Dynamic Neural Relational Inference for Forecasting Trajectories

Colin Graber Alexander Schwing
University of Illinois at Urbana-Champaign
{cgraber2, aschwing}@illinois.edu

Abstract

Understanding interactions between entities, e.g.*, joints of the human body, team sports players,* etc.*, is crucial for tasks like forecasting. However, interactions between entities are commonly not observed and often hard to quantify. To address this challenge, recently, 'Neural Relational Inference' was introduced. It predicts static relations between entities in a system and provides an interpretable representation of the underlying system dynamics that are used for better trajectory forecasting. However, generally, relations between entities change as time progresses. Hence, static relations improperly model the data. In response to this, we develop Dynamic Neural Relational Inference (dNRI), which incorporates insights from sequential latent variable models to predict separate relation graphs for every time-step. We demonstrate on several real-world datasets that modeling dynamic relations improves forecasting of complex trajectories.*

1. Introduction

Relations between entities are versatile and appear everywhere, often without us noticing. For instance, joints of the human body are constrained in their movement by a skeleton, team sport players move in practiced formations, and traffic patterns emerge due to enforced rules and respect for our peers. Despite distinct temporal dynamics between entities which emerge in many different situations, it is extremely challenging to explicitly characterize and recover them from observed trajectories. This is in part due to the fact that there are little to no ground truth labels available. For instance, team sport players often have a hard time specifying the causes of their reactions.

Due to this difficulty, in recent years a considerable amount of work has been invested to develop methods which retrieve those interactions. However, many of those methods only recover interactions implicitly, *e.g.*, via graph networks [43, 34, 39, 18, 47, 11, 46] or via attention [32, 3]. Implicitly characterizing and exploiting relations doesn't grant much insight into the underlying system, as these types of approaches lack an explicitly interpretable component. To address this concern, recently, neural relational inference (NRI) has been proposed [22]. NRI is one of the first methods which produces an interpretable representation of the relations between entities in the process of predicting system dynamics. However, importantly, NRI assumes that these relations remain static across an observed trajectory. This is a significant restriction: in many systems, entity relations change over time. Using NRI in those cases will retrieve interactions averaged over time, which doesn't accurately represent the underlying system.

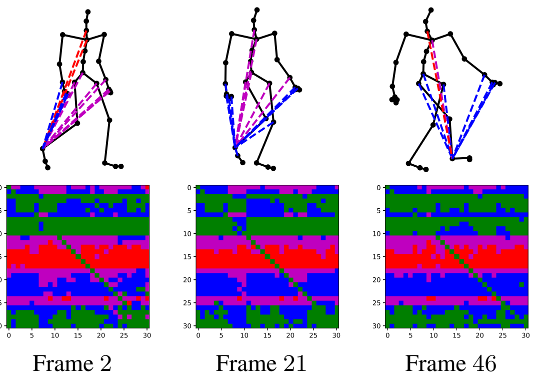


Figure 1: Predicted motion of dNRI on capture subject #35 (top row) and all predicted joint relations (bottom row). The illustrated edges represent those connected to the right heel which change during these three frames.

To address this concern, we develop 'dynamic Neural Relational Inference' (dNRI), a method which recovers interactions between entities at every point in time. More specifically, following NRI, we formulate explicit recovery of the system interactions as a latent variable model: each latent variable denotes the strength of a relation between entities. Using the estimated relational strength, we want to recover the observed trajectory as accurately as possible. However, different from NRI, the developed system estimates latent variables at every point in time (see Fig. 1 for an example visualization of these dynamic relations). Furthermore, we adapt recent advances in sequential latent-variable models to the NRI framework to learn both a sequential relation prior that depends on the history of an input trajectory and an approximate relation posterior which takes both past and future variable states into account.

We assess the proposed dNRI method on challenging motion capture and sports trajectory datasets. We show the developed technique significantly improves recovery of the observed trajectories when compared against static NRI. We additionally demonstrate that the model predicts relations which change across different phases of the trajectories, which static NRI cannot possibly achieve.

2. Background: Neural Relational Inference

To uncover interactions between entities of a system, it is common to study a surrogate task: predicting their trajectories across time. Specifically, given N entities, let $\mathbf{x}_i^{(t)}$ represent the feature vector of entity $i \in \{1, \ldots, N\}$ at time step t, for example position and velocity. The Neural Relational Inference [22] framework models forecasting of trajectories $\mathbf{x}_i = (\mathbf{x}_i^1, \ldots, \mathbf{x}_i^T)$ by first predicting a set of interactions among the entities. Subsequently, these interactions are used to improve prediction of future trajectories. The rationale behind this: accurately recovered interactions permit accurate forecasting.

Formally, interactions between entities take the form of a latent variable $\mathbf{z}_{i,j} \in \{1, \ldots, e\}$ for every pair of entities i and j, where e is the number of relation types being modeled. These relations do not have any pre-defined meaning, but rather the model learns to assign a meaning to each type. In order to predict both the latent relation variables $\mathbf{z}_{i,j}$ and the future trajectories of the entities, NRI learns a variational auto-encoder (VAE) [21, 37]. Its observed variables represent the entity trajectories $\mathbf{x}$ and the latent variables represent the entity relations $\mathbf{z}$. Following a classical VAE, the following evidence lower bound (ELBO) is maximized:

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})], \quad (1)$$

where ϕ and θ are trainable parameters of probability distributions. This formulation consists of three primary probability distributions, which we describe subsequently.

The **encoder** produces a factorized categorical distribution of the form $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{i \neq j} q_\phi(\mathbf{z}_{ij}|\mathbf{x})$ as a function of the entire input sequence $\mathbf{x}$. This is done using a fully-connected graph neural network (GNN) architecture [40, 28, 12] containing one node per entity. This model learns an embedding for each pair of entities which is then used to produce a posterior relation probability for every relation type being predicted.

