Diverse Generation for Multi-agent Sports Games

Raymond A. Yeh Alexander G. Schwing Jonathan Huang† Kevin Murphy†
University of Illinois at Urbana-Champaign
†Google Research
yeh17@illinois.edu, aschwing@illinois.edu, jonathanhuang@google.com, kpmurphy@google.com

Abstract

In this paper, we propose a new generative model for multi-agent trajectory data, focusing on the case of multi-player sports games. Our model leverages graph neural networks (GNNs) and variational recurrent neural networks (VRNNs) to achieve a permutation equivariant model suitable for sports. On two challenging datasets (basketball and soccer), we show that we are able to produce more accurate forecasts than previous methods. We assess accuracy using various metrics, such as log-likelihood and "best of N" loss, based on N different samples of the future. We also measure the distribution of statistics of interest, such as player location or velocity, and show that the distribution induced by our generative model better matches the empirical distribution of the test set. Finally, we show that our model can perform conditional prediction, which lets us answer counterfactual questions such as "how will the players move if A passes the ball to B instead of C?"

1. Introduction

Multi-agent systems are widespread in the real world. In many applications, we would like to learn a model of the interaction between the agents, which we can use to predict plausible future behaviors. The challenges involve modeling the interactions in a parsimonious way, and coping with the inherent multi-modality of future prediction.

In this paper, we focus on the case of modeling trajectory data collected from multi-player sports games, namely basketball [41] and soccer [23]. Our model uses graph-structured variational RNNs, which are based off methods from graph neural networks (reviewed in [3]) with variational recurrent neural networks [7]. The use of a graph, with one node per agent, ensures the model is permutation equivariant, which is necessary since each game segment corresponds to an unordered set of K trajectories, where K is the number of players in the game. Previous works, such as [10, 44], use various heuristics to assign players to roles, thereby fixing an agent ordering across games, but we show that our approach produces better results, since roles can change dynamically. We also show that the use of a variational RNN, as opposed to a deterministic RNN, is helpful (at least in the case of basketball) since there is often stochasticity in the latent dynamics of the players beyond the kind of short-term variation one can capture with observation-level noise.

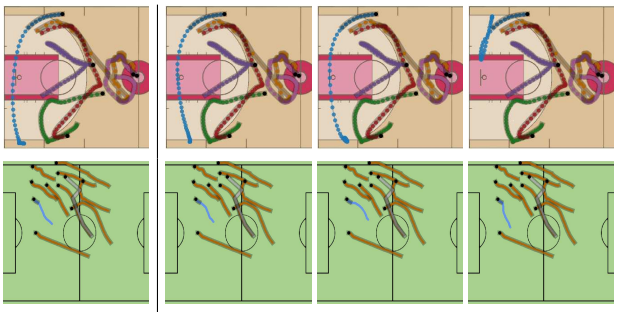

Figure 1. Illustration of basketball (top) and soccer (bottom) data. We visualize possible futures, predicted from our model, for the blue player given ground truth locations of the other agents. The black point indicates the starting position for each agent; the gray-highlight indicates which of the locations are given.

In addition to improving the basic model, we propose a variety of metrics for evaluating the quality of generative forecasting models (*cf*. [35]). In particular, beyond the standard use of log-likelihood and "best of K" loss [5, 26, 31], we propose to evaluate the marginal distributions of features of interest, such as the player locations and speeds, based on ground truth trajectories as well as generated trajectories. We show that simple models, such as constant velocity, can outperform more complex models, such as RNNs, when judged by the standard metrics, but not when judged by these distributional similarity metrics.

Finally, we consider the case of conditional prediction, in which some aspects of the future are observed (and potentially manipulated), and some are predicted. For example, Fig. 1 shows the predicted trajectory of the blue player given the other players, and Fig. 6 shows the predicted trajectories of all the players if we "intervene" and modify the trajectory of the ball. This is a step towards answering counter-factual

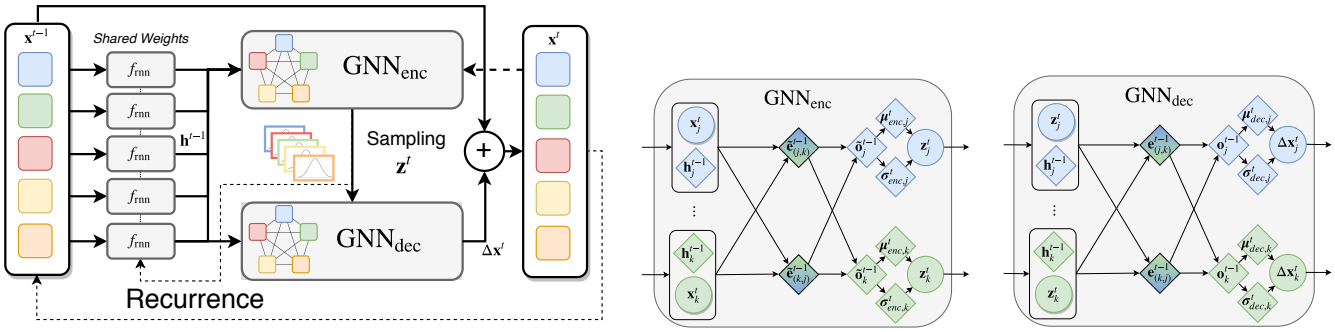

Figure 2. (a): Illustration of the overall approach. Each agents' trajectory is processed through a GRU with shared parameters, while keeping its own individual recurrent state. The graph encoder and decoder models the relationship between agents, and finally outputting the predicted movement of each agent. (b,c): Detailed illustration of the GNN_{enc} and $\text{GNN}_{\text{decoder}}$ on two agents; We use a diamond shape to indicate the value is deterministic, and use a circle to indicate the variable is stochastic. e represent edge states and o represent updated node (output) states.

queries, such as "What would happen if player A passed the ball to player B instead of player C?"

In summary, we propose an improved generative model for multi-player sports data. Moreover, we perform extensive quantitative and qualitative analysis of this model and compare to prior work on two challenging datasets, namely basketball and soccer. [1] [2]

2. Graph Variational RNN for Sports

Problem formulation: Let $\mathbf{x}_k^t \in \mathbb{R}^2$ denote the 2-dimensional location of agent k at time t, and let $\mathbf{x}_k = (\mathbf{x}_k^1, \ldots, \mathbf{x}_k^T)$ be a corresponding trajectory. Finally, let $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ be an unordered set of trajectories, corresponding to one segment of a game, *i.e.*, a play, where K is the number of agents in this segment, and let $\mathcal{D} = \{\mathbf{x}\}$ subsume all the segments. Our goal is to learn a model of the trajectories, *i.e.*, $p(\mathbf{x}_{1:K}^{1:T})$.

