Neural Relational Inference for Interacting Systems

Thomas Kipf [*1] **Ethan Fetaya** [*2 3] **Kuan-Chieh Wang** [2 3] **Max Welling** [1 4] **Richard Zemel** [2 3 4]

Abstract

Interacting systems are prevalent in nature, from dynamical systems in physics to complex societal dynamics. The interplay of components can give rise to complex behavior, which can often be explained using a simple model of the system's constituent parts. In this work, we introduce the neural relational inference (NRI) model: an unsupervised model that learns to infer interactions while simultaneously learning the dynamics purely from observational data. Our model takes the form of a variational auto-encoder, in which the latent code represents the underlying interaction graph and the reconstruction is based on graph neural networks. In experiments on simulated physical systems, we show that our NRI model can accurately recover ground-truth interactions in an unsupervised manner. We further demonstrate that we can find an interpretable structure and predict complex dynamics in real motion capture and sports tracking data.

1. Introduction

A wide range of dynamical systems in physics, biology, sports, and other areas can be seen as groups of interacting components, giving rise to complex dynamics at the level of individual constituents and in the system as a whole. Modeling these type of dynamics is challenging: often, we only have access to individual trajectories, without knowledge of the underlying interactions or dynamical model.

As a motivating example, let us take the movement of basketball players on the court. It is clear that the dynamics of a single basketball player are influenced by the other players, and observing these dynamics as a human, we are able to reason about the different types of interactions that might arise, e.g. defending a player or setting a screen for a teammate. It might be feasible, though tedious, to manually annotate certain interactions given a task of interest. It is more promising to learn the underlying interactions, perhaps shared across many tasks, in an unsupervised fashion.

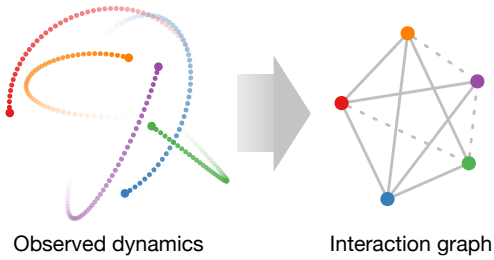


Figure 1. Physical simulation of 2D particles coupled by invisible springs (*left*) according to a latent interaction graph (*right*). In this example, solid lines between two particle nodes denote connections via springs whereas dashed lines denote the absence of a coupling. In general, multiple, directed edge types – each with a different associated relation – are possible.

Recently there has been a considerable amount of work on learning the dynamical model of interacting systems using *implicit* interaction models (Sukhbaatar et al., 2016; Guttenberg et al., 2016; Santoro et al., 2017; Watters et al., 2017; Hoshen, 2017; van Steenkiste et al., 2018). These models can be seen as graph neural networks (GNNs) that send messages over the fully-connected graph, where the interactions are modeled implicitly by the message passing function (Sukhbaatar et al., 2016; Guttenberg et al., 2016; Santoro et al., 2017; Watters et al., 2017) or with the help of an attention mechanism (Hoshen, 2017; van Steenkiste et al., 2018).

In this work, we address the problem of inferring an *explicit* interaction structure while simultaneously learning the dynamical model of the interacting system in an unsupervised way. Our neural relational inference (NRI) model learns the dynamics with a GNN over a discrete *latent* graph, and we perform inference over these latent variables. The inferred edge types correspond to a clustering of the interactions. Using a probabilistic model allows us to incorporate prior beliefs about the graph structure, such as sparsity, in a principled manner.

[*]Equal contribution [1]University of Amsterdam, Amsterdam, The Netherlands [2]University of Toronto, Toronto, Canada [3]Vector Institute, Toronto, Canada [4]Canadian Institute for Advanced Research, Toronto, Canada. Correspondence to: Thomas Kipf <t.n.kipf@uva.nl>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

In a range of experiments on physical simulations, we show that our NRI model possesses a favorable inductive bias that allows it to discover ground-truth physical interactions with high accuracy in a completely unsupervised way. We further show on real motion capture and NBA basketball data that our model can learn a very small number of edge types that enable it to accurately predict the dynamics many time steps into the future.

2. Background: Graph Neural Networks

We start by giving a brief introduction to a recent class of neural networks that operate directly on graph-structured data by passing local messages (Scarselli et al., 2009; Li et al., 2016; Gilmer et al., 2017). We refer to these models as graph neural networks (GNN). Variants of GNNs have been shown to be highly effective at relational reasoning tasks (Santoro et al., 2017), modeling interacting or multi-agent systems (Sukhbaatar et al., 2016; Battaglia et al., 2016), classification of graphs (Bruna et al., 2014; Duvenaud et al., 2015; Dai et al., 2016; Niepert et al., 2016; Defferrard et al., 2016; Kearnes et al., 2016) and classification of nodes in large graphs (Kipf & Welling, 2017; Hamilton et al., 2017). The expressive power of GNNs has also been studied theoretically in (Zaheer et al., 2017; Herzig et al., 2018).

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $v \in \mathcal{V}$ and edges $e = (v, v') \in \mathcal{E}$ [1], we define a single node-to-node message passing operation in a GNN as follows, similar to Gilmer et al. (2017):

$$v \to e: \quad \mathbf{h}_{(i,j)}^{l} = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}]) \tag{1}$$

$$e \to v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j]) \tag{2}$$

where $\mathbf{h}_i^l$ is the embedding of node v_i in layer l, $\mathbf{h}_{(i,j)}^l$ is an embedding of the edge $e_{(i,j)}$, and $\mathbf{x}_i$ and $\mathbf{x}_{(i,j)}$ summarize initial (or auxiliary) node and edge features, respectively (e.g. node input and edge type). $\mathcal{N}_j$ denotes the set of indices of neighbor nodes connected by an incoming edge and $[\cdot, \cdot]$ denotes concatenation of vectors. The functions f_v and f_e are node- and edge-specific neural networks (e.g. small MLPs) respectively (see Figure 2). Eqs. (1)–(2) allow for the composition of models that map from edge to node representations or vice-versa via multiple rounds of message passing.