Given the distribution provided by the encoder, samples of the relation are used as input in the decoder. The sampling procedure needs to be differentiable so that we can update model weights via backpropagation; however, standard sampling from a categorical distribution is non-differentiable. Consequently, we instead take samples from the concrete distribution [33, 20]. This distribution is a continuous approximation to the discrete categorical distribution, and sampling from it proceeds via reparameterization by first sampling a vector $\mathbf{g}$ from the Gumbel$(0, 1)$ distribution and then computing:

$$\mathbf{z}_{i,j} = \mathrm{softmax}((\mathbf{h}_{i,j} + \mathbf{g})/\tau), \quad (2)$$

where $\mathbf{h}_{i,j}$ are the predicted posterior logits for $\mathbf{z}_{i,j}$ and τ is a temperature parameter that controls smoothness of the distribution. This process approximates discrete sampling in a differentiable manner and enables to backpropagate gradients from the decoder reconstruction all the way to the encoder parameters ϕ.

The **decoder** $p_\theta(\mathbf{x}|\mathbf{z})$ uses a sampled set of interactions $\mathbf{z}$ to assist in predicting the future states of the variables $\mathbf{x}$. To this end, it factorizes in an autoregressive manner, *i.e.*, it takes the form

$$p_\theta\left(\mathbf{x}|\mathbf{z}\right) \coloneqq \prod_{t=1}^{T} p_\theta\left(\mathbf{x}^{t+1}|\mathbf{x}^{1:t}, \mathbf{z}\right). \quad (3)$$

Similar to the encoder, the decoder model is based on a GNN. Unlike the encoder, however, a separate GNN is learned for every edge type. When running message passing for a given edge (i, j), the used edge model corresponds to the prediction made by the input latent variable $\mathbf{z}_{i,j}$. One can also hard-code an edge type to represent no interaction, in which case no messages are passed across that edge during computation. Kipf *et al.* [22] experiment with Markovian decoders, in which case the GNN is simply a function of the previous prediction, and decoders that depend on all prior states, in which case a recurrent hidden state is updated using the GNN.

The **prior** $p(\mathbf{z}) \coloneqq \prod_{i \neq j} p(\mathbf{z}_{i,j})$ is a uniform independent categorical distribution per relation variable. If one edge is fixed to represent no interaction, the value of the prior for this edge can be chosen based on the expected sparsity of relations for the given problem. This tunes the loss such that the encoder is biased towards the desired sparsity level.

The training procedure for NRI consists of the following steps: first, the encoder processes the current input $\mathbf{x}$ to predict the posterior relation probability $q_\phi(\mathbf{z}|\mathbf{x})$ for every pair of entities. Next, a set of relations are sampled from the concrete approximation to this distribution. Given these samples $\tilde{\mathbf{z}}$, the final step is to predict the original trajectory $\hat{\mathbf{x}}^2, \ldots, \hat{\mathbf{x}}^T$. To improve decoding performance and ensure that the decoder depends on the predicted edges, Kipf *et al.* [22] provide the decoder at training time with ground-truth inputs for a limited number of steps, *e.g.*, 10, and then predict the remainder of the trajectory as a function of the previous predictions. The ELBO described in Eq. (1) contains two terms: first, the reconstruction error, which assumes the predicted outputs represent means of a Gaussian

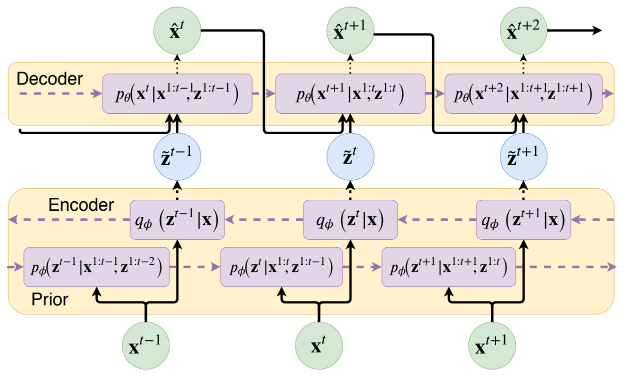

$\hat{\mathbf{x}}^t$
$\hat{\mathbf{x}}^{t+1}$
$\hat{\mathbf{x}}^{t+2}$
Decoder
$p_\theta(\mathbf{x}^t|\mathbf{x}^{1:t-1},\mathbf{z}^{1:t-1})$
$p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^{1:t},\mathbf{z}^{1:t})$
$p_\theta(\mathbf{x}^{t+2}|\mathbf{x}^{1:t+1},\mathbf{z}^{1:t+1})$
$\tilde{\mathbf{z}}^{t-1}$
$\tilde{\mathbf{z}}^{t}$
$\tilde{\mathbf{z}}^{t+1}$
Encoder
$q_\phi(\mathbf{z}^{t-1}|\mathbf{x})$
$q_\phi(\mathbf{z}^{t}|\mathbf{x})$
$q_\phi(\mathbf{z}^{t+1}|\mathbf{x})$
$p_\phi(\mathbf{z}^{t-1}|\mathbf{x}^{1:t-1},\mathbf{z}^{1:t-2})$
$p_\phi(\mathbf{z}^{t}|\mathbf{x}^{1:t},\mathbf{z}^{1:t-1})$
$p_\phi(\mathbf{z}^{t+1}|\mathbf{x}^{1:t+1},\mathbf{z}^{1:t})$
Prior
$\mathbf{x}^{t-1}$
$\mathbf{x}^{t}$
$\mathbf{x}^{t+1}$

Figure 2: The computation graph used by dNRI.

distribution with fixed variance σ and consequently takes the form

$$-\sum_i \sum_{t=2}^{T} \frac{\|\mathbf{x}_i^t - \hat{\mathbf{x}}_i^t\|}{2\sigma^2} + \text{const.} \tag{4}$$

Second, the KL-divergence between the uniform prior and the predicted approximate posterior, which takes the following form:

$$\sum_{i \neq j} H\left(q_\phi\left(\mathbf{z}_{ij}|\mathbf{x}\right)\right) + \text{const.} \tag{5}$$

Here, H represents the entropy function. The constant term is due to the uniform prior, which leads to marginalization of one of the encoder terms in the loss.

The NRI formulation assumes the relations among all of the entities to be static. However, we think this assumption is too strong for many applications – the ways in which entities interact often changes over time. For instance, basketball players adjust their positioning relative to different teammates at different points in time. To address this issue, in the following section, we will describe our 'dynamic Neural Relational Inference' (dNRI).