VRNNs: Our model builds on the Variational RNN (VRNN) [7]. VRNNs have three types of variables: the observed output $\mathbf{x}^t = \mathbf{x}_{1:K}^t$, the stochastic variational auto-encoder (VAE) state $\mathbf{z}^t = \mathbf{z}_{1:K}^t$, and the deterministic RNN hidden state $h^t = h_{1:K}^t$, which summarizes the past history of observations $\mathbf{x}^{\leq t}$ and stochastic choices $\mathbf{z}^{\leq t}$. At a high level, a VRNN is a VAE at every time step, characterized by the following distributions and the RNN update equation:

$$
\begin{aligned}
p_\theta(\mathbf{z}^t|\mathbf{x}^{<t}, \mathbf{z}^{<t}) &= \mathcal{N}(\mathbf{z}^t|\boldsymbol{\mu}_{\text{pri}}^t, (\boldsymbol{\sigma}_{\text{pri}}^t)^2) \quad \text{(prior)},\\
q_\phi(\mathbf{z}^t|\mathbf{x}^{\leq t}, \mathbf{z}^{<t}) &= \mathcal{N}(\mathbf{z}^t|\boldsymbol{\mu}_{\text{enc}}^t, (\boldsymbol{\sigma}_{\text{enc}}^t)^2) \quad \text{(inference)},\\
p_\theta(\mathbf{x}^t|\mathbf{x}^{<t}, \mathbf{z}^{\leq t}) &= \mathcal{N}(\mathbf{x}^t|\boldsymbol{\mu}_{\text{dec}}^t, (\boldsymbol{\sigma}_{\text{dec}}^t)^2) \quad \text{(generation)},\\
h^t &= f_{\text{rnn}}(\mathbf{x}^t, \mathbf{z}^t, h^{t-1}) \quad \text{(recurrence)},
\end{aligned}
$$

where

$$
\begin{aligned}
[\boldsymbol{\mu}_{\text{pri}}^t, \boldsymbol{\sigma}_{\text{pri}}^t] &= f_{\text{pri}}(h^{t-1}; \theta),\\
[\boldsymbol{\mu}_{\text{enc}}^t, \boldsymbol{\sigma}_{\text{enc}}^t] &= f_{\text{enc}}(\mathbf{x}^t, h^{t-1}; \phi),\\
[\boldsymbol{\mu}_{\text{dec}}^t, \boldsymbol{\sigma}_{\text{dec}}^t] &= f_{\text{dec}}(\mathbf{z}^t, h^{t-1}; \theta),
\end{aligned}
$$

and $\mathcal{N}(\cdot|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ denotes a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\text{diag}(\boldsymbol{\sigma}^2)$. Here, f_{pri}, f_{enc}, and f_{dec} are deep nets corresponding to the prior network, encoder and decoder with learnable parameters ϕ and θ.

VRNNs are trained by maximizing the evidence lower bound (ELBO):

$$
\sum_{\mathbf{x}\in\mathcal{D}} \sum_t \mathbb{E}_{q_\phi(\mathbf{z}^t|\mathbf{x}^{\leq t}, \mathbf{z}^{<t})} \Big[\log p_\theta(\mathbf{x}^t|\mathbf{x}^{<t}, \mathbf{z}^{\leq t}) - D_{\text{KL}}(q_\phi(\mathbf{z}^t|\mathbf{x}^{\leq t}, \mathbf{z}^{<t}) || p_\theta(\mathbf{z}^t|\mathbf{x}^{<t}, \mathbf{z}^{<t})) \Big]. \quad (1)
$$

With Gaussians for the prior and posterior, we can leverage the reparameterization trick to optimize this objective with stochastic gradient descent (SGD), as discussed in [19].

Consistent representation: This standard VRNN formulation implicitly encapsulates the *ordering of the agents* in the index k. However our data consists of *unordered sets of trajectories*, and so does not contain this information. In this section, we discuss the importance of having a consistent ordering.

Consider a dataset with two plays, each with two agents in an "arbitrary agent order":

$$
\mathcal{D}_1 = \Big\{ \{[1, \mathbf{x}_1^{(1)}], [2, \mathbf{x}_2^{(1)}]\}, \{[1, \mathbf{x}_1^{(2)}], [2, \mathbf{x}_2^{(2)}]\} \Big\}.
$$

The superscript in parenthesis indicates the play number, and the ordering index is paired together with the agent location in brackets, *i.e.*, $\mathbf{x}_1$ is index 1 and $\mathbf{x}_2$ is index 2 for both examples. We also consider another agent order:

$$
\mathcal{D}_2 = \Big\{ \{[1, \mathbf{x}_1^{(1)}], [2, \mathbf{x}_2^{(1)}]\}, \{[2, \mathbf{x}_1^{(2)}], [1, \mathbf{x}_2^{(2)}]\} \Big\}.
$$

[1] Data Source: STATS, copyright 2019. Available: https://www.stats.com/data-science/.

[2] We note that, as far as we know, our paper is the first to actually model the dynamics of the ball in basketball. Previous methods ignored the ball and focused only on the players.

Typically, in a deep net, the representation of $\mathbf{x}$ is the *concatenation* of $\mathbf{x}_1$ and $\mathbf{x}_2$ along the agent dimension. Observe that the representation for $\mathcal{D}_1$ and $\mathcal{D}_2$ is different and nothing encourages the same model probabilities.

In order to build a consistent model, *i.e.*, the modeled probability is the same for $\mathcal{D}_1$ and $\mathcal{D}_2$, we need to handle this ordering discrepancy. Prior works [10, 27, 34] have proposed to solve this consistency problem by "sorting" the agents. At a high level, similar behaving agents are placed at the same agent index. Instead, here, we advocate for a permutation-equivariant representation, *i.e.*, a permutation at the input leads to the same permutation at the output. Consequently, the change in agent ordering *does not* impact the modeled probability. One family of models that satisfy this permutation equivariance property are graph networks, introduced in the following section.

