k)$
8:         end for
9:         for $t = 1, \cdots, T$ do
10:            for $k = 1, \dots, K$ do
11:                Estimate parameters $({\mu^{(\theta)}}_t^k, {\mathbf{\Sigma}^{(\theta)}}_t^k)$ by $f_{\theta_e}$ 4
12:                    given $\mathbf{e}_{<t}^k, \{\mathbf{s}_{t-1}^j\}_{j\neq k}, \{\mathbf{a}_{t-1}^j\}_{j\neq k}$
13:                Estimate parameters $({\mu^{(\phi)}}_t^k, {\mathbf{\Sigma}^{(\phi)}}_t^k)$ by $g_{\phi_e}$ 13
14:                    given $\mathbf{e}_{<t}^k, \{\mathbf{s}_{t-1}^j\}_{j\neq k}, \{\mathbf{a}_{t-1}^j\}_{j\neq k}, \mathbf{h}_t^k$
15:                Compute the gradient of
16:                    $D_{\mathrm{KL}}\left(q_\phi\left(\boldsymbol{z}_t^k|\cdot\right)\middle\|p_\theta\left(\boldsymbol{z}_t^k|\cdot\right)\right)$ w.r.t $(\theta, \phi)$
17:                    given ${\mu^{(\theta)}}_t^k, {\mathbf{\Sigma}^{(\theta)}}_t^k$ and ${\mu^{(\phi)}}_t^k, {\mathbf{\Sigma}^{(\phi)}}_t^k$
18:                Sample $\widehat{\mathbf{e}_t^k} \sim \mathcal{N}({\mu^{(\phi)}}_t^k, {\mathbf{\Sigma}^{(\phi)}}_t^k; \phi_e)$
19:                Compute the gradient of $\log p_{\theta_x}\left(\mathbf{x}_t^k|\widehat{\mathbf{e}_{\leq t}^k}\right)$
20:            end for
21:        end for
22:    end for
23:    Update $(\theta, \phi)$ using all gradients with Adam
24: end while
```

Parameter Learning

We learn the parameters (θ, ϕ) of GAMAN by maximizing the ELBO $\mathcal{F}(\theta, \phi)$ in Equation 9 over the set of all behavior demonstrations $\mathcal{D}$.

$$(\theta^*, \phi^*) = \mathrm{argmax}_{\theta,\phi} \sum_{\mathcal{D}} \mathcal{F}(\theta, \phi) \tag{14}$$

We use stochastic backpropagation (Kingma and Welling 2013) and ancestral sampling for estimating all gradients w.r.t (θ, ϕ) and train the networks with Adam optimizer (Kingma and Ba 2014). Algorithm 2 shows the overall learning procedure of GAMAN.

Note that here we assume all agents share the same behavior generating distribution, i.e., the same set of parameters

(θ, ϕ). No agent role or any agent-specific information was used during training, and the proposed model is trained with randomly shuffled trajectories. However, GAMAN can be easily adapted to a system with known agent types, by simply assigning different behavior generating distribution - different sets of parameters (θ, ϕ) - for each agent type. Agents could still interact within and across agent types through the *attention mechanism* introduced above. We will explore the performance of GAMAN on the heterogeneous multi-agent system later in the experiment section.

Experiments

We conduct experiments on one simulated physical dataset, **Spring-Ball**, and two real-world large scale multi-agent datasets, **Stanford Drone** (Robicquet et al. 2016) and **NBA** (Linou 2016). Through experiments, we answer the following questions: (1) How good are the behavioral predictions of our proposed model for multi-agent systems compared to the existing state-of-the-art methods? (2) Are the interactions learned by the proposed *attention mechanism* helpful for multi-agent behavioral modeling? (3) Is the proposed model able to identify different agent groups and interaction types? In the remainder of this section, we illustrate the datasets, tasks, quantitative results, and interpretations to answer the questions above.