3. Dynamic Neural Relational Inference

To uncover dynamic interactions and better track entities for which relations change over time, we develop Dynamic Neural Relational Inference (dNRI). Specifically, we predict a *separate* relation $\mathbf{z}_{i,j}^t$ for every time step t. This allows the model to respond to entities whose relations vary throughout a trajectory, thereby improving its ability to anticipate future states. Using our dNRI formulation requires tracking the evolution of the relations between entities across time, which was not needed by static NRI. This requires a novel encoder, decoder, and prior, which are inspired by prior work on sequential latent variable modeling (see Sec. 5 for details). We discuss these components below after providing an overview of our proposed approach.

3.1. Overview

Predicting separate relations at every time step requires rethinking the purpose of each of the model components. As discussed in Sec. 2, the prior is effectively a tunable component of the loss function. In contrast, to make the prior more useful in a sequential context, we now require it to predict the relations between entities at every point in time given all of the previous states of the system.

In static NRI, the encoder predicts a single edge configuration covering an entire set of input trajectories. In contrast, here, we task the encoder with understanding the state of the system at every point in time based on both the past and the future. This "information" is passed from the encoder to the prior during training due to a KL divergence term of the loss function. This change hence encourages the prior model to better anticipate future relations.

As a result of sequential relation prediction, the decoder is now more flexible too: it can use different models at different points in time based on how the system changes. All of these changes lead to a more expressive model that improves prediction performance. An overview of the computation graph which we use for dNRI is provided in Fig. 2. We now describe each of its components in detail.

3.2. Decoder

Our dNRI formulation permits to use any decoder also developed for static NRI. Importantly however, we obtain additional flexibility. The primary difference in this stage is that the relation variable inputs $\mathbf{z}$ now vary per time step. Formally, the decoder model hence factorizes as follows:

$$p_\theta\left(\mathbf{x}|\mathbf{z}\right) := \prod_{t=1}^{T} p_\theta\left(\mathbf{x}^{t+1}|\mathbf{x}^{1:t}, \mathbf{z}^{1:t}\right). \tag{6}$$

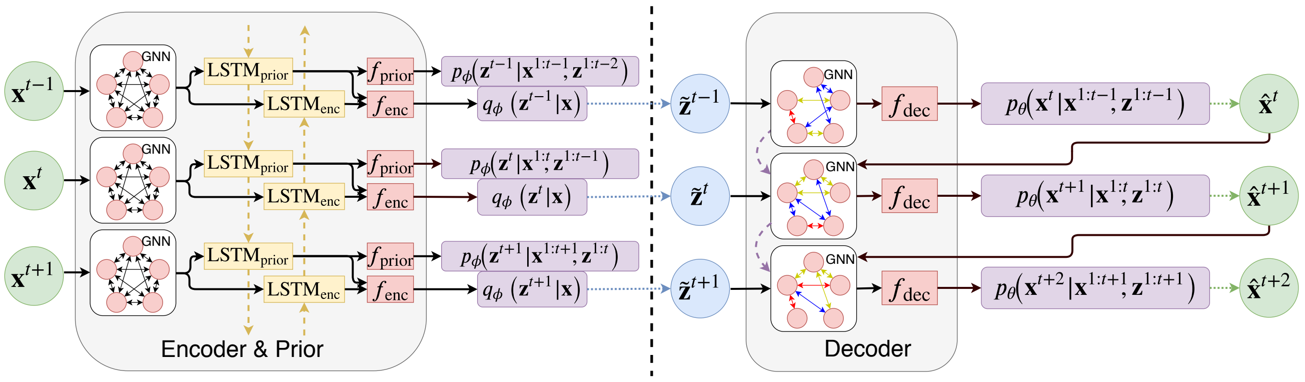
In practice, this amounts to selecting a model for every edge at each time step instead of using the same model throughout the sequence. This allows the decoder to adjust its predictions based on the state of the system, improving its ability to model dynamic systems.

3.3. Prior

Since we expect entity relations to vary at each time step, it is important to capture these changes in the prior distribution. For this, we learn an auto-regressive model of the prior probabilities of the relation variables, where at each time step t the prior is conditioned on previous relations as well as the inputs up to time t. This takes the form:

$$p_\phi\left(\mathbf{z}|\mathbf{x}\right) := \prod_{t=1}^{T} p_\phi\left(\mathbf{z}^t|\mathbf{x}^{1:t}, \mathbf{z}^{1:t-1}\right). \tag{7}$$

The prior architecture which we use is as follows: the input for each time step is passed through the following GNN ar-



$\mathbf{x}^{t-1}$
$\mathbf{x}^{t}$
$\mathbf{x}^{t+1}$
GNN
$\text{LSTM}_{\text{prior}}$
LSTM_{enc}
f_{prior}
f_{enc}
$p_\phi(\mathbf{z}^{t-1}|\mathbf{x}^{1:t-1}, \mathbf{z}^{1:t-2})$
$q_\phi(\mathbf{z}^{t-1}|\mathbf{x})$
$p_\phi(\mathbf{z}^{t}|\mathbf{x}^{1:t}, \mathbf{z}^{1:t-1})$
$q_\phi(\mathbf{z}^{t}|\mathbf{x})$
$p_\phi(\mathbf{z}^{t+1}|\mathbf{x}^{1:t+1}, \mathbf{z}^{1:t})$
$q_\phi(\mathbf{z}^{t+1}|\mathbf{x})$
Encoder & Prior
$\tilde{\mathbf{z}}^{t-1}$
$\tilde{\mathbf{z}}^{t}$
$\tilde{\mathbf{z}}^{t+1}$
f_{dec}
$p_\theta(\mathbf{x}^{t}|\mathbf{x}^{1:t-1}, \mathbf{z}^{1:t-1})$
$p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^{1:t}, \mathbf{z}^{1:t})$
$p_\theta(\mathbf{x}^{t+2}|\mathbf{x}^{1:t+1}, \mathbf{z}^{1:t+1})$
$\hat{\mathbf{x}}^{t}$
$\hat{\mathbf{x}}^{t+1}$
$\hat{\mathbf{x}}^{t+2}$
Decoder

