Factorised Neural Relational Inference for Multi-Interaction Systems

Ezra Webb [1] [*]   **Ben Day** [2] [*]   **Helena Andres-Terre** [2]   **Pietro Lió** [2]

Abstract

Many complex natural and cultural phenomena are well modelled by systems of simple interactions between particles. A number of architectures have been developed to articulate this kind of structure, both implicitly and explicitly. We consider an unsupervised explicit model, the NRI model, and make a series of representational adaptations and physically motivated changes. Most notably we factorise the inferred latent interaction graph into a multiplex graph, allowing each layer to encode for a different interaction-type. This fNRI model is smaller in size and significantly outperforms the original in both edge and trajectory prediction, establishing a new state-of-the-art. We also present a simplified variant of our model, which demonstrates the NRI's formulation as a variational auto-encoder is not necessary for good performance, and make an adaptation to the NRI's training routine, significantly improving its ability to model complex physical dynamical systems.

1. Introduction & Related Work

There are interesting phenomena at every physical scale that are well described by dynamical systems of interacting particles. Thinking about things in this way has proven to be a valuable method for investigating the natural world. As we come to develop more intelligent systems to assist in our investigations, the ability to work within this framework will be of great use.

Many systems have been developed to model interactions implicitly with either single-layer fully connected graphs (Sukhbaatar et al., 2016; Guttenberg et al., 2016; Santoro et al., 2017; Watters et al., 2017) or with attention-based con-

---
[*]Equal contribution   [1]Department of Physics, The Cavendish Laboratory, University of Cambridge, UK. [2]Department of Computer Science & Technology, The Computer Laboratory, University of Cambridge, UK.. Correspondence to: Ben Day <ben.day@cl.cam.ac.uk>.

trol mechanisms (Hoshen, 2017; van Steenkiste et al., 2018). However, to the end of developing an investigative or theorising machine assistant, modelling the interaction graph *explicitly* is more valuable than a high quality trajectory-reconstruction. The neural relational inference (NRI) model, introduced by Kipf et al. (2018b), is an unsupervised neural network that learns to predict the interactions and dynamics of a system of objects from observational data alone. When provided with the trajectories of a system of interacting objects, the model infers an explicit interaction graph for these objects which it uses to predict the evolution of the system.

The NRI model presents a strong foundation, answering key architectural questions and opening the door for further work dealing with explicit representations. In this work we identify two problems within the original formulation – representational and experimental – and in addressing these develop a model that significantly outperforms the original. We also present a variant of our model with greatly improved trajectory prediction that demonstrates the NRI model's formulation as a variational auto-encoder (VAE) is not necessary for good performance.

Specifically, in a system with multiple independent interactions, representing the interaction relationships as a single graph with many edge-types requires exponentially many types to accommodate all possible combinations of interactions. Critically, feedback in such a system is unable to distinguish partially correct and entirely incorrect predictions. In this work we adopt a multiplex structure wherein different interactions are *factorised* into separate layer-graphs, greatly compressing the representation whilst also permitting better directed feedback and improved training.

1.1. NRI in brief

We provide an outline of the NRI architecture with formal definitions given only for those parts that we modify[1]. We adopt the formalism and nomenclature of Kipf et al. (2018b) throughout.

Most simply, the NRI takes the form of a variational-autoencoder (VAE): trajectories are encoded as a latent interaction graph that is decoded when predicting trajec-

---
[1]An extended description with original diagrams is provided as supplementary material.

tories for given initial conditions. A trajectory is a series of features over time, where $\mathbf{x}_i^t$ is the feature vector of the i-th object at step t. The latent interaction graph has K-many edge-types encoded as one-hot vectors, where $\mathbf{z}_{ij}$ is the edge-type vector between nodes (objects) i and j.

Encoder The encoder receives each particle's trajectory as the feature of its corresponding node in a fully-connected graph and produces an edge-type vector for each pair of particles. A graph neural network (GNN) computes a series of message passing operations (Gilmer et al., 2017) and produces a K-dimensional edge-embedding vector $\mathbf{h}_{(i,j)}^2$ for each pair of particles (i, j).[2]

Posterior distribution The edge-type posterior distributions are taken as $q_\theta(\mathbf{z}_{ij}|\mathbf{x}) = \text{softmax}(\mathbf{h}_{(i,j)}^2)$, from which the edge-type vectors $\mathbf{z}_{ij}$ are sampled, where θ summarizes the parameters of the full encoder GNN.

Decoder The task of the decoder is to predict the dynamics of the system using the latent interaction graph $\mathbf{z}$ and the past dynamics. We consider the Markovian case; calculating $p_\phi(\mathbf{x}^{t+1}|\mathbf{x}^t; \mathbf{z})$. The message passing section consists of

$$v \to e: \tilde{\mathbf{h}}_{(i,j)}^t = \sum_{k=1}^{K} z_{ij,k} \tilde{f}_e^k\big([\mathbf{x}_i^t, \mathbf{x}_j^t]\big) \tag{1}$$

$$e \to v: \boldsymbol{\mu}_j^{t+1} = \mathbf{x}_j^t + \tilde{f}_v\big([\textstyle\sum_{i \neq j} \tilde{\mathbf{h}}_{(i,j)}^t, \mathbf{x}_j^t]\big) \tag{2}$$

where $[\cdot, \cdot]$ denotes concatenation. We note that each edge-type k has its own function in the edge-to-vertex message passing operation – $\tilde{f}_e^1, \ldots, \tilde{f}_e^K$. The future state of each object is then sampled from an isotropic Gaussian distribution with fixed (user-defined) variance σ^2

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

Objective The model is trained as a VAE maximising the evidence lower bound

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

where D_{KL} is the Kullback-Leibler (KL) divergence. It is also relevant to note that the reconstruction error is estimated by a re-scaled mean-squared error (MSE) of $\boldsymbol{\mu}$ relative to $\mathbf{x}$.[3]

2. Model

2.1. Factorised Neural Relational Inference

Here we introduce our reformulation of the NRI model which we will refer to as the factorised neural relational

---

[2] $\mathbf{h}_{(i,j)}^2$ is used to align with the original paper's notation.
[3] Re-scaled by the hyperparameter $\frac{1}{2\sigma^2}$ (plus a constant).