Datasets

Spring-Ball We design a spring-ball system following (Kipf et al. 2018). In this scenario, balls are bouncing inside a 2D square container of size $L(= 10)$ governed by Hooke's law. $N(= 100)$ balls with mass ratio of 0.2:1:5 generated at the probability of 0.3:0.4:0.3 are randomly connected by $\{$*no*, *soft*, *hard*$\}$ springs at the probability of 0.8:0.1:0.1, where the elastic coefficient of soft and hard springs is 0.5:2. The static lengths of all springs are arbitrarily set to 1. The balls are initialized with random positions and velocities. Balls collide with the walls governed by the laws of elastic collisions as balls are viewed as mass points. The trajectories are simulated at 1000 FPS, and we sample the data every 100 frames. We generated 1000 cases, each has 49 frames.

Stanford Drone This dataset consists of multiple aerial videos of 8 unique scenes captured around the Stanford campus, and there are six types of agents (e.g., biker, car) navigating the crowded spaces with dynamic interactions. For experiments, we choose the top 3 types of agents, `Biker`, `Pedestrian`, and `Car` from the dataset, and we use all the videos except the ones in scene *little*, *coupa*, and *quad*, since the selected types of agents are much more sparse in these 3 scenes. The original frame rate of these videos is around 29.97 FPS, and we down-sampled all the videos to 5 FPS. Besides, we divide each video into 12-second long (i.e., 60 frames) distinct sub-videos along the timeline, and we treat each sub-video as one *multi-agent case*. We only keep the cases with the number of agents between 10 and 90. Table 1 shows the statistics of the processed dataset.

NBA This dataset is composed of 49,628 12-second sequences of the 2D basketball players and ball overhead-view point trajectories from 200 games in the 15/16 NBA season captured by SportVU Player Tracking technology. It tracks positions of each player and collects data every 40ms. We down-sampled all the data to 5 FPS. To reduce the uncertainty of the trajectories, we align the sequences and ensure the offense always shoots toward the same side.

Table 1: Statistics of the processed Stanford Drone dataset.

Scenes	# of Cases	# of Bikers	# of Pedestrians	# of Cars
gates	93	716	652	19
nexus	240	108	2182	1316
bookstore	191	733	1784	15
deathCircle	110	1517	1174	189
hyang	234	1004	2649	15
Total	868	4078	8441	1554

Experimental Design

Prediction Task For a multi-agent case with T frames, given its first L frames (of all agents), one needs to predict each agent's future behaviour in the last $T - L$ frames. For our datasets, the (T, L) pairs are: Spring-ball, $(49, 35)$; NBA, $(60, 50)$; Stanford Drone, $(60, 50)$.

Evaluation Metrics Given we consider the agent's trajectory as its behavior, we evaluate the behavior forecasting with four different metrics:

- L_2 *distance* (**L2**): The L_2 distance between predicted trajectories and the ground truth, averaged over each predicted frame for each agent.
- *Maximum* L_2 *distance* (**maxL2**): The maximum L_2 distance between the prediction and ground truth throughout a predicted trajectory, averaged over each agent.
- *Miss rate*: The fraction of trajectories whose distance between the final predicted and ground-truth point pairs exceeds a constant number x. We choose x as 0.2 and 0.5, noted by **MR0.2** and **MR0.5** respectively.

Baselines We compare our GAMAN with multiple baseline models in the trajectory forecasting task. To demonstrate the advantage of *attention mechanism* and learned interactions in GAMAN, we remove all interactions from GAMAN for ablation comparisons.

- *Extrapolation and linear method*:
  - **Velocity**: a velocity-based extrapolation.
  - **KF**: Kalman Filter.
- *Deep neural networks based models*:
  - **LSTM**: Long Short Term Memory Network (Hochreiter and Schmidhuber 1997)
  - **SocialGAN**: Socially Acceptable Trajectories with Generative Adversarial Networks (Gupta et al. 2018)
  - **SocialLSTM**: Social Pooling with LSTMs (Alahi et al. 2016)
  - **NRI**: Neural Relational Inference Model (Kipf et al. 2018)
  - **IN**: Interaction Networks (Battaglia et al. 2016)
  - **GAMAN-NI**: **GAMAN** with **N**o **I**nteraction.