Figure 3: The three model components of dNRI. The inputs are fed through a fully-connected GNN to produce an embedding for every pair of entities at every time step. These are aggregated using a forward LSTM to encode the past history of entity relations and a backwards LSTM to encode the future history of entity relations. The prior is computed as a function of only the past history, while the approximate posterior is computed as a function of both the past and future. A set of edge variables are sampled from the approximate posterior, and these are used to select edge models for the decoder GNN. The decoder evolves a hidden state using this GNN and the previous predictions and predicts the state of the entities at the next time step.

chitecture to produce an embedding per edge per time step:

$$\mathbf{h}_{i,1}^t = f_{\text{emb}}\left(\mathbf{x}_i^t\right) \tag{8}$$

$$v \to e: \quad \mathbf{h}_{(i,j),1}^t = f_e^1\left(\left[\mathbf{h}_{i,1}^t, \mathbf{h}_{j,1}^t\right]\right) \tag{9}$$

$$e \to v: \quad \mathbf{h}_{j,2}^t = f_v^1\left(\sum_{i \neq j} \mathbf{h}_{(i,j),1}^t\right) \tag{10}$$

$$v \to e: \quad \mathbf{h}_{(i,j),\text{emb}}^t = f_e^2\left(\left[\mathbf{h}_{i,2}^t, \mathbf{h}_{j,2}^t\right]\right) \tag{11}$$

This architecture implements a form of neural message passing in a graph, where vertices v represent entities i and edges e represent relations between entity pairs (i, j). Every model f is a multilayer perceptron (MLP), and each $\mathbf{h}$ represents intermediate hidden states over the entities or relations during computation. The output of this computation is the embedding $\mathbf{h}_{(i,j),\text{emb}}^t$, which captures the state of the relations between entities i and j at time t.

Each of these embeddings is fed into an LSTM [17]. Intuitively, this LSTM models the evolution of the relations between entities across time. Finally, another MLP transforms the hidden state at each time step into the logits of the prior distribution. These final two steps are formally specified as follows:

$$\mathbf{h}_{(i,j),\text{prior}}^t = \text{LSTM}_{\text{prior}}\left(\mathbf{h}_{(i,j),\text{emb}}^t, \mathbf{h}_{(i,j),\text{prior}}^{t-1}\right), \tag{12}$$

$$p_\phi(\mathbf{z}^t|\mathbf{x}^{1:t}, \mathbf{z}^{1:t-1}) = \text{softmax}\left(f_{\text{prior}}\left(\mathbf{h}_{(i,j),\text{prior}}^t\right)\right). \tag{13}$$

Fig. 3 provides an illustration of the prior model. Note that, in lieu of passing the previous relation predictions to the prior as input, we encode the dependence of the prior on the relations for previous time steps in the hidden state $\mathbf{h}_{(i,j),\text{prior}}$.

3.4. Encoder

The role of the encoder is to approximate the distribution of relations at each time step as a function of the entire input, as opposed to just the past input history. As described by Krishnan *et al*. [24] and Fraccaro *et al*. [10], the true posterior distribution over the latent variables $p_\theta(\mathbf{z}|\mathbf{x})$ is a function of the *future* states of the observed variables $\mathbf{x}$. Thus, a key component of our encoder is an LSTM that processes the states of the variables in reverse. We re-use the relation embedding $\mathbf{h}_{(i,j),\text{emb}}^t$ described previously and pass these representations through a backward LSTM. The final approximate posterior is then obtained by concatenating this reverse state and the forward state provided by the prior and passing the result into a MLP. The encoder is also illustrated in Fig. 3, and is formally described via:

$$\mathbf{h}_{(i,j),\text{enc}}^t = \text{LSTM}_{\text{enc}}\left(\mathbf{h}_{(i,j),\text{emb}}^t, \mathbf{h}_{(i,j),\text{enc}}^{t+1}\right), \tag{14}$$

$$q_\phi\left(\mathbf{z}_{(i,j)}^t|\mathbf{x}\right) = \text{softmax}\left(f_{\text{enc}}\left(\left[\mathbf{h}_{(i,j),\text{enc}}^t, \mathbf{h}_{(i,j),\text{prior}}^t\right]\right)\right). \tag{15}$$

Note that the encoder and prior models share parameters, so we use ϕ to refer to the parameters of both of these models.

Since the model components of dNRI have changed from static NRI, the training and inference procedures also require modifications. These will be discussed next.

3.5. Training/Inference

To train the parameters ϕ and θ of the encoder/prior and decoder, we proceed as follows: the input trajectories $\mathbf{x}$ are passed through the GNN model to produce relation embeddings $\mathbf{h}_{(i,j),\text{emb}}^t$ for every time t and every entity pair (i, j). These representations are input into the forward/backward LSTMs, and the prior $p_\phi(\mathbf{z}|\mathbf{x})$ and approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ are computed. We then sample from the approximate posterior to get predicted relations $\tilde{\mathbf{z}}$. Given these, we then predict the trajectory distribution $p_\theta(\mathbf{x}|\tilde{\mathbf{z}})$. Unlike in the static NRI case, we always provide ground-truth states to the decoder as input during training, as we observed that providing ground-truth for a fixed number of steps and then using predictions as input for the rest of the trajectory performed worse for dNRI. Finally, we calculate the ELBO: the reconstruction error is computed following Eq. (4), and the KL divergence is computed as

$$\sum_{t=1}^{T}\left(H\left(q_\phi\left(\mathbf{z}_{ij}^t|\mathbf{x}\right)\right)-\sum_{\mathbf{z}_{ij}^t} q_\phi\left(\mathbf{z}_{ij}^t|\mathbf{x}\right)\log p_\phi\left(\mathbf{z}_{ij}^t|\mathbf{x}^{1:t},\mathbf{z}^{1:t-1}\right)\right). \tag{16}$$

At test time, we are tasked with predicting future states of the system. This means that we cannot utilize the encoder to predict edges, as we do not have the proper information about the future. Therefore, given previous predictions $\mathbf{x}^{1:t}$, we compute the prior distribution over relations $p_\phi\left(\mathbf{z}^{1:t}|\mathbf{x}^{1:t},\mathbf{z}^{1:t-1}\right)$. We sample from the prior to obtain relation predictions $\tilde{\mathbf{z}}^t$, and use this as well as our previous predictions to estimate the next state of the variables $p_\theta\left(\mathbf{x}^{t+1}|\mathbf{x}^{1:t},\tilde{\mathbf{z}}^{1:t}\right)$. This process continues until the entire trajectory is predicted.