---

inference (fNRI) model. In this model the NRI's single latent interaction graph with K edge-types is *factorised* into an n-layer multiplex graph (see figure 1), where the a-th layer-graph has K_a edge-types.

The K-dimensional edge-embedding vector $\mathbf{h}_{(i,j)}^2$ returned by the NRI encoder (as in equation **??**) is first segmented

$$\mathbf{h}_{(i,j)}^2 = \big[\mathbf{h}_{(i,j)}^{2,1}, \ldots, \mathbf{h}_{(i,j)}^{2,n}\big] \tag{4}$$

where segment $\mathbf{h}_{(i,j)}^{2,a}$ is a K_a-dimensional vector and $K = \sum_{a=1}^{n} K_a$ is the total number of edge types. The posterior distribution for each layer-graph is then formed as

$$q_\theta(\mathbf{z}_{ij}^a|\mathbf{x}) = \text{softmax}(\mathbf{h}_{(i,j)}^{2,a}) \tag{5}$$

where $\mathbf{z}_{ij}^a$ denotes the one-hot edge-type vector between objects i and j in the a-th layer-graph. As in the NRI, during training the vectors are sampled from a 'continuous relaxation' of their respective posterior distributions using the concrete distribution (Maddison et al., 2017)

$$\mathbf{z}_{ij}^a = \text{softmax}\big((\mathbf{h}_{(i,j)}^{2,a} + \mathbf{g})/\tau\big).$$

where $\mathbf{g} \in \mathbb{R}^{K_a}$ is a vector of i.i.d samples drawn from a Gumbel$(0, 1)$ distribution and τ is the 'softmax temperature.' Concatenating these vectors forms the combined edge-type vector of the multiplex interaction graph

$$\mathbf{z}_{ij} = [\mathbf{z}_{ij}^1, ..., \mathbf{z}_{ij}^n]. \tag{6}$$

These $\mathbf{z}_{ij}$ are no longer one-hot vectors, but rather multi-categoric with $\sum_k z_{ij,k} = n$, and are supplied to the NRI decoder as described in 1.1. In alignment with the NRI model, the latent graphs are not forced to be undirected ($\mathbf{z}_{ij}$ may not necessarily equal $\mathbf{z}_{ji}$), and if desired, the first edge-type of each layer-graph can be hard-coded as the non-edge. The KL-divergence term in the ELBO is the sum of KL-divergences over the layer-graphs.

$$D_{\text{KL}}[q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] = \sum_{a=1}^{n} D_{\text{KL}}[q_\theta(\mathbf{z}^a|\mathbf{x})||p(\mathbf{z}^a)] \tag{7}$$

2.1.1. MOTIVATIONS

We now expand on the motivations given earlier in light of the model specifications. As the NRI uses a one-hot latent encoding, in multi-interaction systems single edge-types must exist to represent any possible combination of interactions (e.g. *spring+charge*). In contrast, the fNRI edge-types need only encode for one interaction-type, with combinations arising naturally from the multiplex structure. The edge-type decoder networks $\tilde{f}_e^k$ in equation (1) therefore only need to decode the dynamics of a single type of interaction. We theorise this compartmentalisation of the interactions will improve training in complex systems, especially given
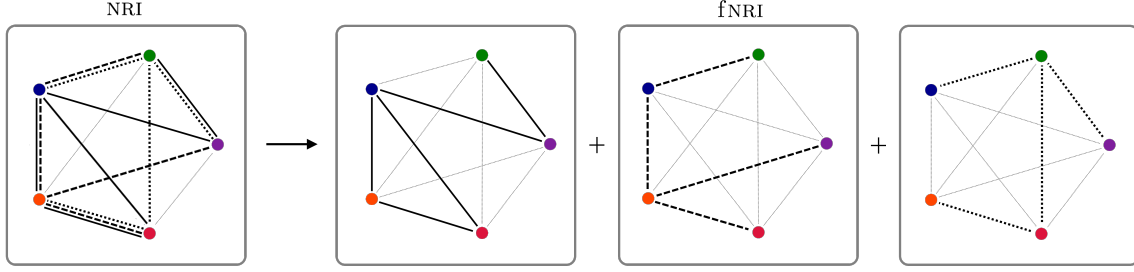
Figure 1. Schematic showing the representational change in the interaction graph between the NRI and fNRI models when there are three independent interaction types represented by solid, dashed and dotted lines, in addition to no interaction, represented by thin grey lines. In the NRI model, the possible combinations of interactions require eight ($= 2^3$) edge-types.

that each of the networks $\tilde{f}_e^k$ will effectively have a larger training set in our formulation. This is because the $\tilde{f}_e^k$ are used by the decoder in *every* instance that its corresponding interaction is present, rather than when a specific combination of interactions is present, as in the NRI. Or in other words, because the density of the latent representation is exponentially greater in the fNRI model. This increase in latent information density also means that factorised model decoders have notably fewer parameters.

In addition, the fNRI model has the capacity to be explicitly fractionally correct about an edge-type. If the encoder correctly predicts one underlying interaction type, but the other incorrectly, in the NRI model the corresponding $\mathbf{z}_{ij}$ is plainly 'incorrect'. However in the fNRI model, the corresponding $\mathbf{z}_{ij}$ will be half-right and treated accordingly, in theory allowing for better directed feedback.

Compartmentalising interactions in the fNRI model will also be useful when attempting to understand the meanings of edge-types in systems where the underlying interactions are unknown. An issue that could be raised with the fNRI model is that in such contexts, due to having $\{K_1, ..., K_n\}$ edge-types rather than just K, the dimensionality of the hyperparameter space has been increased. However, picking $K_a = 2$ for all a allows for the same dimensionality while retaining all functionality, where interactions with more than two discrete edge-types (e.g. colour-charge) are encoded over multiple layer-graphs.