In the original GNN formulation from Scarselli et al. (2009) the node embedding $\mathbf{h}_{(i,j)}^l$ depends only on $\mathbf{h}_i^l$, the embedding of the sending node, and the edge type, but not on $\mathbf{h}_j^l$, the embedding of the receiving node. This is of course a special case of this formulation, and more recent works such as interaction networks (Battaglia et al., 2016) or message passing neural networks (Gilmer et al., 2017) are in line with our more general formulation. We further note that some recent works factor $f_e^l(\cdot)$ into a product of two separate functions, one of which acts as a gating or attention mechanism (Monti et al., 2017; Duan et al., 2017; Hoshen, 2017; Veličković et al., 2018; Garcia & Bruna, 2018; van Steenkiste et al., 2018) which in some cases can have computational benefits or introduce favorable inductive biases.

[1] Undirected graphs can be modeled by explicitly assigning two directed edges in opposite direction for each undirected edge.

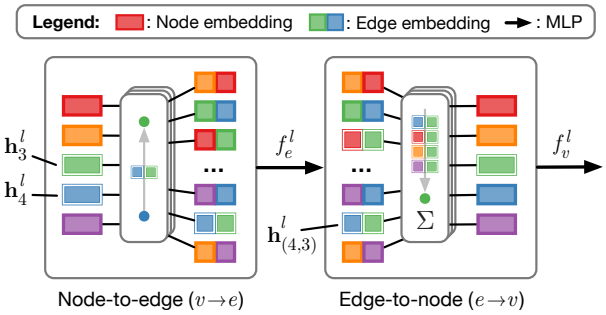


Figure 2. Node-to-edge ($v \to e$) and edge-to-node ($e \to v$) operations for moving between node and edge representations in a GNN. $v \to e$ represents concatenation of node embeddings connected by an edge, whereas $e \to v$ denotes the aggregation of edge embeddings from all incoming edges. In our notation in Eqs. (1)–(2), every such operation is followed by a small neural network (e.g. a 2-layer MLP), here denoted by a black arrow. For clarity, we highlight which node embeddings are combined to form a specific edge embedding ($v \to e$) and which edge embeddings are aggregated to a specific node embedding ($e \to v$).

3. Neural Relational Inference Model

Our NRI model consists of two parts trained jointly: An encoder that predicts the interactions given the trajectories, and a decoder that learns the dynamical model given the interaction graph.

More formally, our input consists of trajectories of N objects. We denote by $\mathbf{x}_i^t$ the feature vector of object v_i at time t, e.g. location and velocity. We denote by $\mathbf{x}^t = \{\mathbf{x}_1^t, ..., \mathbf{x}_N^t\}$ the set of features of all N objects at time t, and we denote by $\mathbf{x}_i = (\mathbf{x}_i^1, ..., \mathbf{x}_i^T)$ the trajectory of object i, where T is the total number of time steps. Lastly, we mark the whole trajectories by $\mathbf{x} = (\mathbf{x}^1, ..., \mathbf{x}^T)$. We assume that the dynamics can be modeled by a GNN given an unknown graph $\mathbf{z}$ where $\mathbf{z}_{ij}$ represents the discrete edge type between objects v_i and v_j. The task is to simultaneously learn to predict the edge types and learn the dynamical model in an unsupervised way.

We formalize our model as a variational autoencoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) that maximizes the ELBO:

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})] \tag{3}$$

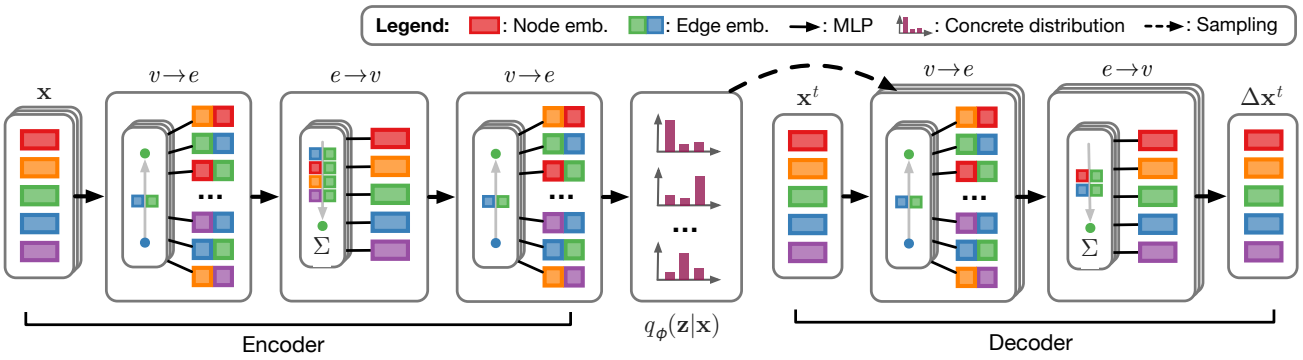


Figure 3. The NRI model consists of two jointly trained parts: An encoder that predicts a probability distribution $q_\phi(\mathbf{z}|\mathbf{x})$ over the latent interactions given input trajectories; and a decoder that generates trajectory predictions conditioned on both the latent code of the encoder and the previous time step of the trajectory. The encoder takes the form of a GNN with multiple rounds of node-to-edge ($v \to e$) and edge-to-node ($e \to v$) message passing, whereas the decoder runs multiple GNNs in parallel, one for each edge type supplied by the latent code of the encoder $q_\phi(\mathbf{z}|\mathbf{x})$.