4. Experiments

To show dNRI's strengths compared to static NRI, we provide experimental results on synthetic particle, human motion capture, basketball player, and traffic trajectory datasets. To show the operation of our models, we additionally visualize sample trajectories and predicted relations.

Unless otherwise specified, we compare the following models and architectures: for the dNRI encoder/prior GNN, f_{emb}, f_e^1, f_v^1, and f_e^2 are all two-layer MLPs with 256 hidden/output units and ELU activations. The LSTM models used by the prior and the encoder use 64 hidden units. Both f_{prior} and f_{enc} are 3-layer MLPs with 128 hidden units and ReLU activations. The static NRI encoder consists of the exact same GNN architecture with the exception that the input into f_{emb} consists of the entire input trajectory. In this case, the encoder logits are produced by passing $\mathbf{h}_{\text{emb}}$ through a 3-layer MLP with 256 hidden units and a number of output units equal to the number of relation types being modeled. This is equivalent to the MLP encoder described by Kipf *et al.* [22], except we add an additional MLP to the output of the GNN. We use the recurrent decoder described by Kipf *et al.* [22] in Eqs. 13-17 and in C.5 for both static and dynamic NRI. Addditionally, every model hard-codes the first edge type to represent no interaction.

For evaluation purposes, models are provided with n initial time steps of input and are tasked with predicting some number of future steps. When evaluating the static model, we use two different inference procedures: the first, labeled as '*Static NRI*', uses the provided initial n time steps of input to predict relation types; these relations are then used for decoding the entire end of the trajectory. The second inference procedure, labeled as '*Static NRI, "Dynamic" Inference*', re-evaluates the relation predictions using the most recent n trajectory predictions.

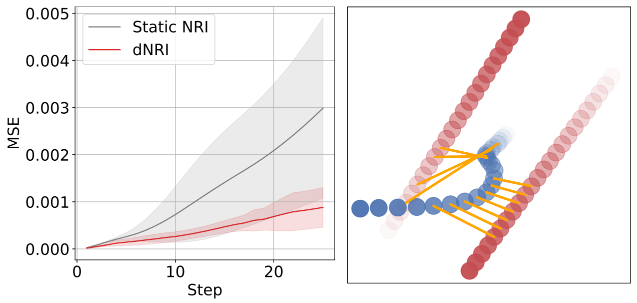


Figure 4: Synthetic data trajectory prediction errors and relation prediction visualization.

In addition to the NRI-based models, we study additional simple baselines: *SingleLSTM* predicts the trajectory for each independently using an LSTM with shared parameters. *JointLSTM* predicts the trajectories for all of the entities jointly using an LSTM, *i.e.*, both the inputs and outputs are the concatenated states of all entities. *FC-Graph* uses the same decoder architecture as dNRI, but assumes a fully-connected graph with one edge type at every time step. Further details and prediction visualizations can be found in the Appendix. Code used to implement these models and run these experiments can be found at `https://github.com/cgraber/cvpr_dNRI`.

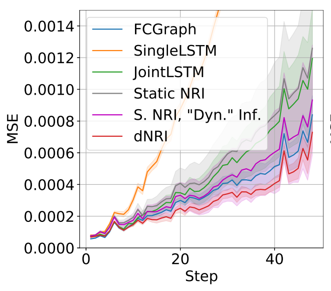
4.1. Synthetic Physics Simulations

The purpose of these experiments is to evaluate the ability of dNRI to recover ground-truth dynamic relations. For this we consider a synthetic dataset constructed to contain dynamic relations. Each trajectory consists of three particles: the first two (red) move with a constant velocity in some direction. The third (blue) is initialized with a random velocity, but is additionally "pushed" away by the other particles whenever the distance separating them is less than 1.

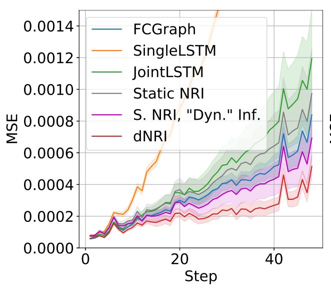
Our findings are summarized in Fig. 4. Static NRI, with average relation prediction F1 of **27.1**, is unable to model the dynamic relations, and performs worse than dNRI, which has average relation prediction F1 of **54.3**.

4.2. Motion Capture Data

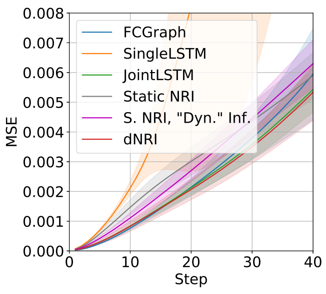
Next we study motion capture recordings from several subjects taken from the CMU motion capture database [8]. We run experiments on two subjects: the first, #35, is the same subject evaluated by Kipf *et al.* [22] and consists of walking trajectories. The second, #118, consists of trials where the subject stands stationary for a differing amount of time and then jumps forward. For the former subject, we follow Kipf *et al.* [22]: train using sequences of length 50



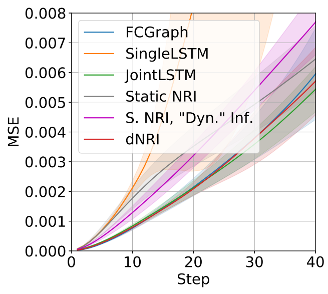
(a) #35, 2 relation types



(b) #35, 4 relation types



(c) #118, 2 relation types



(d) #118, 4 relation types

Figure 5: Trajectory prediction errors on motion capture data. Results are averaged across 5 initializations, with shaded area representing standard deviation.