2.2. Sigmoid Factorisation

We also investigate a drastic simplification of the fNRI model, where each layer-graph effectively only contains a single edge-type and probabilistic sampling is removed completely. In this sfNRI model, rather than using the edge-embedding vectors $\mathbf{h}^2_{(i,j)}$ returned by the encoder to form posterior distributions, they are directly transformed into K-dimensional edge-type vectors by a sigmoid function $\mathbf{z}_{ij} = \sigma(\mathbf{h}^2_{(i,j)})$. These $\mathbf{z}_{ij}$ are then decoded using the same decoder described in section 1.1. In this model there are K layer-graphs, each of which contains a *single* edge-type, in

addition to an explicit non-edge.

As the sampling aspect of the model is removed, the elements of the edge-type vectors $z_{ij,k}$ are no longer strictly binary elements of $\{0, 1\}$, but rather are elements of $[0, 1]$. Furthermore, it is no longer possible to define a KL-divergence so the loss function is just the reconstruction error – a rescaling of the mean squared error between the predicted and ground-truth trajectories.

The motivations here are much the same as for the fNRI; allow each element of the edge-type vector $\mathbf{z}_{ij}$ to represent a separate interaction edge that can be observed in combination. Additionally, the non-interaction edge becomes a more fundamental part of the model. When there are no interactions between a pair of particles, the ground truth edge-vector will, in theory, be all zeros, $\mathbf{z}_{ij} = \mathbf{0}$. This follows as if a particle has no interactions, then the elements of the vector $\sum_{i \neq j} \tilde{\mathbf{h}}^t_{(i,j)}$ in equation (2) will all be zero, and the only non-zero entries to the neural network $\tilde{f}_v$ will be the current state of the particle $\mathbf{x}^t_j$. This means that the non-interaction graph (where there are no interactions between particles) is made explicit by the very architecture of the model, as $\mathbf{z}$ will contain only zeros and therefore each particle's predicted future state $\boldsymbol{\mu}^{t+1}_j$ can *only* depend on its current state $\mathbf{x}^t_j$.

3. Experiments

To make our comparison with the original NRI model as convincing as possible, unless otherwise stated we use the exact same hyperparameters as detailed in the original paper (Kipf et al., 2018b), full details of which can be found in the supplementary material. The only change we make to the training routine is discussed in section 3.1.[4]

We experiment with simulated systems of 5 interacting particles in a finite 2D box. In these systems particles are 'randomly connected' by different physical interactions. We

---

[4]Our implementation is available in full at https://github.com/ekwebb/fNRI.

Table 1. Accuracy (%) in recovering the ground truth interaction graph. Higher is better.

Accuracy	I-Springs+Charges			I-Springs+Charges+F-springs			
	Combined	I-Springs	Charges	**Combined**	I-Springs	Charges	F-Springs
Random	25.0	50.0	50.0	12.5	50.0	50.0	50.0
NRI (learned)	89.1 ± 0.4	97.9 ± 0.0	91.0 ± 0.4	57.9 ± 6.1	88.5 ± 0.9	87.3 ± 6.2	70.7 ± 2.3
fNRI (learned)	$\mathbf{94.0} \pm \mathbf{1.4}$	98.0 ± 0.1	95.8 ± 1.3	$\mathbf{63.3} \pm \mathbf{6.5}$	86.9 ± 2.7	97.7 ± 0.7	69.2 ± 5.5
sfNRI (learned)	88.8 ± 0.8	97.6 ± 0.1	91.1 ± 0.8	45.1 ± 5.1	90.0 ± 2.3	98.2 ± 0.8	52.4 ± 2.7
NRI (supervised)	$\mathbf{98.3} \pm \mathbf{0.0}$	98.6 ± 0.0	99.7 ± 0.0	80.9 ± 0.7	92.4 ± 0.3	99.0 ± 0.1	84.4 ± 0.4
fNRI (supervised)	$\mathbf{98.3} \pm \mathbf{0.0}$	98.8 ± 0.4	99.4 ± 0.4	$\mathbf{81.8} \pm \mathbf{0.1}$	93.3 ± 0.1	99.3 ± 0.0	85.8 ± 0.1
sfNRI (supervised)	98.0 ± 0.0	98.3 ± 0.0	99.6 ± 0.0	81.0 ± 0.3	92.9 ± 0.1	99.2 ± 0.0	85.2 ± 0.2

Table 2. Mean squared error (MSE) $/ 10^{-5}$ in trajectory prediction. Lower is better.

Predictions Steps	I-Springs+Charges			I-Springs+Charges+F-Springs		
	1	10	20	1	10	20
Static	19.4	283	783	12.8	274	782
NRI (learned)	0.88 ± 0.06	4.05 ± 0.22	$11.5 \ \pm 0.5$	0.95 ± 0.05	8.67 ± 0.45	$29.1 \ \ \pm 1.4$
fNRI (learned)	$\mathbf{0.80} \pm \mathbf{0.04}$	3.54 ± 0.09	9.93 ± 0.29	0.81 ± 0.05	7.78 ± 0.20	$26.8 \ \ \pm 0.8$
sfNRI (learned)	1.03 ± 0.09	$\mathbf{3.32} \pm \mathbf{0.23}$	$\mathbf{9.68} \pm \mathbf{0.74}$	$\mathbf{0.77} \pm \mathbf{0.03}$	$\mathbf{5.69} \pm \mathbf{0.21}$	$\mathbf{19.3} \ \ \pm \mathbf{0.8}$
NRI (true graph)	0.85 ± 0.04	1.59 ± 0.26	3.20 ± 0.15	0.75 ± 0.02	1.55 ± 0.07	3.43 ± 0.21
fNRI (true graph)	$\mathbf{0.70} \pm \mathbf{0.03}$	$\mathbf{1.30} \pm \mathbf{0.06}$	$\mathbf{2.52} \pm \mathbf{0.11}$	$\mathbf{0.51} \pm \mathbf{0.05}$	0.97 ± 0.08	2.44 ± 0.28
sfNRI (true graph)	0.86 ± 0.09	1.32 ± 0.06	2.77 ± 0.07	0.56 ± 0.04	$\mathbf{0.89} \pm \mathbf{0.06}$	$\mathbf{2.28} \pm \mathbf{0.15}$

consider three different types of physical interaction: *ideal springs* (I-springs) where particles are randomly connected by Hookean springs of zero length, *finite springs* (F-springs) where particles are randomly connected by Hookean springs of a fixed finite length, and *charges* where particles are randomly selected to be either positively charged or neutral, and charged particles interact via Coulomb's law.