The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ returns a factorized distribution of $\mathbf{z}_{ij}$, where $\mathbf{z}_{ij}$ is a discrete categorical variable representing the edge type between object v_i and v_j. We use a one-hot representation of the K interaction types for $\mathbf{z}_{ij}$.

The decoder

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z}) \tag{4}$$

models $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z})$ with a GNN given the latent graph structure $\mathbf{z}$.

The prior $p_\theta(\mathbf{z}) = \prod_{i \neq j} p_\theta(\mathbf{z}_{ij})$ is a factorized uniform distribution over edges types. If one edge type is "hard coded" to represent "non-edge" (no messages being passed along this edge type), we can use an alternative prior with higher probability on the "non-edge" label. This will encourage sparser graphs.

There are some notable differences between our model and the original formulation of the VAE (Kingma & Welling, 2014). First, in order to avoid the common issue in VAEs of the decoder ignoring the latent code $\mathbf{z}$ (Chen et al., 2017), we train the decoder to predict multiple time steps and not a single step as the VAE formulation requires. This is necessary since interactions often only have a small effect in the time scale of a single time step. Second, the latent distribution is discrete, so we use a continuous relaxation in order to use the reparameterization trick. Lastly, we note that we do not learn the probability $p(\mathbf{x}^1)$ (i.e. for $t = 1$) as we are interested in the dynamics and interactions, and this does not have any effect on either (but would be easy to include if there was a need).

The overall model is schematically depicted in Figure 3. In the following, we describe the encoder and decoder components of the model in detail.

3.1. Encoder

At a high level, the goal of the encoder is to infer pairwise interaction types $\mathbf{z}_{ij}$ given observed trajectories $\mathbf{x} = (\mathbf{x}^1, ..., \mathbf{x}^T)$. Since we do not know the underlying graph, we can use a GNN on the fully-connected graph to predict the latent graph structure.

More formally, we model the encoder as $q_\phi(\mathbf{z}_{ij}|\mathbf{x}) = \mathrm{softmax}(f_{\mathrm{enc},\phi}(\mathbf{x})_{ij,1:K})$, where $f_{\mathrm{enc},\phi}(\mathbf{x})$ is a GNN acting on the fully-connected graph (without self-loops). Given input trajectories $\mathbf{x}_1, ..., \mathbf{x}_N$ our encoder computes the following message passing operations:

$$\mathbf{h}_j^1 = f_{\mathrm{emb}}(\mathbf{x}_j) \tag{5}$$

$$v \to e: \quad \mathbf{h}_{(i,j)}^1 = f_e^1([\mathbf{h}_i^1, \mathbf{h}_j^1]) \tag{6}$$

$$e \to v: \quad \mathbf{h}_j^2 = f_v^1\Big(\sum_{i \neq j} \mathbf{h}_{(i,j)}^1\Big) \tag{7}$$

$$v \to e: \quad \mathbf{h}_{(i,j)}^2 = f_e^2([\mathbf{h}_i^2, \mathbf{h}_j^2]) \tag{8}$$

Finally, we model the edge type posterior as $q_\phi(\mathbf{z}_{ij}|\mathbf{x}) = \mathrm{softmax}(\mathbf{h}_{(i,j)}^2)$ where ϕ summarizes the parameters of the neural networks in Eqs. (5)–(8). The use of multiple passes, two in the model presented here, allows the model to "disentangle" multiple interactions while still using only binary terms. In a single pass, Eqs. (5)–(6), the embedding $\mathbf{h}_{(i,j)}^1$ only depends on $\mathbf{x}_i$ and $\mathbf{x}_j$ ignoring interactions with other nodes, while $\mathbf{h}_j^2$ uses information from the whole graph.

The functions $f_{(...)}$ are neural networks that map between the respective representations. In our experiments we used either fully-connected networks (MLPs) or 1D convolutional networks (CNNs) with attentive pooling similar to (Lin et al., 2017) for the $f_{(...)}$ functions. See supplementary material for further details.

While this model falls into the general framework presented in Sec. 3, there is a conceptual difference in how $\mathbf{h}_{(i,j)}^l$

are interpreted. Unlike in a typical GNN, the messages $\mathbf{h}^l_{(i,j)}$ are no longer considered just a transient part of the computation, but an integral part of the model that represents the edge embedding used to perform edge classification.

3.2. Sampling

It is straightforward to sample from $q_\phi(\mathbf{z}_{ij}|\mathbf{x})$, however we cannot use the reparametrization trick to backpropagate though the sampling as our latent variables are discrete. A recently popular approach to handle this difficulty is to sample from a continuous approximation of the discrete distribution (Maddison et al., 2017; Jang et al., 2017) and use the repramatrization trick to get (biased) gradients from this approximation. We used the concrete distribution (Maddison et al., 2017) where samples are drawn as:

$$\mathbf{z}_{ij} = \mathrm{softmax}((\mathbf{h}^2_{(i,j)} + \mathbf{g})/\tau) \tag{9}$$

where $\mathbf{g} \in \mathbb{R}^K$ is a vector of i.i.d. samples drawn from a $\mathrm{Gumbel}(0, 1)$ distribution and τ (softmax temperature) is a parameter that controls the "smoothness" of the samples. This distribution converges to one-hot samples from our categorical distribution when $\tau \to 0$.