Table 2: Behavior prediction results on 3 datasets.

	Spring-Ball				**Stanford Drone**				**NBA**			
	L2	**maxL2**	**MR0.2**	**MR0.5**	**L2**	**maxL2**	**MR0.2**	**MR0.5**	**L2**	**maxL2**	**MR0.2**	**MR0.5**
Velocity	0.3786	0.9338	0.6778	0.4126	0.0214	0.0436	0.0380	0.0049	0.2470	0.5337	0.7745	0.3885
KF	1.0616	1.8522	0.9500	0.7938	0.2482	0.4790	0.5900	0.1835	0.6566	1.1420	0.9385	0.7303
LSTM	0.1839	0.4108	0.5440	**0.2662**	0.0249	0.0478	0.0323	0.0041	0.1823	**0.3636**	0.6439	0.2107
SocialGAN	0.1952	0.4549	0.6706	0.2989	0.0231	0.0453	0.0384	0.0038	0.2495	0.5031	0.7954	0.3873
SocialLSTM	0.6763	1.2372	0.9628	0.8129	0.0702	0.1239	0.0813	0.0062	0.4995	0.8994	0.9460	0.6998
NRI	0.2311	0.5351	0.6541	0.3529	0.0291	0.0596	0.0382	0.0033	0.1725	0.3647	0.6415	0.2222
IN	0.2657	0.6323	0.7324	0.4212	0.0257	0.0606	**0.0285**	**0.0031**	0.1855	0.4114	0.7444	0.2639
GAMAN-NI	0.2624	0.5968	0.5371	0.4603	0.0264	0.0572	0.0349	0.0052	0.2465	0.5983	0.6428	0.4371
GAMAN	**0.1728**	**0.4049**	**0.4431**	0.2885	**0.0203**	**0.0416**	0.0335	0.0047	**0.1720**	0.3794	**0.6089**	**0.2023**

Table 3: Log-likelihood of generative models on 3 datasets.

	SocialLSTM	**NRI**	**GAMAN-NI**	**GAMAN**
Spring-Ball	-6.25	-6.40	-0.75	**1.21**
Stanford Drone	1.34	4.51	4.04	**6.12**
NBA	-9.54	0.51	-1.1	**0.80**

Implementation details For the generation network in GAMAN, we use multivariate Gaussian with diagonal covariance for both transition distribution π_{θ_e} and emission distribution π_{θ_x}. f_{θ_e} is parameterized by GRU. f_{θ_x} is parameterized by a 3-layer MLP with `ReLU` (Nair and Hinton 2010) activations followed by a linear output layer. Both f_{θ_s} and f_{θ_a} are parameterized by 3-layer MLPs with `ReLU` activations. For the inference network, we also use multivariate Gaussian distribution with diagonal covariance for π_{ϕ_e} which is parameterized by GRU, and f_{θ_s} and f_{θ_a} are inherited from the generation network. All sizes of hidden layers are 32 and model is optimized with Adam (Kingma and Ba 2014). We implement KF model using the pykalman (Duckworth 2013). For other models compared in experiments, we keep the original network structure and use a similar amount of parameters for fair comparison. We randomly split each dataset into the training/validation/test set with the ratio of 7:1:2, choose the best model weights on the validation set, and report the performance on the test set.

Quantitative Results

Prediction Table 2 shows the behavior forecasting results on these 3 datasets. We observe that, in most cases, our proposed GAMAN outperforms all the competitive baselines in terms of 4 metrics and achieves the best performance. Further, there is significant improvement from GAMAN-NI to GAMAN, which demonstrates that the proposed *attention mechanism* can efficiently capture the interactions in multi-agent system and provide better behavior forecasting results.