3.1. Compression Models

A problem we encounter with the NRI training routine is that when attempting to learn more complex interaction graphs, the encoder can instead learn to use the latent space to store a compressed version of the input trajectories. It appears that this can occur to a varying degree, however the problem worsens as the size, and thus the expressiveness, of the latent space increases. These models are easily identified during testing as they are non-predictive, meaning they can only reconstruct the trajectories the encoder received as input.

In order to avoid these compression models, we modify the training routine such that the encoder receives the first half of the particle trajectories, and the decoder predicts the second half of the particle trajectories. For interacting systems with static interaction graphs, this change is reasonable, and has a number of distinct advantages. Firstly, compression solutions are avoided as the models are now trained to predict *unobserved* trajectories, only. As such, training becomes significantly more reliable and far less dependent on the model initialisation. Secondly, the difference in the

reconstruction loss between the training and validation sets is reduced, and we observe a reduction in overfitting. Making this change means the network is formally no longer acting as an auto-encoder, as the decoder network does not learn by *reconstructing* the encoder input **x**, but rather by generating a time-evolution of **x**, which is then compared to the ground-truth time-evolution. We use this modification when training *all* the models presented here. Without it, training is simply not reliable enough, with edge-accuracies often failing to rise above the random level.

4. Results

The edge and trajectory prediction results are summarised in tables 1 and 2 respectively, where each result is the average over 5 runs with the standard error given. In all cases the factorised NRI models match or outperform the original.

For both edge and trajectory prediction, we compare the unsupervised *learned* models to the supervised 'gold standards.' For edge prediction the *supervised* encoders are trained in isolation on the ground-truth interaction graphs, and for trajectory prediction the *true graph* decoders are trained in isolation with the ground-truth interaction graphs their inputs. The static decoder simply returns the state vector it receives as input. For edge prediction, accuracies are decomposed into the prediction accuracy for each interaction type. The combined accuracy is calculated such that it only receives a contribution when the predicted edges between a pair of nodes are correct for all interaction types.

Acknowledgements

References

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/gilmer17a.html.

Guttenberg, N., Virgo, N., Witkowski, O., Aoki, H., and Kanai, R. Permutation-equivariant neural networks applied to dynamics prediction. 12 2016. URL http://arxiv.org/abs/1612.04530.

Hoshen, Y. VAIN: Attentional Multi-agent Predictive Modeling. 6 2017. URL http://arxiv.org/abs/1706.06122.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. 12 2014. URL http://arxiv.org/abs/1412.6980.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural Relational Inference for Interacting Systems. 2 2018a. URL http://arxiv.org/abs/1802.04687.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural Relational Inference for Interacting Systems. 2 2018b. URL http://arxiv.org/abs/1802.04687.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete distribution: a continuous relaxation of discrete random variables. 2017. URL http://arxiv.org/abs/1611.00712.

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. In *NIPS-W*, 2017.

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. 6 2017. URL http://arxiv.org/abs/1706.01427.

Sukhbaatar, S., Szlam, A., and Fergus, R. Learning Multiagent Communication with Backpropagation. 5 2016. URL http://arxiv.org/abs/1605.07736.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. 2 2018. URL http://arxiv.org/abs/1802.10353.

Watters, N., Tacchetti, A., Weber, T., Pascanu, R., Battaglia, P., and Zoran, D. Visual Interaction Networks. 6 2017. URL http://arxiv.org/abs/1706.01433.

Factorised Neural Relational Inference for Multi-Interaction Systems: Supplementary Material

Overview

These supplementary materials are provided to support the workshop paper 'Factorised Neural Relational Inference for Multi-Interaction Systems' published at the Learning and Reasoning with Graph-Structured Data workshop at ICML 2019.

The materials include an extended description of the NRI model in section 5, details of the physics simulations and experimental procedures in sections 6 and 7, and a note on calculating edge accuracy in unsupervised systems is added in section 8.

5. Neural Relational Inference

Here we describe the NRI model as presented by Kipf et al. (2018a) along with our own schematic and comments. Here we provide an extended description of the NRI model; adopting the formalism and nomenclature of Kipf et al. (2018a) throughout. The NRI model takes the generalised form of a variational auto-encoder (VAE), where the encoding network infers a latent interaction graph for the system, and the decoding network predicts the future dynamics of the system using this interaction graph. This graph is described by a set of edge-types $\mathbf{z}$ (the latent variables of the VAE) which tell the decoding network about the types of interactions between each pair of particles.

The NRI model differs from the standard VAE implementation in a number ways. Most notably, it does not use a continuous isotropic multivariate Gaussian distribution as its prior. Rather, its prior distribution is *discrete*; and the encoder returns a probability vector for the edge-type between each pair of particles. The edge-types $\mathbf{z}$ in the latent interaction graph are then sampled from these probability vectors (see figure 2).

In order for the NRI model to be successful in predicting the future dynamics of a system, the underlying interactions of the system must be *discrete*. In the context of physics, this means that the interactions must be discrete in both form and strength. For example, if we have a box containing a collection of interacting charged particles, the NRI model has the potential to successfully model the dynamics of this system provided the strengths of the charges are picked from some finite set, rather than being picked from a continuum. The reason for this is that the number of edge-types K in the

latent interaction graph, is a hyperparameter of the model and represents the number of distinct 'interaction types' the model will be able to encode for. If the strength of the charges are drawn from a continuum, although interactions will be discrete in form (with all the forces between particles being proportional to the inverse square of their separation), an interaction graph cannot be drawn for the system using a discrete set of edge-types.[5]

In the latent interaction graph, the edge-type between objects i and j is encoded for using a one-hot vector of length K, denoted $\mathbf{z}_{ij}$. This means each edge in the interaction graph is one of K *discrete* edge-types, formalised as $\sum_{k=1}^{K} z_{ij,k} = 1$, where $z_{ij,k} \in \{0, 1\}$ denotes the k-th element of the vector $\mathbf{z}_{ij}$.