3.3. Decoder

The task of the decoder is to predict the future continuation of the interacting system's dynamics $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z})$. Since the decoder is conditioned on the graph $\mathbf{z}$ we can in general use any GNN algorithm as our decoder.

For physics simulations the dynamics is Markovian $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z}) = p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, \mathbf{z})$, if the state is location and velocity and $\mathbf{z}$ is the ground-truth graph. For this reason we use a GNN similar to interaction networks; unlike interaction networks we have a separate neural network for each edge type. More formally:

$$v \to e: \quad \tilde{\mathbf{h}}^t_{(i,j)} = \sum_k z_{ij,k} \tilde{f}^k_e([\mathbf{x}^t_i, \mathbf{x}^t_j]) \tag{10}$$

$$e \to v: \quad \boldsymbol{\mu}^{t+1}_j = \mathbf{x}^t_j + \tilde{f}_v(\textstyle\sum_{i \neq j} \tilde{\mathbf{h}}^t_{(i,j)}) \tag{11}$$

$$p(\mathbf{x}^{t+1}_j|\mathbf{x}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}^{t+1}_j, \sigma^2\mathbf{I}) \tag{12}$$

Note that $z_{ij,k}$ denotes the k-th element of the vector $\mathbf{z}_{ij}$ and σ^2 is a fixed variance. When $z_{ij,k}$ is a discrete one-hot sample the messages $\tilde{\mathbf{h}}^t_{(i,j)}$ are $\tilde{f}^k_e([\mathbf{x}^t_i, \mathbf{x}^t_j])$ for the selected edge type k, and for the continuous relaxation we get a weighted sum. Also note that since in Eq. 11 we add the present state $\mathbf{x}^t_j$ our model only learns the change in state $\Delta\mathbf{x}^t_j$.

3.4. Avoiding degenerate decoders

If we look at the ELBO, Eq. 3, the reconstruction loss term has the form $\sum_{t=1}^T \log[p(\mathbf{x}^t|\mathbf{x}^{t-1}, \mathbf{z})]$ which involves only single step predictions. One issue with optimizing this objective is that the interactions can have a small effect on short-term dynamics. For example, in physics simulations a fixed velocity assumption can be a good approximation for a short time period. This leads to a sub-optimal decoder that ignores the latent edges completely and achieves only a marginally worse reconstruction loss.

We address this issue in two ways: First, we predict multiple steps into the future, where a "degenerate" decoder (which ignores the latent edges) would perform much worse. Second, instead of having one neural network that computes the messages given $[\mathbf{x}^t_i, \mathbf{x}^t_j, \mathbf{z}_{ij}]$, as was done in (Battaglia et al., 2016), we have a separate MLP for each edge type. This makes the dependence on the edge type more explicit and harder to be ignored by the model.

Predicting multiple steps is implemented by replacing the correct input $\mathbf{x}^t$, with the predicted mean $\boldsymbol{\mu}^t$ for M steps (we used $M = 10$ in our experiments), then feed in the correct previous step and reiterate. More formally, if we denote our decoder as $\boldsymbol{\mu}^{t+1}_j = f_{\mathrm{dec}}(\mathbf{x}^t_j)$ then we have:

$$\begin{aligned}
\boldsymbol{\mu}^2_j &= f_{\mathrm{dec}}(\mathbf{x}^1_j) \\
\boldsymbol{\mu}^{t+1}_j &= f_{\mathrm{dec}}(\boldsymbol{\mu}^t_j) && t = 2, \ldots, M \\
\boldsymbol{\mu}^{M+2}_j &= f_{\mathrm{dec}}(\mathbf{x}^{M+1}_j) \\
\boldsymbol{\mu}^{t+1}_j &= f_{\mathrm{dec}}(\boldsymbol{\mu}^t_j) && t = M+2, \ldots, 2M \\
&\ldots
\end{aligned}$$

We are backpropagating through this whole process, and since the errors accumulate for M steps the degenerate decoder is now highly suboptimal.

3.5. Recurrent decoder

In many applications the Markovian assumption used in Sec. 3.3 does not hold. To handle such applications we use a recurrent decoder that can model $p_\theta(\mathbf{x}^{t+1}|\mathbf{x}^t, ..., \mathbf{x}^1, \mathbf{z})$. Our recurrent decoder adds a GRU (Cho et al., 2014) unit to the GNN message passing operation. More formally:

$$v \to e: \quad \tilde{\mathbf{h}}^t_{(i,j)} = \sum_k z_{ij,k} \tilde{f}^k_e([\tilde{\mathbf{h}}^t_i, \tilde{\mathbf{h}}^t_j]) \tag{13}$$

$$e \to v: \quad \mathrm{MSG}^t_j = \textstyle\sum_{i \neq j} \tilde{\mathbf{h}}^t_{(i,j)} \tag{14}$$

$$\tilde{\mathbf{h}}^{t+1}_j = \mathrm{GRU}([\mathrm{MSG}^t_j, \mathbf{x}^t_j], \tilde{\mathbf{h}}^t_j) \tag{15}$$

$$\boldsymbol{\mu}^{t+1}_j = \mathbf{x}^t_j + f_{\mathrm{out}}(\tilde{\mathbf{h}}^{t+1}_j) \tag{16}$$

$$p(\mathbf{x}^{t+1}|\mathbf{x}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}^{t+1}, \sigma^2\mathbf{I}) \tag{17}$$

The input to the message passing operation is the recurrent hidden state at the previous time step. f_{out} denotes an output transformation, modeled by a small MLP. For each node v_j the input to the GRU update is the concatenation of the aggregated messages MSG^{t+1}_j, the current input $\mathbf{x}^{t+1}_j$, and the previous hidden state $\tilde{\mathbf{h}}^t_j$.