Graph networks: The basic idea of graph neural nets, summarized in [3], is as follows: we start with a feature vector for each node v_i. We then derive a feature vector for each edge e_{ij} based on the nodes it is connected to. Afterwards we send the edge feature vectors as "messages" to each of the connected nodes to compute their new output state o_i. More formally, a single round of message passing operations of a graph net are characterized below:

$$v \to e: \quad \mathrm{e}_{(i,j)} = f_e([\mathrm{v}_i, \mathrm{v}_j]), \tag{2}$$

$$e \to v: \quad \mathrm{o}_i = f_v\Big(\sum_{j \in N(i)} \mathrm{e}_{(i,j)}\Big), \tag{3}$$

where v_i is the initial state of node i, o_i is the updated (output) representation for node i, $N(i)$ is the set of neighbors of node i, $\mathrm{e}_{(i,j)}$ is the representation for edge (i,j), and lastly f_e and f_v are deep networks.

In summary, a GNN takes in feature vectors $\mathrm{v}_{1:K}$, and an adjacency matrix, A, and outputs a vector for each node, $\mathrm{o}_{1:K}$, that is, $\mathrm{o}_{1:K} = \mathrm{GNN}(\mathrm{v}_{1:K})$. Here, we consider fully connected graphs only, and thus drop A for simplicity.

Observe that the operations of the GNN satisfy the permutation equivariance property as the edge construction is symmetric between pairs of nodes and the summation operator ignores the ordering of the edges (*cf*. [42]).

Graph VRNNs: We now describe our proposed Graph Variational RNN (GVRNN) model. An overview of the proposed model is illustrated in Fig. 2. Our model has independent RNNs and observations for each agent. We model interactions between them at each step using GNNs, where each node is an agent, and the graph is fully connected.[3]

More precisely, the distributions in Eq. (1) and RNN update are define as follows:

$$p_\theta(\mathbf{z}^t|\mathbf{x}^{<t}, \mathbf{z}^{<t}) = \prod_k \mathcal{N}(\mathbf{z}^t|\boldsymbol{\mu}^t_{\text{pri},k}, (\boldsymbol{\sigma}^t_{\text{pri},k})^2),$$

$$q_\phi(\mathbf{z}^t|\mathbf{x}^{\le t}, \mathbf{z}^{<t}) = \prod_k \mathcal{N}(\mathbf{z}^t|\boldsymbol{\mu}^t_{\text{enc},k}, (\boldsymbol{\sigma}^t_{\text{enc},k})^2),$$

$$p_\theta(\mathbf{x}^t|\mathbf{x}^{<t}, \mathbf{z}^{\le t}) = \prod_k \mathcal{N}(\mathbf{x}^t|\boldsymbol{\mu}^t_{\text{dec},k}, (\boldsymbol{\sigma}^t_{\text{dec},k})^2),$$

$$h^t_k = f_{\text{rnn}}(\mathbf{x}^t_k, \mathbf{z}^t_k, h^{t-1}_k),$$

where

$$[\boldsymbol{\mu}^t_{\text{pri},1:K}, \boldsymbol{\sigma}^t_{\text{pri},1:K}] = \mathrm{GNN}_{\text{pri}}(h^{t-1}_{1:K}),$$

$$[\boldsymbol{\mu}^t_{\text{enc},1:K}, \boldsymbol{\sigma}^t_{\text{enc},1:K}] = \mathrm{GNN}_{\text{enc}}([\mathbf{x}^t_{1:K}, h^{t-1}_{1:K}]),$$

$$[\boldsymbol{\mu}^t_{\text{dec},1:K}, \boldsymbol{\sigma}^t_{\text{dec},1:K}] = \mathrm{GNN}_{\text{dec}}([\mathbf{z}^t_{1:K}, h^{t-1}_{1:K}]).$$

In Fig. 2 (b,c) we illustrate the computation of the encoder and decoder for a two agent case. Here, the prior network, encoder, and decoder are chosen to be GNNs. Note that, while the RNN state for agent k, $\mathbf{h}^t_k$, only depends directly on quantities related to k (*i.e.*, $\mathbf{h}^t_k$, $\mathbf{z}^t_k$ and $\mathbf{x}^t_k$), these quantities do depend indirectly on the other agents through the GNNs.

In practice, the model generates the change in coordinates at each step, $\Delta\mathbf{x}^t_k$, and then computes $\mathbf{x}^t_k = \mathbf{x}^{t-1}_k + \Delta\mathbf{x}^t_k$. Also, we use as input to the GNNs the previous observations, $\mathbf{x}^{t-1}_{1:K}$, as well as the previous RNN states, $\mathbf{h}^{t-1}_{1:K}$. This is a form of "skip connection," which we found to slightly improve performance. We have also tried adding a GNN to model direct interaction between the $\mathbf{h}^t_k$ nodes, but this did not improve performance, and slowed down training, so we omit this.

Adding type information: The graph network is completely exchangeable between agents, but sometimes this is too strong an assumption. For example, players and balls often move very differently. One way to achieve partial exchangeability is to use type information. Specifically, we can use a different kind of node function f_v for each type of node, and a different kind of edge function f_e for every pair of node type. However, to avoid the quadratic explosion in the number of parameters, we choose instead to use a single edge function. But to make it "type aware" we add an embedding $\mathrm{t}_{i,j}$ depending on the edge type and node type t_i, *e.g.*, a one-hot vector. Consequently, our message passing operations are:

$$v \to e: \quad \mathrm{e}_{(i,j)} = f_e([\mathrm{v}_i, \mathrm{v}_j, \mathrm{t}_{i,j}]), \tag{4}$$

$$e \to v: \quad \mathrm{o}_i = f_v\Big(\Big[\sum_{j \in N(i)} \mathrm{e}_{(i,j)}, \mathrm{t}_i\Big]\Big). \tag{5}$$

In the experiments, we show that adding type information improves the performance.

[3] Although the graph is fully connected, the "effective strength" of each edge is computed dynamically, which is similar to the approach of graph attention networks [36].