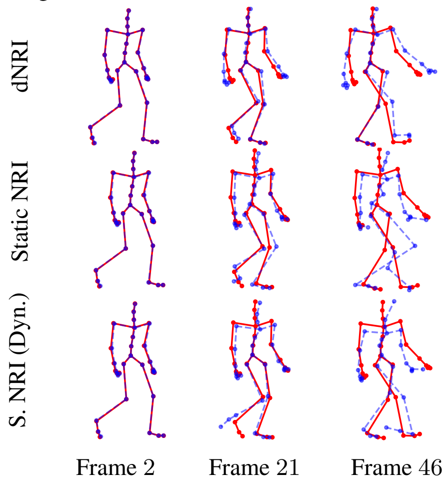


Figure 6: Sample predictions for dNRI (top row), static NRI (middle row), and static NRI with "dynamic" inference (bottom row) on a test trajectory for motion capture subject #35 using 4 relation types. The red, solid skeleton represents the ground-truth state, and the blue, dotted skeleton represents model predictions.

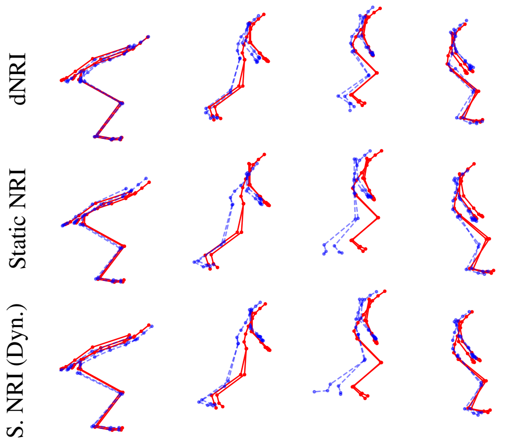
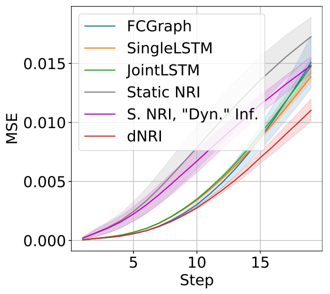


Figure 7: Sample predictions for dNRI (top row), static NRI (middle row), and static NRI with "dynamic" inference (bottom row) on a test trajectory for motion capture subject #118 using 4 relation types. The red, solid skeleton represents the ground-truth state, and the blue, dotted skeleton represents model predictions. Each frame is predicted 20 time steps after the most recent ground-truth was provided.

and evaluate on sequences of length 99 by providing the first 50 frames and predicting the following 49 frames. Due to the lack of regular motion in the trials for subject 118, however, we cannot evaluate in the same way – the initial stationary period varies per trial and provides no information about the jumping motion. Instead, we evaluate as follows: after providing the models with the initial 50 frames of a given trial, we save the current encoder/prior/decoder states and predict the next 40 frames. We then restore the previous states, provide the model with the next step of input, and then predict another 40 frames. This process continues until the end of the trial is reached. We then average the errors for every number of steps, between 1 and 40, since ground-truth states were provided to the model. Separate models are trained to predict two and four relation types.

Fig. 5a and Fig. 5b display the prediction errors for subject #35. The dNRI models are able to predict future trajectories better than both the static NRI models and the simpler baselines. As demonstrated in Fig. 6, the dNRI model is able to predict many frames into the future of the walk cycle without straying too far from the ground-truth skeleton. In contrast, the static NRI model makes significant errors much earlier, reaching a point where significant deformities in the skeleton appear. A visualization of some of the predicted edges for this trajectory are shown in Fig. 1. We observe that, relative to the 'heel' of the skeleton, different relation types are active when picking it up, moving it forward, and placing it back down. This indicates that different models are useful during these three phases of movement.

Fig. 5c and Fig. 5d display the prediction errors for subject #118. Once again, the dNRI models outperform the static NRI models in predicting the future, while performing comparably to the other baselines. However, unlike these baselines, dNRI aids with prediction interpretation, *i.e.*, relation prediction. Fig. 7 shows four predicted time steps for the static and dynamic models, each of which is the 20th step of prediction after the most recent ground-truth states were provided. All models are able to capture the general



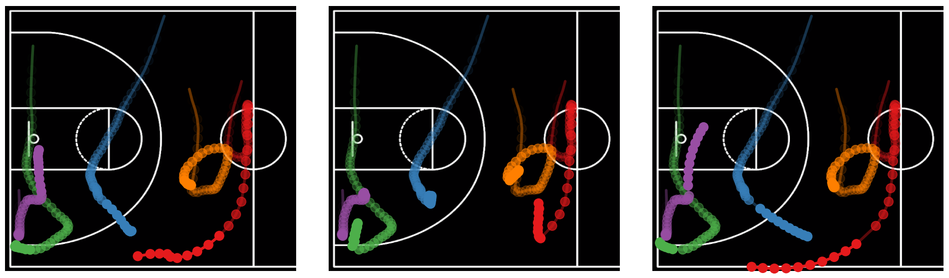


Figure 8: Prediction errors (left) and sample trajectory predictions (right 3) on basketball data. From left to right, these plots represent ground-truth, static NRI, and dNRI (ours). The first 40 frames are provided to the models (transparent), and the models are tasked with predicting the final 9 frames (solid).

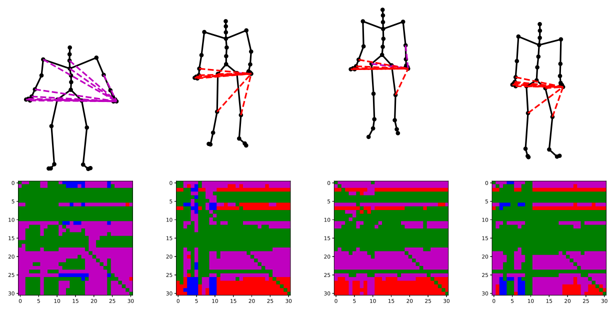


Figure 9: Edge predictions corresponding to dNRI predictions in Fig. 7. The displayed edges represent those connected to the left hand which change during these frames.

jumping motion, but dNRI much more accurately tracks the locations of the leg and hip joints. Fig. 9 visualizes edges used to make these predictions. The relations predicted by the model during the jump preparation phase differ from the relations predicted while the subject is in the middle of jumping. The static model cannot select different relations between different movement phases, and therefore is less flexible than the dynamic model.