5.1. Message Passing Operation

The encoding and decoding networks in the NRI model are described as graph neural networks (GNNs). These are a broad class of artificial neural networks which operate on graph structured data and are defined by their use of the 'message passing' operation introduced by Gilmer et al. (2017). For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $v \in \mathcal{V}$ and edges $e = (v, v') \in \mathcal{E}$, where vertex v_i has features $\mathbf{x}_i$ and edge $e_{(i,j)}$ has features $\mathbf{x}_{(i,j)}$, a single node-to-node message passing operation is defined as

$$v \to e: \quad \mathbf{h}_{(i,j)}^l = f_e^l\left(\left[\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}\right]\right) \qquad (8)$$

$$e \to v: \quad \mathbf{h}_j^{l+1} = f_v^l\left(\left[\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j\right]\right) \qquad (9)$$

where $\mathbf{h}_j^l$ is the embedding of the features of vertex v_i in layer l of the GNN and $\mathbf{h}_{(i,j)}^l$ is the embedding of the features of edge $e_{(i,j)}$ in layer l of the GNN. These edge feature embeddings are sometimes referred to as a 'messages'. $\mathcal{N}_j$ denotes the set of indices of vertices which are connected to vertex v_j 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 respectively, for example small multi-layer perceptrons (MLPs). We note that the message passing operation operates on the edge and node *features*, and does not alter the shape of the graph.

---

[5]At least one cannot be drawn using a set of edge-types that is smaller than the total number of edges in the interaction graph.
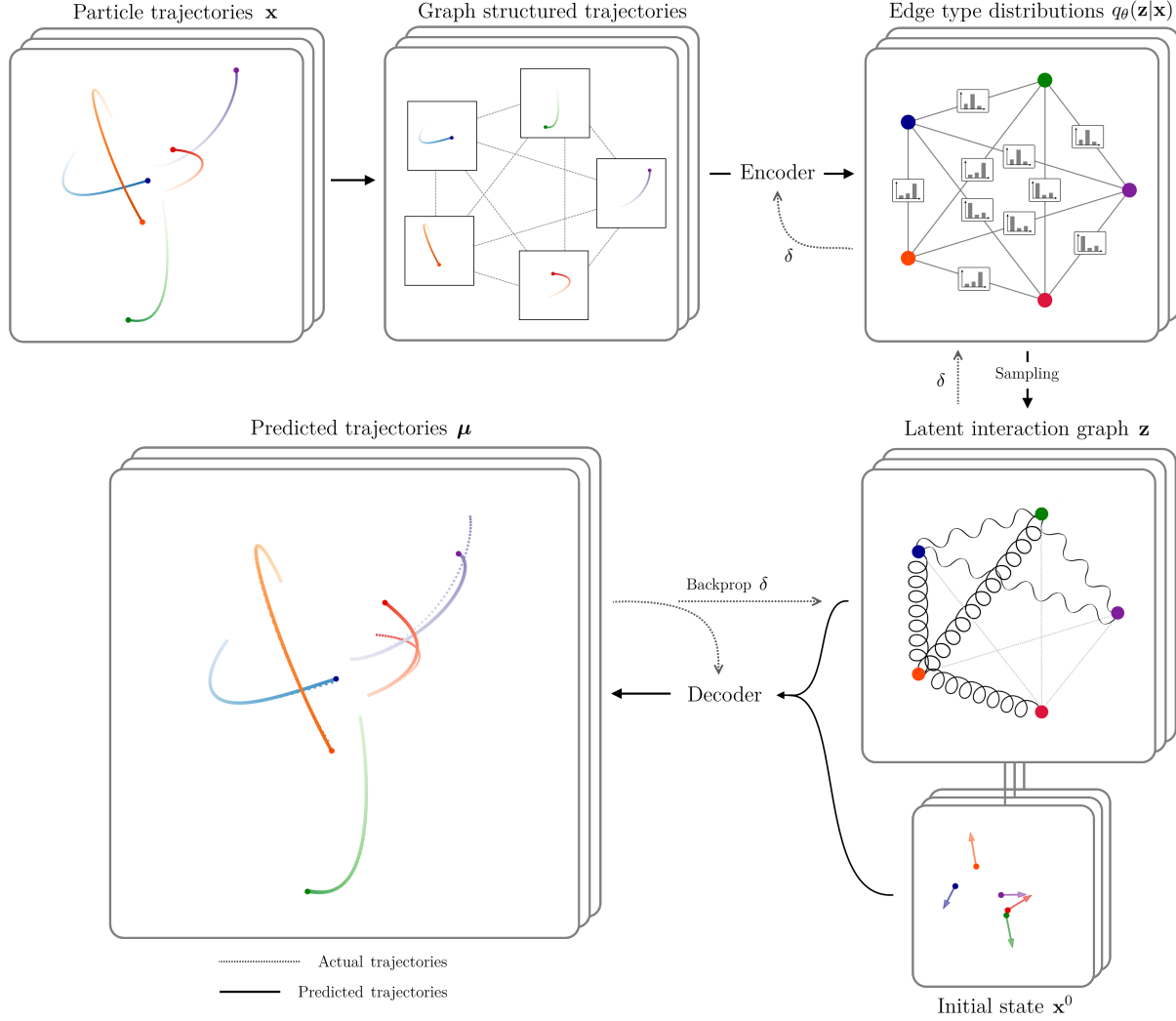
Figure 2. Schematic of the batch-wise NRI training procedure, where the dashed arrows with δ s indicate backpropagation. The system in the schematic has 5 interacting particles and three inferred edge-types. In the latent interaction graph **z** these three different edge-types are represented by sinusoidal lines, curly lines and thin grey lines.