Approach	Order	NLL	L_2 (Avg)	L_2 (Best)	Max-L_2 (Best)	Miss Rate (%)	Cond. L_2 (Best)
Velocity	-	-	10.40 ± .03	10.40 ± .03	17.20 ± .04	74.2 ± .10	13.87 ± .06
RNN	Random	-2171	13.72 ± .04	13.72 ± .04	25.21 ± .09	82.8 ± .04	14.23 ± .06
RNN	Template	-2308	11.46 ± .05	11.46 ± .05	20.93 ± .12	79.7 ± .05	12.04 ± .04
RNN	Tree	-2318	11.55 ± .05	11.55 ± .05	21.03 ± .12	80.2 ± .05	11.69 ± .04
GRNN-Diag	Equivariant	-2252	10.75 ± .02	10.75 ± .02	16.97 ± .04	82.7 ± .05	13.10 ± .05
GRNN-Full	Equivariant	-2363	12.33 ± .03	12.33 ± .03	19.53 ± .05	86.8 ± .05	13.65 ± .05
GRNNT-Diag	Equivariant	-2374	09.70 ± .02	09.70 ± .02	16.73 ± .05	75.9 ± .05	12.18 ± .05
GRNNT-Full	Equivariant	-2264	11.24 ± .03	11.24 ± .03	19.07 ± .07	79.6 ± .06	12.20 ± .04
VRNN	Random	$\leq$ -2667	10.52 ± .12	09.59 ± .06	16.44 ± .21	76.0 ± .05	11.36 ± .05
VRNN	Template	$\leq$ -2750	**09.44** ± .01	09.02 ± .02	15.51 ± .06	71.5 ± .08	09.79 ± .04
VRNN	Tree	$\leq$ -2748	09.88 ± .01	09.40 ± .02	16.05 ± .05	72.8 ± .05	09.83 ± .04
GVRNN-Diag	Equivariant	$\leq$ -2814	11.09 ± .01	08.86 ± .02	14.26 ± .03	72.4 ± .08	09.02 ± .03
GVRNN-Full	Equivariant	$\leq$ -2814	10.71 ± .01	***08.39*** ± .02	**13.46** ± .03	***69.1*** ± .08	***08.63*** ± .03
GVRNNT-Diag	Equivariant	$\leq$ ***-2818***	***09.51*** ± .01	08.87 ± .02	15.17 ± .06	70.6 ± .08	10.63 ± .04
GVRNNT-Full	Equivariant	$\leq$ **-2832**	10.37 ± .01	**08.26** ± .02	**13.46** ± .04	**68.6 ± .06**	**07.88** ± .03

Table 1. Quantitative results on basketball dataset modeling offensive players and ball. We report mean and standard deviation of the mean (sample size is 13,845). Lower numbers are better. Bold is the best and italics is second best. For conditional generation (last column), the task is to predict the ball's trajectory conditioned on the offensive players. The L_2 metrics have the units of feet.

3. Related Work

Multi-agent modeling for sports: Learning based methods have demonstrated success in predictive modeling for a variety of sports, including basketball, soccer, American football, water-polo, *etc*. Commonly employed formulations are regression, *e.g*., predicting future trajectories [9, 23, 25, 44], and classification, *e.g*., event detection [38], most of which leverage deep nets. More specifically, for basketball, [10] proposed to use a conditional variational autoencoder, which personalizes the agent's behavior by conditioning on player and team identity. However, a different network architecture is necessary for each time horizon prediction. A recurrent network solves this issue. In [43, 44] the challenge of modeling long sequential data is addressed by decomposing the task into "micro" and "macro" goals, capturing short-term and long-term behavior separately via a recurrent net. In [10, 43, 44], the agents are first pre-processed into a specific "order," using template-based or tree-based methods [27, 34]. The idea is to sort "similarly" behaving agents into the same bin so as to obtain a consistent representation. In contrast, we use a permutation-equivariant representation based on graphs, that avoids the needs for such preprocessing. We show below that this results in improved performance.

Pedestrian trajectory forecasting: There is a large literature on modeling of pedestrian movements. Most recent works, *e.g*., [1, 15, 24, 30, 40] focus on effectively aggregating information across a large number of people. For aggregation, specialized pooling modules are often used [1, 15, 24, 40]. Interestingly, [4] showed that a simple baseline of an RNN with an MLP decoder outperformed many of these prior works on the TrajNet benchmark [32].

Generative and time series models: In [11], autoregressive RNNs have shown good forecasting performance for simple 1d time series, such as sales data. However, for multi-dimensional data, the use of latent variables often gives improved performance, as shown in [7]. Various other time series models have been proposed. For example, [22] and [12] propose nonlinear stochastic state space models, where the latent states at each time step do not depend on past model outputs, unlike an RNN. In [17], a sequence-to-sequence model is presented, where there is only one latent sample per sequence. Similarly, [37] uses a conditional VAE, which predicts future pixel trajectories given a single image. In contrast, in our model, we have a latent state per time step with dependencies on the past model output, which follows the structure of a variational RNN in [7].

Graph neural networks: To model interactions between variables, graph neural networks (GNN) [33] have been proposed. A plethora of variants exist, including Interaction Network [2], Message Passing Network [13], Graph Convolutional Network [21] and others [16, 28, 36, 42]. A summary is provided in [3]. GNNs have recently been applied to many datasets, including sports trajectory data. In particular, [20] modeled pick-and-roll basketball data using a latent graph. However, the focus of their approach is to discover the underlying latent graph. Another relevant work is [16], who applied GNNs to soccer data. However, their model predicts from a single frame and does not model past trajectories, whereas we use temporal history and a stochastic model of the future.

4. Experimental Evaluation

In this section, we compare our model to various baselines on two datasets: modeling of basketball and soccer game trajectories.

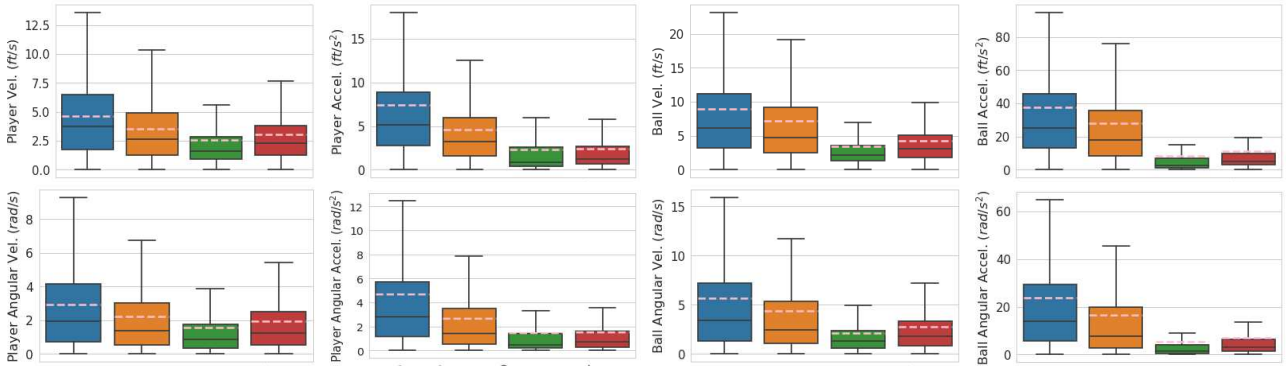

Figure 3. Quantitative results for basketball dataset with the distributional metrics evaluated on the offensive players and ball model visualized in box-plot, the dashed-line indicates the mean. Refer to the y-axis for the specific statistic being visualized. The barplots are ordered as follows: **Ground truth**, **GVRNNT-Full**, **GVRNNT-Diag**, **VRNN** with template ordering.