If we wish to predict multiple time steps in the recurrent setting, the method suggested in Sec. 3.4 will be problematic. Feeding in the predicted (potentially incorrect) path and then periodically jumping back to the true path will generate artifacts in the learned trajectories. In order to avoid this issue we provide the correct input $\mathbf{x}_j^t$ in the first $(T-M)$ steps, and only utilize our predicted mean $\boldsymbol{\mu}_j^t$ as input at the last M time steps.

3.6. Training

Now that we have described all the elements, the training goes as follows: Given training example $\mathbf{x}$ we first run the encoder and compute $q_\phi(\mathbf{z}_{ij}|\mathbf{x})$, then we sample $\mathbf{z}_{ij}$ from the concrete reparameterizable approximation of $q_\phi(\mathbf{z}_{ij}|\mathbf{x})$. We then run the decoder to compute $\boldsymbol{\mu}^2, ..., \boldsymbol{\mu}^T$. The ELBO objective, Eq. 3, has two terms: the reconstruction error $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ and KL divergence $\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})]$. The reconstruction error is estimated by:

$$-\sum_j \sum_{t=2}^{T} \frac{||\mathbf{x}_j^t - \boldsymbol{\mu}_j^t||^2}{2\sigma^2} + \text{const} \tag{18}$$

while the KL term for a uniform prior is just the sum of entropies (plus a constant):

$$\sum_{i \neq j} H(q_\phi(\mathbf{z}_{ij}|\mathbf{x})) + \text{const}. \tag{19}$$

As we use a reparameterizable approximation, we can compute gradients by backpropagation and optimize.

4. Related Work

Several recent works have studied the problem of learning the dynamics of a physical system from simulated trajectories (Battaglia et al., 2016; Guttenberg et al., 2016; Chang et al., 2017) and from generated video data (Watters et al., 2017; van Steenkiste et al., 2018) with a graph neural network. Unlike our work they either assume a known graph structure or infer interactions implicitly.

Recent related works on graph-based methods for human motion prediction include (Alahi et al., 2016) where the graph is not learned but is based on proximity and (Le et al., 2017) tries to cluster agents into roles.

A number of recent works (Monti et al., 2017; Duan et al., 2017; Hoshen, 2017; Veličković et al., 2018; Garcia & Bruna, 2018; van Steenkiste et al., 2018) parameterize messages in GNNs with a soft attention mechanism (Luong et al., 2015; Bahdanau et al., 2015). This equips these models with the ability to focus on specific interactions with neighbors when aggregating messages. Our work is different from this line of research, as we explicitly perform inference over the latent graph structure. This allows for the incorporation of prior beliefs (such as sparsity) and for an interpretable discrete structure with multiple relation types.

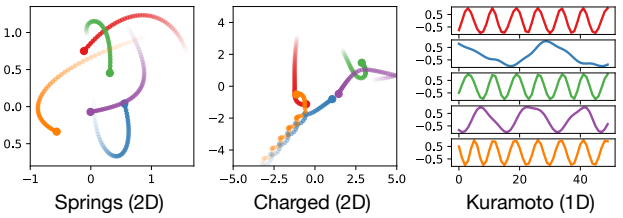


Figure 4. Examples of trajectories used in our experiments from simulations of particles connected by springs (left), charged particles (middle), and phase-coupled oscillators (right).

The problem of inferring interactions or latent graph structure has been investigated in other settings in different fields. For example, in causal reasoning Granger causality (Granger, 1969) infers causal relations. Another example from computational neuroscience is (Linderman et al., 2016; Linderman & Adams, 2014) where they infer interactions between neural spike trains.

5. Experiments

Our encoder implementation uses fully-connected networks (MLPs) or 1D CNNs with attentive pooling as our message passing function. For our decoder we used fully-connected networks or alternatively a recurrent decoder. Optimization was performed using the Adam algorithm (Kingma & Ba, 2015). We provide full implementation details in the supplementary material. Our implementation uses PyTorch (Paszke et al., 2017) and is available online[2].

5.1. Physics simulations

We experimented with three simulated systems: particles connected by springs, charged particles and phase-coupled oscillators (Kuramoto model) (Kuramoto, 1975). These settings allow us to attempt to learn the dynamics and interactions when the interactions are known. These systems, controlled by simple rules, can exhibit complex dynamics. For the springs and Kuramoto experiments the objects do or do not interact with equal probability. For the charged particles experiment they attract or repel with equal probability. Example trajectories can be seen in Fig. 4. We generate 50k training examples, and 10k validation and test examples for all tasks. Further details on the data generation and implementation are in the supplementary material.