5.2. Encoder

The input of the encoder consists of the trajectories of N objects. We denote the feature vector of object i at time t by $\mathbf{x}_i^t$; in our work this vector contains the location and velocity of the particle. We denote the set of all N objects at time t by $\mathbf{x}^t = \{\mathbf{x}_1^t, ..., \mathbf{x}_N^t\}$, the trajectory of object i by $\mathbf{x}_i = (\mathbf{x}_i^1, ..., \mathbf{x}_i^T)$ and the set of all trajectories by $\mathbf{x} = (\mathbf{x}^1, ..., \mathbf{x}^T)$, where T is the total number of time steps.

The trajectory of each particle enters the encoder as the features of a node in a fully-connected graph of N nodes, where each node represents one of the interacting objects. Using the message passing operations defined in section 5.1, the action of the encoding network on this graph can be

defined as follows:

$$\mathbf{h}_i^1 = f_{\text{emb}}(\mathbf{x}_i) \qquad (10)$$

$$v \to e: \quad \mathbf{h}_{(i,j)}^1 = f_e^1\big([\mathbf{h}_i^1, \mathbf{h}_j^1]\big) \qquad (11)$$

$$e \to v: \quad \mathbf{h}_j^2 = f_v^1\big(\textstyle\sum_{i \neq j} \mathbf{h}_{(i,j)}^1\big) \qquad (12)$$

$$v \to e: \quad \mathbf{h}_{(i,j)}^2 = f_e^2\big([\mathbf{h}_i^2, \mathbf{h}_j^2]\big) \qquad (13)$$

The edge-type posterior distributions are then taken as $q_\theta(\mathbf{z}_{ij}|\mathbf{x}) = \text{softmax}(\mathbf{h}_{(i,j)}^2)$, where $\mathbf{h}_{(i,j)}^2 \in \mathbb{R}^K$ and θ summarizes the parameters of the neural networks in equations (10)-(13). By studying equations (10)-(13), it can be noted that as the input graph is fully connected, the node embeddings $\mathbf{h}_j^2$ and subsequent edge embeddings $\mathbf{h}_{(i,j)}^2$ are influenced by the trajectories of all the particles in the system.

The neural networks f_{emb}, f_e^1 and f_v^1 are 2-layer MLPs with hidden and output dimension 256, batch normalization, and ELU activations. The last neural network f_e^2 has these same properties with the addition of an extra dense layer of output dimension K.

5.3. Sampling

A softmax function is used to transform the edge feature embedding $\mathbf{h}_{(i,j)}^2$ into a posterior distribution $q_\theta(\mathbf{z}_{ij}|\mathbf{x})$. However, sampling directly from this distribution is not a differentiable process. To circumvent this problem, the NRI model uses a 'continuous relaxation' of the discrete posterior distribution in the form of the concrete distribution (Maddison et al., 2017), which *reparametrises* the sampling using the Gumbel distribution. This means rather than sampling the edge-type vectors $\mathbf{z}_{ij}$ directly from posterior as

$$\mathbf{z}_{ij} \sim q_\theta(\mathbf{z}_{ij}|\mathbf{x}) = \text{softmax}(\mathbf{h}_{(i,j)}^2) \quad (14)$$

The edge-type vectors are sampled using

$$\mathbf{z}_{ij} = \text{softmax}\big((\mathbf{h}_{(i,j)}^2 + \mathbf{g})/\tau\big) \quad (15)$$

where $\mathbf{g} \in \mathbb{R}^K$ is a vector of independent samples drawn from a Gumbel(0,1) distribution and τ is the softmax temperature. This is a *continuous relaxation* of the discrete posterior distribution $q_\theta(\mathbf{z}_{ij}|\mathbf{x})$ as the edge-type vectors $\mathbf{z}_{ij}$ returned by equation (15) are not one-hot, but rather smoothly converge to one-hot vectors sampled from $q_\theta(\mathbf{z}_{ij}|\mathbf{x})$ in the limit $\tau \to 0$.

5.4. Decoder

The task of the decoder is to predict the future dynamics of the system using the latent interaction graph and the past dynamics. Formally, this means calculating the likelihood $p_\phi(\mathbf{x}^{t+1}|\mathbf{x}^t; ...; \mathbf{x}^1; \mathbf{z})$. In our work we only consider systems where the dynamics are Markovian, meaning the dependence in the likelihood reduces to $p_\phi(\mathbf{x}^{t+1}|\mathbf{x}^t; \mathbf{z})$.

In the NRI model, each edge-type has a separate neural network in the edge-to-vertex message passing operation. The message passing section of the Markovian decoder is formalised as:

$$v \to e : \tilde{\mathbf{h}}_{(i,j)}^t = \sum_{k=1}^K z_{ij,k} \tilde{f}_e^k\big([\mathbf{x}_i^t, \mathbf{x}_j^t]\big) \quad (16)$$

$$e \to v : \boldsymbol{\mu}_j^{t+1} = \mathbf{x}_j^t + \tilde{f}_v\big([\textstyle\sum_{i \neq j} \tilde{\mathbf{h}}_{(i,j)}^t, \mathbf{x}_j^t]\big) \quad (17)$$

The future state of each object is then sampled from an isotropic Gaussian distribution with a mean vector $\boldsymbol{\mu}_j^{t+1}$ and a fixed (user-defined) variance σ^2:

$$p_\phi(\mathbf{x}_j^{t+1}|\mathbf{x}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_j^{t+1}, \sigma^2\mathbf{I}) \quad (18)$$

Note that in equation (16), when the edge-type vector $\mathbf{z}_{ij}$ is one-hot, $\tilde{\mathbf{h}}_{(i,j)}^t$ only receives a contribution from the neural network representing the 'hot' edge-type, but for continuous relaxations, the message is a weighted sum. We note that the first edge-type can be 'hard-coded' to be the non-edge, representing no interaction between particles, by modifying the sum in equation (16) to start at $k = 2$.

When the dynamics of the system are not Markovian, a recurrent neural network can be used in the decoder to use the full history of the particle in predicting its future dynamics.