4.1. Models

We compare our approach to several deep learning baselines. For the non-permutation equivariant models, we evaluate on two ordering methods, a template-based method and a tree-based method [34].

Velocity: As a sanity check, we used velocity extrapolation as a simple baseline, *i.e.*, each of the agent's predictions is linearly extrapolated using its past observed velocity.

RNN: A recurrent neural net baseline implemented using a gated recurrent unit [6]. The model uses an MLP-decoder for prediction, rather than the output-gate of the RNN. This is a simple, yet effective baseline shown in [4].

VRNN: A variational version of the above RNN.

GVRNN (Ours): A graph variational recurrent neural net, either with a fully connected graph (Full) or a graph containing only self-loops, *i.e.*, a diagonal adjacency matrix (Diag). In the latter case, we do not model interaction between the agents; the model is still permutation-equivariant.

GVRNNT (Ours): GVRNN extended with agent types.

Training Details: We train all the models using the Adam optimizer [18] with default parameters. The initial learning rate is 0.0005. We decay the learning rate exponentially by a factor of 0.999 per epoch. All models are trained using teacher forcing [39], and the reparameterization-trick with random sample size of 1 is used. All fully connected layers are initialized using Xavier initialization [14]. To prevent over-fitting, we select the best performing model using log-likelihood on the validation set.

4.2. Evaluation Metrics:

Due to the difficult nature of evaluating a generative model, we evaluate on several different metrics to demonstrate the efficacy of our approach.

Negative Log-Likelihood (NLL): We report the negative log-likelihood on the test set, $-\sum_{t=2}^{T} \log p(x^t|x^{1:t-1})$. For the deterministic baselines the NLL is exact. For the variational models, we report the negative ELBO, which is an upper bound on the NLL (indicated by the $\leq$ symbol). Unfortunately, such bounds are not directly comparable between methods, and tighter bounds do not necessarily lead to better performance [29], so we also report several other more informative metrics detailed below.

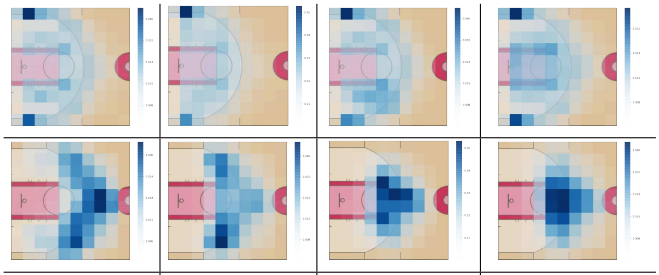

Figure 4. **Top:** Average location of players sampled from leave-one-out generation for each model. **Bottom:** Average location of the ball sampled conditioned on the players from each model. At a glance, GVRNNT-Full's map matches the ground truth more closely for both player and ball, which is confirmed with the quantitative evaluation in Tab. 2.

Approach	Player-SKL	Player-JS	Ball-SKL	Ball-JS
VRNN-Template	0.0068	0.0017	0.3556	0.0954
GVRNNT-Diag	0.0084	0.0020	0.3985	0.1080
GVRNNT-Full	**0.0036**	**0.0009**	**0.2658**	**0.0713**

Table 2. Basketball's quantitative comparison of each models' heatmap in Fig. 4 with the ground truth. We evaluate on two different divergences, where SKL refers to Symmetric KL-divergence, and JSD refers to Jensen-Shannon divergence.

Mean and best L_2-error: We evaluate the models on the task of future prediction, *i.e.*, conditioned on the first 10 frames of all agents' trajectories we predict an additional 40 frames. We report the L_2-error (in feet or meters) between the predicted and the ground truth. Due to the multimodal nature of the system, for each test case, we randomly sample $N = 10$ trajectories and report the *average or the best* of the samples. More precisely, we compute the L_2-error between the ground truth and the n'th generated sample $\hat{x}^n$ using:

$$L_2^n = \frac{1}{T \cdot K} \sum_{t=1}^{T} \sum_{k=1}^{K} ||\hat{\mathbf{x}}_k^{n,t} - \mathbf{x}_k^t||_2.$$

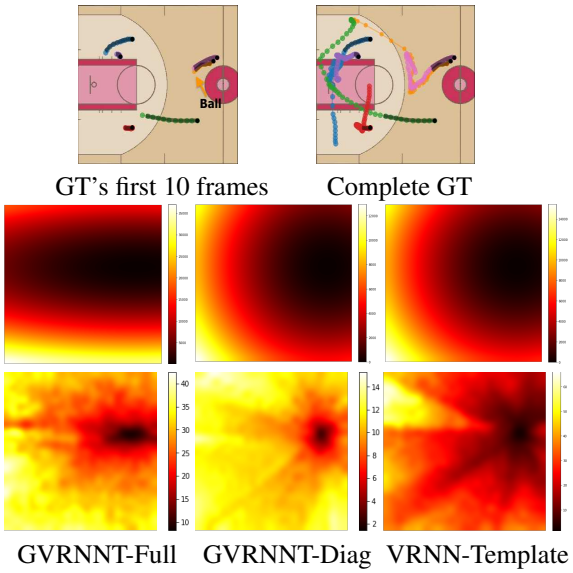

Figure 5. **First row**: We visualize the first 10 and 50 frames of the trajectory (ball is in orange). **Second row**: negative log posterior predictive for ball location at $t = 11$ for for GVRNNT-Full, GVRNNT-Diag and VRNN-Template. **Third row**: Bayesian surprise for ball location at $t = 11$ for GVRNNT-Full, GVRNNT-Diag and VRNN-Template.

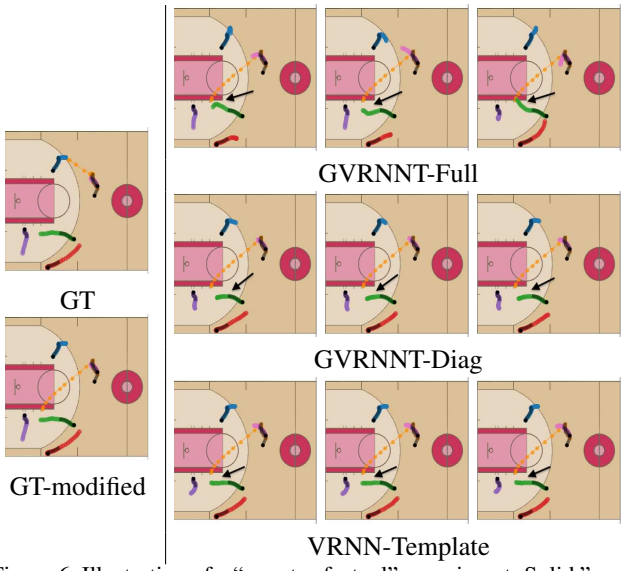

Figure 6. Illustration of a "counter-factual" experiment. Solid "vapor trails" correspond to past trajectories until the time when we perform an intervention by modifying ball trajectory (orange), colored trails correspond to subsequent trajectories. We then compute 3 different future trajectories for all the players for each model. See text for details.