We note that the simulations are differentiable and so we can use it as a ground-truth decoder to train the encoder. The charged particles simulation, however, suffers from instabil-

[2] https://github.com/ethanfetaya/nri

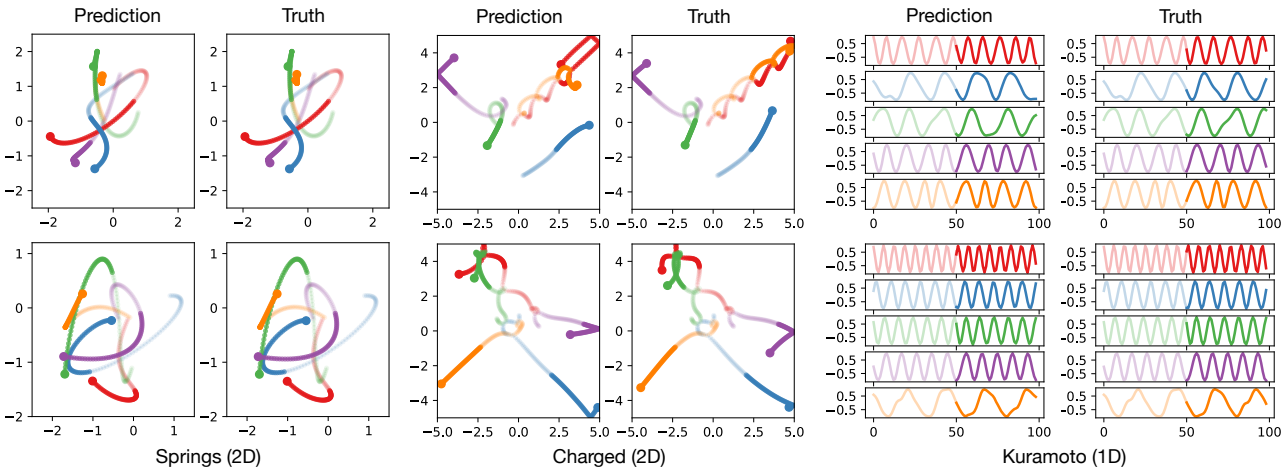


Figure 5. Trajectory predictions from a trained NRI model (unsupervised). Semi-transparent paths denote the first 49 time steps of ground-truth input to the model, from which the interaction graph is estimated. Solid paths denote self-conditioned model predictions.

Table 1. Accuracy (in %) of unsupervised interaction recovery.

Model	**Springs**	**Charged**	**Kuramoto**
	5 objects		
Corr. (path)	52.4±0.0	55.8±0.0	62.8±0.0
Corr. (LSTM)	52.7±0.9	54.2±2.0	54.4±0.5
NRI (sim.)	**99.8**±0.0	59.6±0.8	–
NRI (learned)	**99.9**±0.0	**82.1**±0.6	**96.0**±0.1
Supervised	99.9±0.0	95.0±0.3	99.7±0.0
	10 objects		
Corr. (path)	50.4±0.0	51.4±0.0	59.3±0.0
Corr. (LSTM)	54.9±1.0	52.7±0.2	56.2±0.7
NRI (sim.)	**98.2**±0.0	53.7±0.8	–
NRI (learned)	**98.4**±0.0	**70.8**±0.4	**75.7**±0.3
Supervised	98.8±0.0	94.6±0.2	97.1±0.1

ity which led to some performance issues when calculating gradients; see supplementary material for further details. We used an external code base (Laszuk, 2017) for stable integration of the Kuramoto ODE and therefore do not have access to gradient information in this particular simulation.

Results We ran our NRI model on all three simulated physical systems and compared our performance, both in future state prediction and in accuracy of estimating the edge type in an unsupervised manner.

For edge prediction, we compare to the "gold standard" i.e. training our encoder in a supervised way given the ground-truth labels. We also compare to the following baselines: Our NRI model with the ground-truth simulation decoder, NRI (sim.), and two correlation based baselines, Corr. (path) and Corr. (LSTM). Corr. (path) estimates the interaction graph by thresholding the matrix of correlations between trajectory feature vectors. Corr. (LSTM) trains an LSTM (Hochreiter & Schmidhuber, 1997) with shared parameters to model each trajectory individually and calculates correlations between the final hidden states to arrive at an interaction matrix after thresholding. We provide further details on these baselines in the supplementary material.

Results for the unsupervised interaction recovery task are summarized in Table 1 (average over 5 runs and standard error). As can be seen, the unsupervised NRI model, NRI (learned), greatly surpasses the baselines and recovers the ground-truth interaction graph with high accuracy on most tasks. For the springs model our unsupervised method is comparable to the supervised "gold standard" benchmark. We note that our supervised baseline is similar to the work by (Santoro et al., 2017), with the difference that we perform multiple rounds of message passing in the graph. Additional results on experiments with more than two edge types and non-interacting particles are described in the supplementary material.

For future state prediction we compare to the static baseline, i.e. $\mathbf{x}^{t+1} = \mathbf{x}^t$, two LSTM baselines, and a full graph baseline. One LSTM baseline, marked as "single", runs a separate LSTM (with shared weights) for each object. The second, marked as "joint" concatenates all state vectors and feeds it into one LSTM that is trained to predict all future states simultaneously. Note that the latter will only be able to operate on a fixed number of objects (in contrast to the other models).

In the full graph baseline, we use our message passing decoder on the fully-connected graph without edge types, i.e. without inferring edges. This is similar to the model

Table 2. Mean squared error (MSE) in predicting future states for simulations with 5 interacting objects.