Inference Meanwhile, we compare the log-likelihood of generative models in Table 3. Models with higher log-likelihood value indicate they are fitted tighter on datasets. Our proposed GAMAN excels all generative models with the highest log-likelihood values on all 3 datasets.

Combined with the performance of prediction and inference, it is certain that GAMAN not only provides more accurate predictions but also offers preferable inference.

Interaction Analysis

To further understand the interpretability of GAMAN, especially agent vectors $\mathbf{e}_t$ and interactions $\mathbf{s}_t$, we conduct case study on 3 datasets. We use weighted interaction vectors $\mathbf{s}_t^k$ to get the pairwise relational interaction vectors (e.g. pairwise distance or other metrics) and visualize the relational interaction vectors. To acquire a reasonable visualization of the learned vectors, we projected vectors on the 2D plane using T-SNE (Maaten 2008). Figure 3 shows several concrete examples of learned vectors. The same group of agents and the same type of interactions share the same colors.

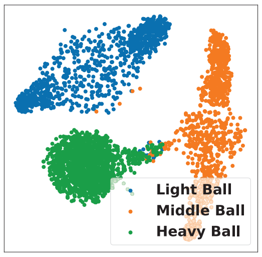
For **Spring-Ball**, we randomly choose 1000 frames from test set and get the corresponding vectors. Figure 3(a) and Figure 3(b) show the clusters of 3 groups of agents (balls) and 3 types of interactions (springs) among balls. Moreover, in **Stanford Drone**, we randomly select 500 frames from test set. Figure 3(c) and Figure 3(d) show the clusters of 3 groups of agents (e.g., bikers) and 3 types of interactions respectively. Finally, we randomly choose 500 frames from test set on **NBA**, Figure 3(e) and Figure 3(f) indicate 3 groups of agents (balls, offense and defense) and 3 types of interactions (ball-players, collaborations and hostilities). Conceivably, agent groups and interaction types identified by GAMAN are reasonable and convictive, which demonstrates its capability to handle heterogeneous systems and provide interpretable representations.

Summary

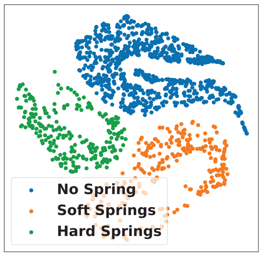
We proposed the *Generative Attentional Multi-Agent Network* (GAMAN) — a novel deep generative model designed for multi-agent systems, which approximates the underlying behavior generating process with a latent state space model and integrated with *attention mechanism* to capture agent interactions in a computationally efficient and interpretable fashion. Empirically we showed that GAMAN outperforms existing models on prediction accuracy and correctly identifies agent groups and interaction types in different systems. Given the increasing amount of data we collected everyday, GAMAN would empower us to discover new knowledge of real-world complicated multi-agent systems.

Acknowledgments

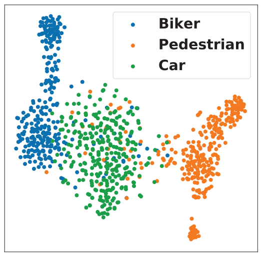
This research has been funded in part by NSF grants IIS-1254206. Thank Dr. Ming Zhang from Peking University for her support on this work.



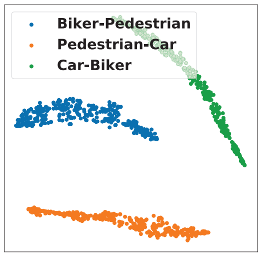
(a) $\mathbf{e}_t$ on **Spring-Ball**.



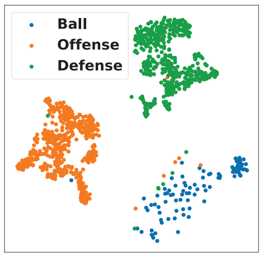
(b) $\mathbf{s}_t$ on **Spring-Ball**.



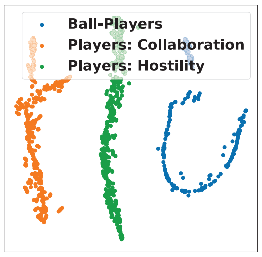
(c) $\mathbf{e}_t$ on **Stanford Drone**.