We then report the average, $\frac{1}{N}\sum_n L_2^n$, and the best, $\min_n L_2^n$, as is standard practice [5, 26, 31].

Max L_2-error (Best): We also compute the average, over agents, maximum L_2-error between the prediction and the ground truth trajectory:

$$M^n = \frac{1}{K}\sum_{k=1}^{K} \max_{t=1}^{T} ||\hat{\mathbf{x}}_k^{n,t} - \mathbf{x}_k^t||_2.$$

For each test case, we sample $N = 10$ random trajectories from the model and report the best, $\min_n M^n$.

Miss rate: This denotes the percentage of predictions having an L_2-error greater than 3 feet for basketball and 1 meter for soccer. This is reported on the best out of 10 random samples per example.

Conditional L_2-error (Best): We evaluate on conditional generation. For example, we condition on all the players' trajectories and predict the ball trajectory. We report the lowest L_2 error out of 10 random samples for each test case. This metric is used to evaluate whether the model learned relations among the agents in the system.

Distributional metrics: To measure properties of the overall distribution of trajectories $\mathbf{x} = \mathbf{x}_{1:K}^{1:T}$, we compute the marginal distributions of 8 univariate statistics, denoted $\phi_1(\mathbf{x}), \ldots, \phi_8(\mathbf{x})$ namely: linear/ angular velocity/ acceleration of player/ ball; we also compute the marginal distribution over agent locations, $\phi_l(\mathbf{x}) \in \mathbb{R}^2$. We then compare the distribution of these statistics induced by various generative models to the true (empirical) distribution induced by the test set. We do this informally via boxplots of the distributions $p_\theta(\phi_i(\mathbf{x}))$ and $p_*(\phi_i(\mathbf{x}))$ for each of the 8 univariate statistics, where p_θ is a learned model and p_* is the true (empirical) test distribution. For the 2d location distributions, we use heatmaps as a visualization method, and quantitatively compare the distributions using two similarity metrics, namely symmetric Kullback-Leibler (KL) divergence and Jensen-Shannon divergence.

4.3. Modeling Basketball Dynamics

We use the basketball dataset from [41, 43], which contains tracking trajectories of professional basketball players and the ball. We use the pre-processed version of the dataset. Each example is sampled at 6Hz for 50 frames (roughly 8 seconds), with the offense team always going towards the left-side of the court. In total the dataset contains 107,146 training and 13,845 test examples. The data is centered and normalized to be within $[-1, 1]$.

Trajectory metrics: In Tab. 1, we report the quantitative results for modeling the offensive players and the ball. It can be observed that our GVRNNT-Full model outperforms all the others on all metrics except for average L_2. However, that metric is not particularly informative, since it does not measure sample diversity. We also observe that the constant velocity baseline outperforms the simple RNN with fixed ordering (template or tree based) on all metrics except for predicting the ball.

We performed several ablation studies verifying the necessity of each of the components. For the non-graph models, we observed that random ordering performs the worst, then tree-based ordering, then template-based ordering. Graph-based models outperform all of these non-

Approach	Order	NLL	L_2 (Avg)	L_2 (Best)	Max-L_2 (Best)	Miss Rate (%)	Cond. L_2 (Best)
Velocity	-	-	4.58 ± .02	4.58 ± .02	8.72 ± .05	80.0 ± .13	10.74 ± .09
RNN	Random	-5244	3.33 ± .01	3.33 ± .01	7.05 ± .03	69.5 ± .08	09.10 ± .09
RNN	Template	-5350	3.17 ± .01	3.17 ± .01	6.68 ± .04	67.6 ± .08	08.58 ± .09
RNN	Tree	-5353	3.20 ± .01	3.20 ± .01	6.74 ± .03	68.5 ± .08	08.18 ± .08
GRNN-Diag	Equivariant	-5221	4.37 ± .02	4.37 ± .01	8.40 ± .06	79.7 ± .09	10.99 ± .13
GRNN-Full	Equivariant	-5292	4.55 ± .03	4.55 ± .02	8.34 ± .05	78.9 ± .11	11.79 ± .13
GRNNT-Diag	Equivariant	-5333	***3.16*** ± .01	3.16 ± .03	6.54 ± .03	67.9 ± .09	08.77 ± .09
GRNNT-Full	Equivariant	-5349	**3.05** ± .01	**3.05** ± .01	**6.32** ± .03	67.3 ± .09	08.02 ± .07
VRNN	Random	≤ -5238	3.98 ± .01	3.97 ± .01	8.95 ± .04	72.6 ± .07	09.17 ± .09
VRNN	Template	≤ **-5579**	4.36 ± .09	3.38 ± .02	7.43 ± .11	***65.1*** ± .09	12.10 ± .14
VRNN	Tree	≤ ***-5569***	3.57 ± .01	3.42 ± .02	7.46 ± .04	65.8 ± .09	12.41 ± .14
GVRNN-Diag	Equivariant	≤ -5207	3.35 ± .01	3.11 ± .01	6.47 ± .03	65.2 .09	8.95 ± .08
GVRNN-Full	Equivariant	≤ -5369	3.25 ± .01	3.25 ± .01	6.80 ± .03	66.9 ± .09	8.55 ± .10
GVRNNT-Diag	Equivariant	≤ -5438	3.60 ± .01	3.11 ± .01	***6.39*** ± .04	**64.5** ± .08	***7.77*** ± .10
GVRNNT-Full	Equivariant	≤ -5325	3.17 ± .01	**3.07** ± .02	6.49 ± .08	66.8 ± .09	**7.61** ± .13

Table 3. Quantitative results on soccer dataset modeling offensive players (without the goal keeper) and ball. For conditional generation, the task is to predict the ball's trajectory conditioned on the offensive players. The L_2 metrics are measured in meters.

equivariant models. Furthermore, by comparing GVRNNT-Full and GVRNNT-Diag, we observed that the graph network is indeed learning relations between the agents. By comparing GVRNNT-Full and GVRNN-Full, we demonstrate that including agent types leads to further improvement in performance. Finally, we see that variational models outperform ones that only have stochasticity at the output nodes.