5.5. Training

The NRI model takes the form of a variational auto-encoder and it is therefore trained to maximise the evidence lower bound

$$\mathcal{L} = \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\phi(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}[q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \quad (19)$$

where the likelihood $p_\phi(\mathbf{x}|\mathbf{z})$ can be expanded as $p_\phi(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^T p_\phi(\mathbf{x}^{t+1}|\mathbf{x}^t; \mathbf{z})$, and the prior $p(\mathbf{z}) = \prod_{i \neq j} p(\mathbf{z}_{ij})$ is generally a factorised uniform distribution over edge-types. For a uniform prior, $p(z_{ij,k}) = 1/K$, the overall KL-divergence in the ELBO function is given by

$$D_{KL}[q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] = \sum_{i \neq j} \Big[-H\big[q_\theta(\mathbf{z}_{ij}|\mathbf{x})\big] + \log K \Big] \quad (20)$$

where $H\big[q_\theta(\mathbf{z}_{ij}|\mathbf{x})\big]$ is the entropy of the posterior distribution $q_\theta(\mathbf{z}_{ij}|\mathbf{x})$. The reconstruction error in the ELBO is estimated by

$$\mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\phi(\mathbf{x}|\mathbf{z})] = -\sum_j \sum_{t=2}^T \frac{||\mathbf{x}_j^t - \boldsymbol{\mu}_j^t||^2}{2\sigma^2} + \text{const} \quad (21)$$

This reconstruction error only depends on single time step predictions. However, the interactions between objects often only have a small effect on the short term dynamics. This means the decoder could quite easily learn to ignore the latent interaction graph, whilst achieving only a marginally worse reconstruction error. In order to avoid these 'degenerate' decoders, the NRI model predicts the dynamics multiple time-steps in to future. Denoting the decoder as $\boldsymbol{\mu}_j^{t+1} = f_{\text{dec}}(\mathbf{x}_j^t)$, the NRI model implements this by replacing the actual system state $\mathbf{x}^t$ with the previous predicted mean state $\boldsymbol{\mu}_j^t$ for M time-steps. Doing this means that any errors in the reconstruction accumulate over M steps, which makes correctly predicting the latent interaction graph essential for maximising the ELBO. This procedure can be

formalised as

$$\boldsymbol{\mu}_j^2 = f_{\text{dec}}(\mathbf{x}_j^1)$$
$$\boldsymbol{\mu}_j^{t+1} = f_{\text{dec}}(\boldsymbol{\mu}_j^t) \qquad t = 2, ..., M$$
$$\boldsymbol{\mu}_j^{M+2} = f_{\text{dec}}(\mathbf{x}_j^{M+1})$$
$$\boldsymbol{\mu}_j^{t+1} = f_{\text{dec}}(\boldsymbol{\mu}_j^t) \qquad t = M + 2, ..., 2M$$
$$...$$

If we have some prior knowledge of the system, this can be included in the form a non-uniform prior. For example, when the first edge-type is hard-coded to be the non-edge, a non-uniform prior with a higher probability on the non-edge could be used to encourage sparser graphs.

6. Simulations

In accordance with the work by Kipf et al. (2018a), we simulate $N = 5$ point mass particles in a finite 2D box, where collisions with the box wall are elastic and there are no external forces. The initial locations of the particles are sampled from a Gaussian distribution $\mathcal{N}(0, 0.5)$, and the initial velocity of each particle is a random vector with norm 0.5. We consider 3 different types of particle interactions in this investigation:

Ideal spring interactions   where particles connected by an ideal spring are acted on by forces given by Hooke's law

$$\mathbf{F}_{ij} = -k_I(\mathbf{r}_i - \mathbf{r}_j) \tag{22}$$

where $\mathbf{F}_{ij}$ is the force applied to particle i by particle j, k_I is the spring constant, and $\mathbf{r}_i$ is the 2D location vector of particle i. These are 'ideal springs' (I-springs) because they have zero length and are therefore only attractive.

Finite spring interactions   where particles connected by a finite length spring are acted on by forces given by a modified Hooke's law

$$\mathbf{F}_{ij} = -k_F\Big(\mathbf{r}_i - \mathbf{r}_j - l \cdot \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|}\Big) \tag{23}$$

where k_F is the spring constant and l is the spring length. The forces these finite length springs (F-springs) generate between particles can be attractive or repulsive.

Charge interactions   where charged particles are acted on by forces given by Coulomb's Law

$$\mathbf{F}_{ij} = q_i q_j C \cdot \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3} \tag{24}$$

where C is a positive constant and q_i is the charge of particle i. Due to the simulation instabilities that arise when
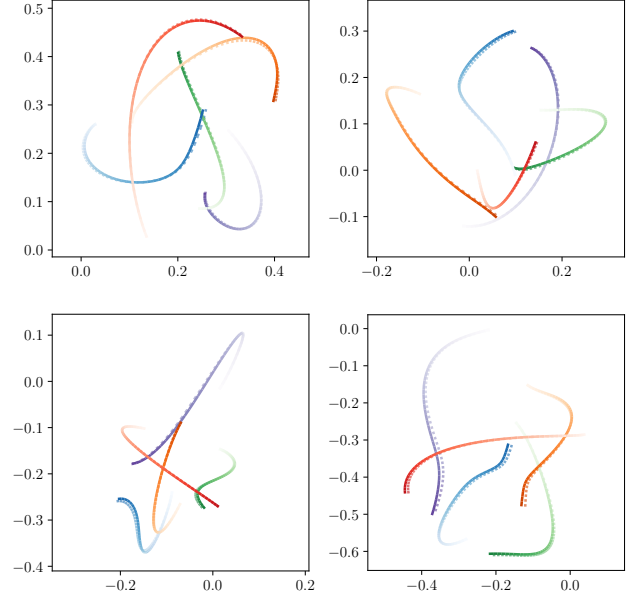

Figure 3. Trajectories of four sets of 5 interacting particles in the I+C system for 50 time-steps, where the predicted trajectories are solid lines, the ground truth trajectories are dashed lines and the line colour gets darker along each particle's trajectory. The predicted trajectories were generated by the fNRI (learned) model using the edge-types inferred by the encoder on the prior 50 time-steps (not shown) and the initial state *only* (i.e. predicted steps = 50). In each of these examples the edge-accuracy was 100%.

divergent forces are present, in our investigation we only consider repulsive charge interactions where $q_i \in \{0, +1\}$.