(d) $\mathbf{s}_t$ on **Stanford Drone**.



(e) $\mathbf{e}_t$ on **NBA**.



(f) $\mathbf{s}_t$ on **NBA**.

Figure 3: Visualization of learned agent vectors $\mathbf{e}_t$ and interaction vectors $\mathbf{s}_t$ on three datasets, indicating different agent groups and interaction types.

References

Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; and Savarese, S. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*.

Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D. J.; et al. 2016. Interaction networks for learning about objects, relations and physics. In *NIPS*, 4502–4510.

Che, Z.; Purushotham, S.; Li, G.; Jiang, B.; and Liu, Y. 2018. Hierarchical deep generative models for multi-rate multivariate time series. In *ICML*, 783–792.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*.

Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A. C.; and Bengio, Y. 2015. A recurrent latent variable model for sequential data. In *NIPS*.

Duckworth, D. 2013. pykalman. https://pykalman.github.com.

Grover, A.; Al-Shedivat, M.; Gupta, J. K.; Burda, Y.; and Edwards, H. 2018. Learning policy representations in multiagent systems. *arXiv:1806.06464*.

Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S.; and Alahi, A. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*.

Helbing, D., and Molnar, P. 1995. Social force model for pedestrian dynamics. *Physical review E* 51(5):4282.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hoshen, Y. 2017. Vain: Attentional multi-agent predictive modeling. In *NIPS*, 2701–2711.

Ivanovic, B.; Schmerling, E.; Leung, K.; and Pavone, M. 2018. Generative modeling of multimodal multi-human behavior. *arXiv:1803.02015*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv:1312.6114*.

Kipf, T.; Fetaya, E.; Wang, K.-C.; Welling, M.; and Zemel, R. 2018. Neural relational inference for interacting systems. *arXiv:1802.04687*.

Krishnan, R. G.; Shalit, U.; and Sontag, D. 2016. Structured inference networks for nonlinear state space models. *arXiv:1609.09869*.

Le, H. M.; Yue, Y.; Carr, P.; and Lucey, P. 2017. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1995–2003. JMLR. org.

Lee, N.; Choi, W.; Vernaza, P.; Choy, C. B.; Torr, P. H.; and Chandraker, M. 2017. Desire: Distant future prediction in dynamic scenes with interacting agents. In *CVPR*, 336–345.

Linou, K. 2016. NBA dataset from raw SportVU logs. https://github.com/linouk23/NBA-Player-Movements.

Liu, Y.; Yu, R.; Zheng, S.; and Yue, Y. 2018. Long range sequence generation via multiresolution adversarial training.

Maaten, L. v. d. 2008. t-distributed stochastic neighbor embedding in scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

Manh, H., and Alaghband, G. 2018. Scene-lstm: A model for human trajectory prediction. *arXiv:1808.04018*.

Mehran, R.; Oyama, A.; and Shah, M. 2009. Abnormal crowd behavior detection using social force model. In *CVPR*, 935–942. IEEE.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv:1401.4082*.

Robicquet, A.; Sadeghian, A.; Alahi, A.; and Savarese, S. 2016. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*.

Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. In *NIPS*.

Vemula, A.; Muelling, K.; and Oh, J. 2018. Social attention: Modeling attention in human crowds. In *ICRA*, 1–7. IEEE.

Xu, Y.; Piao, Z.; and Gao, S. 2018. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *CVPR*, 5275–5284.

Yeh, R. A.; Schwing, A. G.; Huang, J.; and Murphy, K. 2019. Diverse generation for multi-agent sports games. In *CVPR*, 4610–4619.

Zhan, E.; Zheng, S.; Yue, Y.; and Lucey, P. 2018. Generative multi-agent behavioral cloning. *arXiv:1803.07612*.