Distributional metrics: Next, we report the distributional metrics. Fig. 3 shows boxplots illustrating the distribution of the 8 statistics of interest. In general, our GVRNNT-Full model (orange) better matches the true distribution (blue) than GVRNNT-Diag (green), which is comparable in performance to a vanilla VRNN (red). Note that although the constant velocity baseline performs well for the metrics in Tab. 1, it performs poorly in terms of boxplots for certain statistics, such as the acceleration of an agent (results not shown). This illustrates the importance of using multiple performance metrics.

Location heatmaps: Fig. 4 shows the 2d marginal distribution of the agent locations as a heatmap. These distributions are generated from the averages of leave-one-out conditional generation, *i.e.*, predict an agent's trajectory conditioned on all other agents. Visually it appears that GVRNNT-Full better matches the true marginal distribution than GVRNNT-Diag and VRNN-Template. We quantify this in Tab. 2 where we report on two different divergence metrics, symmetric KL and Jensen-Shannon, applied to the heatmaps.

Qualitative samples: In Fig. 1, we visualize random samples generated from our model. We observe that our approach successfully generates a variety of possible trajectories, consistent with our expectation that the trajectories are multimodal in nature.

Predicting the ball location: Most prior works on basketball modeling, such as [10, 43, 44], only predict player locations. Here we assess the model's ability to predict future ball locations, which is much harder. To visualize this, we plot the negative log posterior predictive, *i.e.*, $-\log p(x_b^t|\mathbf{x}_{1:K}^{1:t-1})$, where b is the ball index, for each possible ball location (value of x_b^t). This is shown in Fig. 5 for 3 different models: GVRNNT-Full, GVRNNT-Diag and VRNN. Unfortunately these plots are not particularly informative, due to the Gaussian assumption of $p(x_b^t|\mathbf{x}_{1:K}^{1:t-1})$. A more insightful way to measure the predictive ability of latent variable models is to use the "Bayesian surprise" metric of [8], defined as follows:

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z}^t|\mathbf{x}^{\leq t}, \mathbf{z}^{<t})||p_\theta(\mathbf{z}^t|\mathbf{x}^{<t}, \mathbf{z}^{<t})).$$

This indicates how "surprised" the model is at observing $\mathbf{x}^t$ given past observations $\mathbf{x}^{<t}$. We use $\mathbf{x}^t = (\mathbf{x}_b^t, \mathbf{x}_{1:P}^t)$ where we clamp $\mathbf{x}_b^t$ to all possible ball locations, and using the ground truth locations for the K players. The results are shown in Fig. 5 (third row). Without modeling the interactions (middle), the surprise as a function of future ball location behaves like a Gaussian decay, with a higher density along the current velocity direction. However, the GVRNNT-full model (left) takes into account the player position and models a higher density towards the players. This demonstrates that the graph component successfully models the agents' relations.

Counter-factual predictions: Lastly, we assess the abilities for "counter-factual" reasoning by modifying the ball trajectory. In particular, in Fig. 6, we modified the ground truth so that instead of passing the ball (dotted orange trajectory) to the blue player (top), the pass goes to the green player (bottom). We observe that with the GVRNNT-Full model, the green player runs towards the ball. Instead, in the GVRNNT-Diag and VRNN model, the green player simply follows the original trajectory.

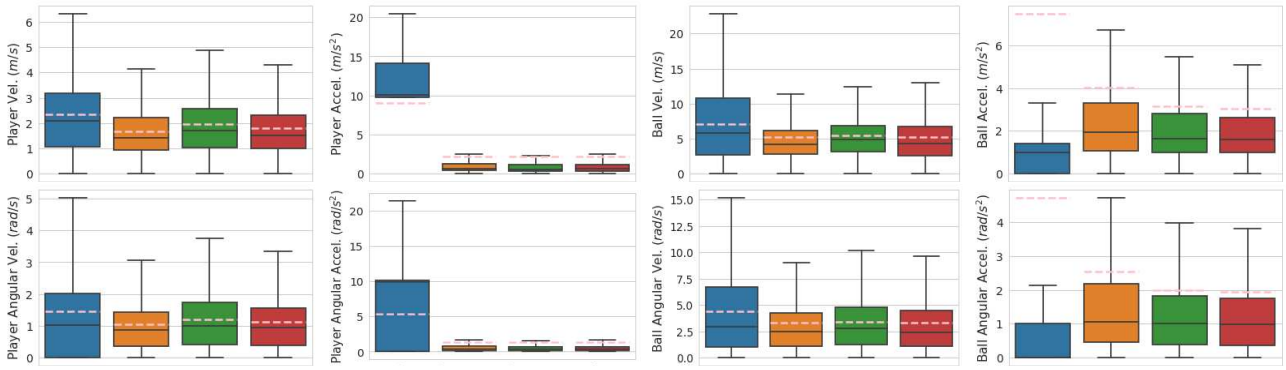

Figure 7. Quantitative results for soccer dataset with the distributional metrics evaluated on the offensive players and ball model visualized in box-plot, the dashed-line indicates the mean. Refer to the y-axis for the specific statistic being visualized. The barplots are ordered as follows: **Ground truth**, **GVRNNT-Full**, **GVRNNT-Diag**, **RNN** with template ordering.

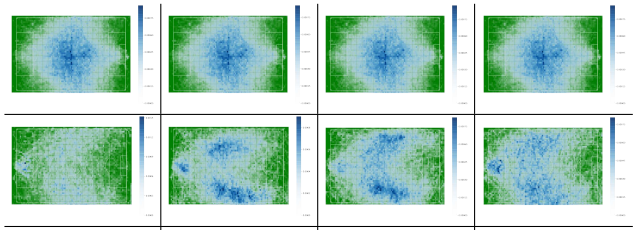

Figure 8. **Top:** Average location of soccer players sampled from leave-one-out generation for each model. **Bottom:** Average location of the ball sampled conditioned on the players from each model. We provide quantitative comparisons in Tab. 4.

Approach	Player-SKL	Player-JS	Ball-SKL	Ball-JS
RNN-Template	0.0018	0.00049	0.2136	0.0708
GRNNT-Full	**0.0016**	0.00045	0.1910	0.0571
GVRNNT-Full	0.0017	**0.00044**	**0.1853**	**0.0535**

Table 4. Soccer's quantitative comparison of each models' heatmap with the ground truth evaluated on the offensive players and the ball.