We combine these three interaction types in two different ways to form two types of simulated system. In the I+C system, ideal spring and charge interactions are randomly added between particles, and in the I+C+F system, ideal spring, charge and finite spring interactions are randomly added between particles. The procedure for this random interaction assignment is described below.

Particle trajectories (see figure 3) are generating by solving Newton's equations of motion using leapfrog integration with a time-step of 1.0 ms. To obtain our training, validation and testing datasets, these trajectories are sub-sampled every 100 time-steps. For each simulated system we generate 50k training examples, 10k validation examples and 10k test examples, where each example contains 100 time-samples with a step size of 0.1 s.

The only major change we make in generating our simulations relative to those generated by Kipf et al. (2018a) is as follows. For each example, rather than randomly connecting each pair of particles by a spring with probability 0.5, the number of springs $n_s \in \{0, 1, ..., \frac{1}{2}N(N-1)\}$ is drawn from a uniform distribution. Particles are then randomly connected using this number of springs. This means the

probability a pair of particles is connected by a spring is still 0.5, while providing a significantly greater variety of interaction graphs. This is desirable as it means a decoder which learns some kind of 'average interaction' will perform poorly. Furthermore, when particles are instead randomly connected with probability 0.5, the total number of springs follows a binomial distribution. It is possible the model could learn to use this fact to preferentially assign a number of springs close to the centre of this distribution. This could artificially inflate the obtained edge accuracies and mean that a trained model is less successful when it is used to predict the dynamics of less familiar interaction graphs.

We apply a similar technique when assigning charges. Rather than assigning a charge to each particle with probability 0.5; the number of charges $n_c \in \{1, ..., N\}$ is drawn from a uniform distribution, then this number of particles are randomly assigned positive charges.

In both of the I+C and I+C+F systems, constants k_I, k_F, l and C are the same for all interactions and are kept constant between systems (with $k_I = k_F = 0.1$ Nm^{-1}, $C = 0.2$ Nm^2 and $l = 1$ m). The particles of mass 1 Kg, interact in a square 2D box (side-length 5 m, centred on the origin) where their initial locations are sampled from an isotropic 2D Gaussian distribution $\mathcal{N}(0, 0.5$ m$)$ and the initial velocity of each particle is a random vector with fixed length 0.5 ms^{-1}.

7. Experimental Details

In all experiments the models were optimised using the Adam algorithm (Kingma & Ba, 2014) with a learning rate of 0.0005, decayed by a factor of 0.5 every 200 epochs. All experiments were run for 500 training epochs using a batch size 128 with shuffling. For *learned* and *true graph* models, checkpointing used the reconstruction loss on the validation set for 10 prediction steps. For *supervised* models, checkpointing used the edge accuracy on the validation set. In the NRI and fNRI models, the concrete distribution was used with a softmax temperature $\tau = 0.5$.

In the work by Kipf et al. (2018a) edge-types are inferred by observing the trajectories for 50 time-steps of size 0.1 s. These same 50 time-steps are then supplied to decoder for reconstruction. We modify this training routine by supplying the trajectories of the first 50 time-steps to the encoder and the *next* 50 time-steps to the decoder. In order to do this, the simulations we generate are twice as long as the training and validation trajectories used by Kipf et al. This modification is used when training *all* the models in this work.

For all artificial neural networks we use the same architecture and hyperparameters as Kipf et al. (2018a); using hidden and output dimensions of 256, batch-normalization and ELU activations. During training of the decoder, we use an M value of 10, meaning every 10th time-step the

decoder receives a ground truth state. We note that in order to prevent exploding gradients in the encoder of the sfNRI model when training on the I+C system, a tiny amount of L2 regularisation was added to the loss function (5e-8 for learned, 2e-5 for supervised).

Table 3 compares the size of the different models in terms of number of parameters and summarises the number of edge-types used in each model in our experiments. These K and K_a values were chosen as for each model they allow for a complete description of the interactions present in each system without redundancy.

Table 3. Summary of the number of edge-types used by each model (i.e. the dimension K of edge-type vectors $\mathbf{z}_{ij}$) as well as the total number of parameters in the encoder and decoder of each model.

	I+C			I+C+F		
	K	Encoder	Decoder	K	Encoder	Decoder
NRI	4	710,660	406,020	8	711,688	678,404
fNRI	2+2	710,660	406,020	2+2+2	711,174	542,212
sfNRI	2	710,146	269,828	3	710,403	337,924

In our edge and trajectory prediction experiments, the following baselines are used:

- **Supervised**: The encoder is trained in isolation and the ground-truth interaction graphs are provided as labels. For the NRI and fNRI models we train using the cross-entropy error, and for the sfNRI model we use the binary cross-entropy error. All models are trained using a dropout of $p = 0.5$ on the hidden layer representation of every MLP to avoid overfitting, and the edge accuracy on the validation set is used for checkpointing.

- **True Graph**: The decoder is trained in isolation and the ground-truth interaction graphs are provided as inputs and we train using the reconstruction error ($M = 10$).

- **Static**: The decoder copies the previous state vector $\mathbf{x}^{t+1} = \mathbf{x}^t$ for M prediction steps.

8. Edge Accuracy

In order to calculate the edge accuracies, we have to work out the permutation of the edge-type labels the network uses. For the NRI model this is straightforward as the edge-types vectors are already one-hot. For each batch, we compute the edge accuracy for each label permutation. We expect the index permutation which gives us highest edge accuracy to correspond to the permutation the network uses. We

can confirm this to be true by looking at the frequency distribution of which label permutations give us this max accuracy over the whole dataset. If the network has settled on a label permutation, we observe all batches to give the max accuracy for the same label permutation. In the fNRI and sfNRI models where the edge-type vectors are no longer one-hot, this process is more slightly complicated as we also have to account for layer-graph label permutations.

In the results tables, edge accuracies are decomposed into the accuracy for each interaction type. The combined accuracy is calculated such that it only receives a contribution when the predicted edges between a pair of nodes are correct for all interaction types. This gives the combined accuracy a consistent meaning between the models.