	Springs			Charged			Kuramoto		
Prediction steps	1	10	20	1	10	20	1	10	20
Static	7.93e-5	7.59e-3	2.82e-2	5.09e-3	2.26e-2	5.42e-2	5.75e-2	3.79e-1	3.39e-1
LSTM (single)	2.27e-6	4.69e-4	4.90e-3	2.71e-3	7.05e-3	1.65e-2	7.81e-4	3.80e-2	8.08e-2
LSTM (joint)	4.13e-8	2.19e-5	7.02e-4	1.68e-3	6.45e-3	1.49e-2	**3.44e-4**	**1.29e-2**	4.74e-2
NRI (full graph)	1.66e-5	1.64e-3	6.31e-3	**1.09e-3**	3.78e-3	9.24e-3	2.15e-2	5.19e-2	8.96e-2
NRI (learned)	**3.12e-8**	**3.29e-6**	**2.13e-5**	**1.05e-3**	**3.21e-3**	**7.06e-3**	1.40e-2	2.01e-2	**3.26e-2**
NRI (true graph)	1.69e-11	1.32e-9	7.06e-6	1.04e-3	3.03e-3	5.71e-3	1.35e-2	1.54e-2	2.19e-2

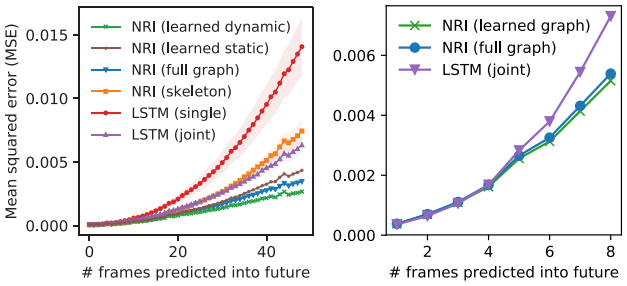


Figure 6. Test MSE comparison for motion capture (walking) data (*left*) and sports tracking (SportVU) data (*right*).

used in (Watters et al., 2017). We also compare to the "gold standard" model, denoted as NRI (true graph), which is training only a decoder using the ground-truth graph as input. The latter baseline is comparable to previous works such as interaction networks (Battaglia et al., 2016).

In order to have a fair comparison, we generate longer test trajectories and only evaluate on the last part unseen by the encoder. Specifically, we run the encoder on the first 49 time steps (same as in training and validation), then predict with our decoder the following 20 unseen time steps. For the LSTM baselines, we first have a "burn-in" phase where we feed the LSTM the first 49 time steps, and then predict the next 20 time steps. This way both algorithms have access to the first 49 steps when predicting the next 20 steps. We show mean squared error (MSE) results in Table 2, and note that our results are better than using LSTM for long term prediction. Example trajectories predicted by our NRI (learned) model for up to 50 time steps are shown in Fig. 5.

For the Kuramoto model, we observe that the LSTM baselines excel at smoothly continuing the shape of the waveform for short time frames, but fail to model the long-term dynamics of the interacting system. We provide further qualitative analysis for these results in the supplementary material.

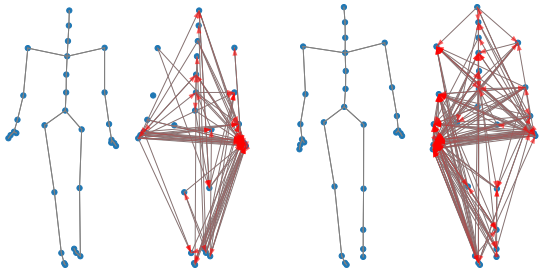
It is interesting to note that the charged particles experiment achieves an MSE score which is on par with the NRI model given the true graph, while only predicting 82.6% of the edges accurately. This is explained by the fact that far away particles have weak interactions, which have only small effects on future prediction. An example can be seen in Fig. 5 in the top row where the blue particle is repelled instead of being attracted.

5.2. Motion capture data

The CMU Motion Capture Database (CMU, 2003) is a large collection of motion capture recordings for various tasks (such as walking, running, and dancing) performed by human subjects. We here focus on recorded walking motion data of a single subject (subject #35). The data is in the form of 31 3D trajectories, each tracking a single joint. We split the different walking trials into non-overlapping training (11 trials), validation (4 trials) and test sets (7 trials). We provide both position and velocity data. See supplementary material for further details. We train our NRI model with an MLP encoder and RNN decoder on this data using 2 or 4 edge types where one edge type is "hard-coded" as non-edge, i.e. messages are only passed on the other edge types. We found that experiments with 2 and 4 edge types give almost identical results, with two edge types being comparable in capacity to the fully connected graph baseline while four edge types (with sparsity prior) are more interpretable and allow for easier visualization.

Dynamic graph re-evaluation We find that the learned graph depends on the particular phase of the motion (Fig. 7), which indicates that the ideal underlying graph is dynamic. To account for this, we dynamically re-evaluate the NRI encoder for every time step during testing, effectively resulting in a dynamically changing latent graph that the decoder can utilize for more accurate predictions.

Results The qualitative results for our method and the same baselines used in Sec. 5.1 can be seen in Fig. 6. As one can see, we outperform the fully-connected graph setting in long-term predictions, and both models outperform the LSTM baselines. Dynamic graph re-evaluation significantly



(a) Right hand focus (b) Left hand focus

Figure 7. Learned latent graphs on motion capture data (4 edge types)[4]. Skeleton shown for reference. Red arrowheads denote directionality of a learned edge. The edge type shown favors a specific hand depending on the state of the movement and gathers information mostly from other extremities.

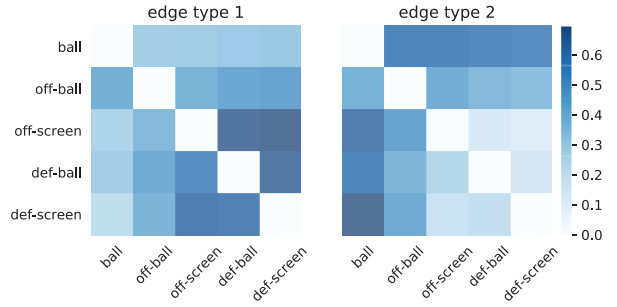


Figure 8. Distribution of learned edges between players (and the ball) in the basketball sports tracking (SportVU) data.

improves predictive performance for this dataset compared to a static baseline. One interesting observation is that the skeleton graph is quite suboptimal, which is surprising as the skeleton is the "natural" graph. When examining the edges found by our model (trained with 4 edge types and a sparsity prior) we see an edge type that mostly connects a hand to other extremities, especially the opposite hand, as seen in Fig. 7. This can seem counter-intuitive as one might assume that the important connections are local, however we note that some leading approaches for modeling motion capture data (Jain et al., 2016) do indeed include hand to hand interactions.