4.4. Modelling Soccer Dynamics

To demonstrate the generality of our approach, we apply it to the soccer dataset [23]. It contains trajectories of soccer players and the ball from a professional soccer league's game play. A total of 7500 sequences are in the training and the test set. As the split does not contain a validation set, we randomly sampled 10% of the training set to be the validation set. The sequences are of variable length sampled at 10Hz. We preprocess the data into segments of length 50 by using a sliding window with 50% overlap on both the training and test set. The trajectories are centered and normalized to be in the range of $[-1, 1]$. We do not model the goal-keeper as the goal keepers tend to move little.

Trajectory metrics: We evaluate the predictions for the offensive players and the ball. The results are shown in Tab. 3. We observe that tree-based ordering performs on par with template-based ordering, and permutation equivariant models with edge types outperform all baselines. Surprisingly, variational models did not outperform non-variational models on all metrics, unlike for basketball. We hypothesize that soccer dynamics, due to the bigger field and higher sampling rate, contain less stochasticity. Players tend to run more linearly compared to basketball, see Fig. 1. This is also supported by the results in Tab. 4, where we show that the variational model does better than the non-variational model at representing the distribution of ball locations, which is quite complex, but the two methods perform similarly when representing the player locations, which is much simpler.

Distributional metrics: In Fig. 7, we show the boxplots of the 8 statistics for the soccer dataset. In general, all 3 models perform similarly. None of them properly model the player's acceleration, perhaps because rapid acceleration is a sparse and bursty phenomenon not well captured by a Gaussian likelihood.

Location Heatmaps: In Fig. 8, we show the marginal distributions of agent locations as 2d heatmaps. We see that the ball is much less predictable than the players. We also see that all models capture the player distribution, but the ball is best modeled using GVRNNT-Full. This is quantified in Tab. 4.

5. Conclusion

We address the problem of generative modeling for multi-agent systems, focusing on sports applications. Motivated by the challenges in role-based approaches, we investigate permutation-equivariant (graph-based) stochastic temporal models. Empirically, we demonstrate the effectiveness of each of the proposed components in our GVRNNT model. Additionally, we propose several new evaluation metrics that quantify the quality of the generated samples at a distribution level. These metrics provide additional insights beyond the average prediction error in player locations and pave the way for future research.

Acknowledgments: This work is supported in part by NSF under Grant No. 1718221, Samsung, 3M and a Google PhD Fellowship to RY. We thank NVIDIA for providing GPUs used for this work.

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 *Proc. CVPR*, 2016. 4

[2] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Proc. NeurIPS*, 2016. 4

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 1, 3, 4

[4] S. Becker, R. Hug, W. Hübner, and M. Arens. Red: A simple but effective baseline predictor for the trajnet benchmark. In *Proc. ECCV Workshop*, 2019. 4, 5

[5] A. Bhattacharyya, B. Schiele, and M. Fritz. Accurate and diverse sampling of sequences based on a best of many sample objective. In *Proc. CVPR*, 2018. 1, 6

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 5

[7] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Proc. NeurIPS*, 2015. 1, 2, 4

[8] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 2018. 7

[9] P. Felsen, P. Agrawal, and J. Malik. What will happen next? forecasting player moves in sports videos. In *Proc. ICCV*, 2017. 4

[10] P. Felsen, P. Lucey, and S. Ganguly. Where will they go? predicting fine-grained adversarial multi-agent motion using conditional variational autoencoders. In *Proc. ECCV*, 2018. 1, 3, 4, 7

[11] V. Flunkert, D. Salinas, and J. Gasthaus. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. Apr. 2017. 4

[12] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Proc. NeurIPS*, 2016. 4

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proc. ICML*, 2017. 4

[14] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, 2010. 5

[15] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proc. CVPR*, 2018. 4

[16] Y. Hoshen. Vain: Attentional multi-agent predictive modeling. In *Proc. NeurIPS*, 2017. 4

[17] W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Proc. NeurIPS*, 2017. 4

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 5

[19] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proc. ICLR*, 2014. 2

[20] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *Proc. ICML*, 2018. 4

[21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2017. 4

[22] R. G. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Proc. AAAI*, 2017. 4

[23] H. M. Le, Y. Yue, and P. Carr. Coordinated multi-agent imitation learning. In *Proc. ICML*, 2017. 1, 4, 8

[24] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proc. CVPR*, 2017. 4

[25] N. Lee and K. M. Kitani. Predicting wide receiver trajectories in american football. In *Proc. WACV*, 2016. 4

[26] S. Lee, S. Purushwalkam Shiva Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *Proc. NeurIPS*. 1, 6

[27] P. Lucey, A. Bialkowski, P. Carr, S. Morgan, I. Matthews, and Y. Sheikh. Representing and discovering adversarial team behaviors using player roles. In *Proc. CVPR*, 2013. 3, 4

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. CVPR*, 2017. 4

[29] T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh. Tighter variational bounds are not necessarily better. In *Proc. ICML*, 2018. 5

[30] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proc. ECCV*, 2016. 4

[31] C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, N. Navab, and G. D. Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proc. ICCV*, 2017. 1, 6

[32] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018. 4

[33] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *Trans. on Neural Networks*, 2009. 4

[34] L. Sha, P. Lucey, S. Zheng, T. Kim, Y. Yue, and S. Sridharan. Fine-grained retrieval of sports plays using tree-based alignment of trajectories. *arXiv preprint arXiv:1710.02255*, 2017. 3, 4, 5

[35] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *ICLR*, 2016. 1

[36] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *Proc. ICLR*, 2018. 3, 4

[37] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016. 4

[38] K.-C. Wang and R. Zemel. Classifying nba offensive plays using neural networks. In *Proceedings of MIT Sloan Sports Analytics Conference*, 2016. 4

[39] R. J. Williams and D. Zipser. A learning algorithm for con-

tinually running fully recurrent neural networks. *Neural computation*, 1989. 5

[40] Y. Xu, Z. Piao, and S. Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *Proc. CVPR*, 2018. 4

[41] Y. Yue, P. Lucey, P. Carr, A. Bialkowski, and I. Matthews. Learning fine-grained spatial models for dynamic sports play prediction. In *Proc. ICDM*, 2014. 1, 6

[42] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Proc. NeurIPS*, 2017. 3, 4

[43] E. Zhan, S. Zheng, Y. Yue, and P. Lucey. Generative multi-agent behavioral cloning. *arXiv preprint arXiv:1803.07612*, 2018. 4, 6, 7

[44] S. Zheng, Y. Yue, and J. Hobbs. Generating long-term trajectories using deep hierarchical networks. In *Proc. NeurIPS*, 2016. 1, 4, 7