4.3. Basketball Data

We next study basketball player trajectory data [53]. Each trajectory contains the 2D positions and velocities of the offensive team, consisting of 5 players. They are preprocessed into 49 frames which span approximately 8 seconds of play. All models are trained on the first 40 frames of the training trajectories; at evaluation time, the models are provided with either the first 30 or 40 frames of input and are tasked with predicting the remaining frames. We train models predicting two relation types.

Fig. 8 displays the prediction errors on the test data for these experiments. On this data, dNRI significantly outperforms the static NRI model in predicting the future trajectory of the players. Fig. 8 also presents a sample player trajectory from the validation dataset, and Fig. 10 displays the predicted edges during the third and 45th time steps.

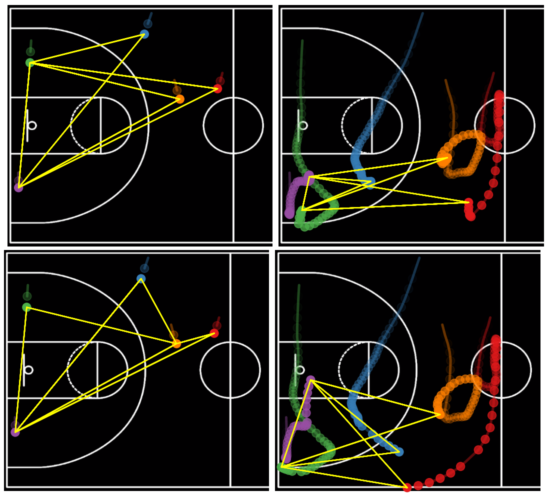


Figure 10: Sample predicted edges for basketball data. The top row represents static NRI, and the bottom row represents dNRI (ours).

The static model mispredicts the general path of the red and blue players, while dNRI is able to capture the correct movement direction. This may be a consequence of the predicted edges: the static model does not predict a relation between the orange player and either the red or the blue player, and therefore the model does not use the path of the orange player to inform their trajectories. In contrast, the dynamic model predicts a relation between these players at the beginning of the trajectory, which informs the initial movement of these entities. In the later frame, the dynamic model no longer predicts these relations, indicating they are not useful to predict the motion of these entities at this time.

4.4. Traffic Trajectory Data

Finally, we study the newly-introduced inD traffic dataset [5]. This dataset consists of recorded vehicle, bicycle, and pedestrian trajectories at traffic intersections. Different from the other studied datasets, the number of entities being tracked varies over time as they enter/leave the area.

Consequently, RNN models or static NRI are not applicable: they assume presence of the same entities at all times. The data contains 36 recordings; we use 19/7/10 for train, validation, and test. For evaluation, we divide each recording into sequences of 50 steps. For each entity that is present in the sequence, we provide the model with ground-truth position and velocity for the first 5 steps it is present, after which the model forecasts the remainder of its trajectory.

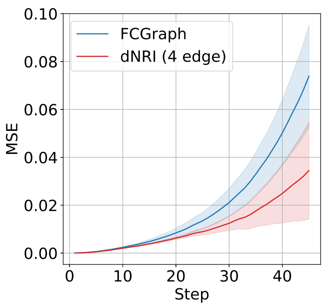


Figure 11: Trajectory prediction errors on inD dataset.

Fig. 11 presents results on this data for dNRI, trained with 4 relations, against an FCGraph baseline. dNRI, which has the ability to model multiple types of interactions between different entities, outperforms FCGraph by a margin.

5. Related Work

Related to our developed 'dynamic Neural Relational Inference' (dNRI) is the recently introduced static NRI [22]. NRI is an unsupervised model which explicitly represents and infers interactions purely from observational data. For this, a variational auto-encoder model [21, 37] is formulated where the latent code represents the underlying interaction graph in the form of an adjacency matrix. Both the encoder and reconstruction models are based on graph neural nets [40, 28, 12]. Different from our dNRI, this static version assumes that the interaction remains identical across time. While this assumption is valid for some systems, it is violated most of the time. We address this concern by developing a model that predicts separate relations at every point in time. Additionally, an independent uniform prior per latent variable is used, whereas we learn a data-dependent sequential prior. Other recent work has attempted to extend NRI in other ways, *e.g.*, by using factorized graphs [49] or including additional structural priors [27]. These extensions are orthogonal to our approach.

Many prior works have attempted to learn the dynamics of various types of systems. These include physical systems, using data from simulated trajectories [4, 16, 6, 35, 31] or generated video data [48, 46], human motion [1, 25, 15, 50, 51, 38], and simulated or real agents [44, 19, 52]. Different from our work, these methods either know/assume the underlying graph structure or infer interactions implicitly. Attention mechanisms [32, 3, 41, 42] can also be viewed as uncovering the interactions of systems, and they have been used previously as a component of graph neural networks [34, 18, 47, 11, 46, 36]. However, different from these works, we explicitly infer interactions over the latent graph structure. There have been attempts to discover relations in other settings as well, including causal reasoning [14] and computational neuroscience [29, 30].

A recent line of work has investigated sequential versions of latent variable models that extend the variational auto-encoder to sequential data. Deep Kalman filters [24], though motivated as an extension to Kalman filters with nonlinear transition/observation functions, learn an autoregressive approximate posterior over the latent state variables within a VAE framework which is a function of both past and future observation states. Other related works are motivated as introducing stochastic variables into recurrent neural net models. These include VRNN [7], which learns a smoothing prior/approximate posterior which is the function of past inputs at every time step, SRNN [10], whose prior/approximate posterior are a function of the entire input at every time step, and Z-Forcing [13], which uses a similar prior/approximate posterior but provides the predicted latent variables as input to the decoder. Aneja *et al*. [2] apply a similar model to the task of image captioning, but they use separate hidden states for the encoder and decoder. We differ from these approaches in several ways: most importantly, our latent variables have an explicit interpretation that represent relations between entities, while theirs do not have a direct interpretable meaning. In addition, we apply our model to predict the future of a provided input trajectory, while their models are used to analyze the structure of text/speech and to generate realistic samples from the training distribution. Similarly, other works which predict trajectories using latent-variable approaches (*e.g.*, [26, 9, 23, 45]) differ in that the learned latent variables represent the state of individual entities or a scene rather than interactions. Several of these works augment the ELBO with additional auxiliary losses to improve performance. A similar loss may be able to improve the performance of dNRI, which we leave to future work.