5.3. Pick and Roll NBA data

The National Basketball Association (NBA) uses the SportVU tracking system to collect player tracking data, where each frame contains the location of all ten players and the ball. Similar to our previous experiments, we test our model on the task of future trajectory prediction. Since the interactions between players are dynamic, and our current formulation assumes fixed interactions during training, we focus on the short Pick and Roll (PnR) instances of the games. PnR is one of the most common offensive tactics in the NBA where an offensive player sets a screen for the ball handler, attempting to create separation between the ball handler and his matchup.

We extracted 12k segments from the 2016 season and used 10k, 1k, 1k for training, validation, and testing respectively. The segments are 25 frames long (i.e. 4 seconds) and consist of only 5 nodes: the ball, ball hander, screener, and defensive matchup for each of the players.

[4]The first edge type is "hard-coded" as non-edge and was trained with a prior probability of 0.91. All other edge types received a prior of 0.03 to favor sparse graphs that are easier to visualize. We visualize test data not seen during training.

We trained a CNN encoder and a RNN decoder with 2 edge types. For fair comparison, and because the trajectory continuation is not PnR anymore, the encoder is trained on only the first 17 time steps (as deployed in testing). Further details are in the supplementary material. Results for test MSE are shown in Figure 6. Our model outperforms a baseline LSTM model, and is on par with the full graph.

To understand the latent edge types we show in Fig. 8 how they are distributed between the players and the ball. As we can see, one edge type mostly connects ball and ball handler (off-ball) to all other players, while the other is mostly inner connections between the other three players. As the ball and ball handler are the key elements in the PnR play, we see that our model does learn an important semantic structure by separating them from the rest.

6. Conclusion

In this work we introduced NRI, a method to simultaneously infer relational structure while learning the dynamical model of an interacting system. In a range of experiments with physical simulations we demonstrate that our NRI model is highly effective at unsupervised recovery of ground-truth interaction graphs. We further found that it can model the dynamics of interacting physical systems, of real motion tracking and of sports analytics data at a high precision, while learning reasonably interpretable edge types.

Many real-world examples, in particular multi-agent systems such as traffic, can be understood as an interacting system where the interactions are dynamic. While our model is trained to discover static interaction graphs, we demonstrate that it is possible to apply a trained NRI model to this evolving case by dynamically re-estimating the latent graph. Nonetheless, our solution is limited to static graphs during training and future work will investigate an extension of the NRI model that can explicitly account for dynamic latent interactions even at training time.

Acknowledgements

The authors would like to thank the Toronto Raptors and the NBA for the use of the SportVU data. We would further like to thank Christos Louizos and Elise van der Pol for helpful discussions. This project is supported by SAP SE Berlin.

References

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Li, F., and Savarese, S. Social LSTM: human trajectory prediction in crowded spaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. J., and Kavukcuoglu, K. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*, 2014.

Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations (ICLR)*, 2017.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. In *International Conference on Machine Learning, (ICML)*, 2017.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

CMU. Carnegie-Mellon Motion Capture Database. http://mocap.cs.cmu.edu, 2003.

Dai, H., Dai, B., and Song, L. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning (ICML)*, 2016.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Duvenaud, D. K., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Garcia, V. and Bruna, J. Few-shot learning with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017.

Granger, C. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37, 1969.

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

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Herzig, R., Raboh, M., Chechik, G., Berant, J., and Globerson, A. Mapping images to scene graphs with permutation-invariant structured prediction. 2018. URL http://arxiv.org/abs/1802.05451.

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

Hoshen, Y. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Jain, A., Zamir, A. R., Savarese, S., and Saxena, A. Structural-rnn: Deep learning on spatio-temporal graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kingma, D. P. and Welling, M. Auto encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Kuramoto, Y. Self-entrainment of a population of coupled nonlinear oscillators. In *International Symposium on Mathematical Problems in Theoretical Physics. (Lecture Notes in Physics, vol. 39.)*, pp. 420–422, 1975.

Laszuk, D. Python implementation of Kuramoto systems. http://www.laszukdawid.com/codes, 2017.

Le, H. M., Yue, Y., Carr, P., and Lucey, P. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning, (ICML)*, 2017.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel., R. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Lin, Z., Feng, M., Nogueira dos Santos, C., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. A structured self-attentive sentence embedding. In *International Conference on Learning Representations (ICLR)*, 2017.

Linderman, S. W. and Adams, R. P. Discovering latent network structure in point process data. In *International Conference on Machine Learning (ICML)*, 2014.

Linderman, S. W., Adams, R. P., and Pillow, J. W. Bayesian latent structure discovery from multi-neuron recordings. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Luong, M.-T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations (ICLR)*, 2017.

Monti, F., Boscaini, D., and Masci, J. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning (ICML)*, 2016.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. *NIPS Autodiff Workshop*, 2017.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

Sukhbaatar, S., Szlam, A., and Fergus, R. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations (ICLR)*, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Watters, N., Zoran, D., Weber, T., Battaglia, P., Pascanu, R., and Tacchetti, A. Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.