6. Conclusions

We introduced Dynamic Neural Relational Inference, extending the NRI framework to systems where the relations between entities are expected to change across time. We demonstrated that modeling dynamic entity relations leads to better performance across various tasks. In the future, we will investigate whether we can adapt additional methods used by recent sequential latent variable models, such as auxiliary loss functions, to further improve performance.

Acknowledgements. This work is supported in part by NSF under Grant No. 1718221 and MRI #1725729, UIUC, Samsung, 3M, Cisco Systems Inc. (Gift Award CG 1377144) and Adobe. We thank Raymond Yeh for visualization code, Yurii Vlasov for the helpful discussions, and Cisco for access to the Arcetri cluster.

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. 8

[2] J. Aneja, H. Agrawal, D. Batra, and A. Schwing. Sequential latent spaces for modeling the intention during diverse image captioning. In *Proc. ICCV*, 2019. 8

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015. 1, 8

[4] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Proc. NIPS*, 2016. 8

[5] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *arXiv preprint arXiv:1911.07602*, 2019. 7

[6] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *Proc. ICLR*, 2017. 8

[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. NIPS*, 2015. 8

[8] CMU. Carnegie-mellon motion capture database, 2003. 5

[9] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proc. CVPRW*, 2018. 8

[10] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Proc. NIPS*, 2016. 4, 8

[11] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *Proc. ICLR*, 2018. 1, 8

[12] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proc. ICML*, 2017. 2, 8

[13] A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. Z-forcing: Training stochastic recurrent networks. In *Proc. NIPS*, 2017. 8

[14] C. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 1969. 8

[15] A. Gupta, J. Johnson, F. Li., S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proc. CVPR*, 2018. 8

[16] N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*, 2016. 8

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 4

[18] Y. Hoshen. Vain: Attentional multi-agent predictive modeling. In *Proc. NIPS*, 2017. 1, 8

[19] B. Ivanovic and M. Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proc. ICCV*, 2019. 8

[20] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *Proc. ICLR*, 2017. 2

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014. 2, 8

[22] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *Proc. ICML*, 2018. 1, 2, 5, 8

[23] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Proc. NeurIPS*, 2019. 8

[24] R. G. Krishnan, U. Shalit, and D. Sontag. Deep Kalman Filters. In *https://arxiv.org/abs/1511.05121*, 2015. 4, 8

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

[26] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proc. CVPR*, 2017. 8

[27] Y. Li, C. Meng, C. Shahabi, and Y. Liu. Structure-informed graph auto-encoder for relational inference and simulation. In *ICML Workshop on Learning and Reasoning with Graph-Structured Data*, 2019. 8

[28] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. In *Proc. ICLR*, 2016. 2, 8

[29] S. Linderman and R. Adams. Discovering latent network structure in point process data. In *Proc. ICML*, 2014. 8

[30] S. Linderman, R. Adams, and J. Pillow. Bayesian latent structure discovery from multi-neuron recordings. In *Proc. NIPS*, 2016. 8

[31] I.-J. Liu*, R. Yeh*, and A. G. Schwing. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. In *Proc. CORL*, 2019. * equal contribution. 8

[32] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*, 2015. 1, 8

[33] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: a continuous relaxation of discrete random variables. In *Proc. ICLR*, 2017. 2

[34] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, 2017. 1, 8

[35] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. Fei-Fei, J. Tenenbaum, and Daniel L. Yamins. Flexible neural representation for physics prediction. In *Proc. NeurIPS*, 2018. 8

[36] M. Narasimhan, S. Lazebnik, and A. G. Schwing. Out of the Box: Reasoning with Graph Convolution Nets for Factual Visual Question Answering. In *Proc. NeurIPS*, 2018. 8

[37] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. ICML*, 2014. 2, 8

[38] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. Human motion trajectory prediction: A survey. *arXiv preprint arXiv:1905.06113*, 2019. 8

[39] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Proc. NIPS*, 2017. 1

[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. on NN*, 2008. 2, 8

[41] I. Schwartz, A. G. Schwing, and T. Hazan. High-Order Attention Models for Visual Question Answering. In *Proc. NIPS*, 2017. 8

[42] I. Schwartz, S. Yu, T. Hazan, and A. G. Schwing. Factor Graph Attention. In *Proc. CVPR*, 2019. 8

[43] S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. In *Proc. NIPS*, 2016. 1

[44] C. Sun, P. Karlsson, J. Wu, J. Tenenbaum, and K. Murphy. Stochastic prediction of multi-agent interactions from partial observations. In *Proc. ICLR*, 2019. 8

[45] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *Proc. NeurIPS*, 2019. 8

[46] S. Van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *Proc. ICLR*, 2018. 1, 8

[47] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *Proc. ICLR*, 2018. 1, 8

[48] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Proc. NIPS*, 2017. 8

[49] E. Webb, B. Day, H. Andres-Terre, and P. Lió. Factorised neural relational inference for multi-interaction systems. In *ICML Workshop on Learning and Reasoning with Graph-Structured Data*, 2019. 8

[50] Z. Xu, Z. Liu, C. Sun, K. Murphy, W. T. Freeman, J. B. Tenenbaum, and J. Wu. Unsupervised discovery of parts, structure, and dynamics. In *Proc. ICLR*, 2019. 8

[51] Y. Ye, M. Singh, A. Gupta, and S. Tulsiani. Compositional video prediction. In *Proc. ICCV*, 2019. 8

[52] R. Yeh, A. G. Schwing, J. Huang, and K. Murphy. Diverse Generation for Multi-agent Sports Games. In *Proc. CVPR*, 2019. 8

[53] 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. 7