Dynamic Factorised Neural Relational Inference

Samuel Pullely
ETH Zürich
pullelys@student.ethz.ch

Patrick Neuweiler
ETH Zürich
npatrick@student.ethz.ch

Leonie Muggenthaler
ETH Zürich
leoniem@student.ethz.ch

***Abstract*—Many multi-agent systems can best be understood by modeling the interactions of their components. The neural relational inference model (NRI) infers explicit interaction graphs and uses them to make predictions on future trajectories. Further work adapted the model to dynamically changing interaction graphs (dNRI) and improved the performance by factorising the interaction graph (fNRI). Starting from a previously published architecture, we investigate the effect of combining those two modifications (dfNRI). We evaluate the new model on human motion capture and show that dfNRI achieves prediction accuracy matching or slightly better than dNRI while compressing the model in some cases.**

I. Introduction

Multi-agent trajectory prediction is useful to understand various kinds of interacting systems, from purely physical settings to complex social dynamics. Characterizing interactions from observed trajectories in situations such as traffic scenarios, sports games or physical observations has a large range of possible applications. This task is difficult because usually, there are no labels provided for the interacting entities and a (possibly) large number of interaction types is needed to describe the dynamics. Moreover, the behavior patterns can be very complex and there is no guarantee that the interactions do not change over time.

The neural relational inference (NRI) model [1] addresses the challenge of multi-agent trajectory prediction by first retrieving an explicit interaction structure from observational data in an unsupervised fashion and then using the interaction graph to predict future trajectories. It takes the form of a variational auto-encoder, in which the encoder infers a discrete latent interaction graph from observed trajectories and the decoder predicts future dynamics based on the latent code of the encoder.

Being the first approach that relies on an explicit representation of the interactions, NRI laid the foundation for a line of research, addressing weaknesses of the model and adapting it to more challenging tasks. For example, it was proposed to additionally infer the hierarchies of the observed entities [2] (for example body parts in motion observations), to expand the model to heterogeneous agents including context information and to allow multi-modal predictions in ambiguous situations [3].

This work is based upon two of the proposed improvements, namely dNRI [4] and fNRI [5], which are now explained in more detail. In a system with multiple independent interactions, representing the interaction relationships as a single graph with many edge-types requires exponentially many types

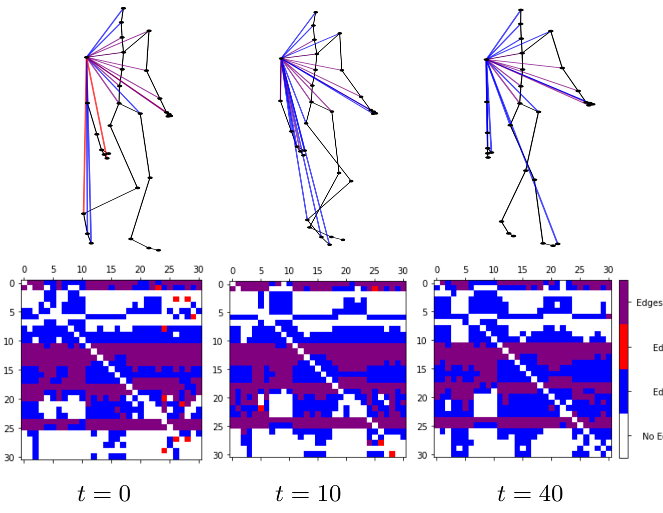


Fig. 1: Edges that send messages to the left shoulder joint and their type (color-coded) for the dfNRI $(2, 2)$ model applied to subject $\#35$, at predicted time steps $1, 10$ and 40. Red and blue correspond to different, independent edge types, while purple indicates presence of both edge types. No colored edge means no message gets send through the decoder GNN at that point in time (top). Adjacency matrix of inferred latent edge type graph (directed). The latent graph is hard coded to contain no loops, thus the diagonal never contains an edge (bottom).

to accommodate all possible combinations of interactions when using NRI. Therefore, factorised neural relational inference (fNRI) [10] factorises the latent interaction graph into a multiplex graph, where different interactions are represented using separate layer-graphs, greatly compressing the representation whilst also permitting better directed feedback and improved training.

Another weakness of NRI is that the inferred interaction graph is static. However, in many interacting systems the underlying interactions change over time. To address this concern, dynamic neural relational inference (dNRI) [2] recovers separate interaction graphs for every time step. In contrast to fNRI, the dNRI architecture still represents the interaction relationships in a single graph.

We implement a version of the dNRI model that allows for factorised latent interaction graphs as described in [10]. The new model, named dfNRI, is evaluated on a dataset also used in the dNRI paper [2]. By applying the same metric we isolate the possible advantage of a factorised latent interaction representation over the single interaction graph method.

II. Models and Methods

A. dNRI

We provide an overview of the dNRI model, highlighting major differences to the original NRI model and giving formal descriptions only for the parts that we modify. For a detailed explanation of the NRI model we refer to [1] and for a brief overview to [5].

Given N entities, let $\mathbf{x}_i^t$ be the feature vector of entity $i \in \{1, \ldots, N\}$ at time step t, for example position and velocity. The goal is to reconstruct the input trajectories $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$ by explicitly inferring a set of latent interactions $\mathbf{z}$ where $\mathbf{z}_{ij}$ is the discrete interaction type between entity i and j. During this process, the underlying dynamical model of the system is learned and can be used to forecast future trajectories. Both NRI and dNRI take the form of a variational autoencoder (VAE) [6] where the encoder and decoder are based on graph neural networks (GNNs) [7]–[9] containing one node per entity. Following a classical VAE, the evidence lower bound (ELBO) is maximised and is given as follows for dNRI:

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

where ϕ and θ are trainable parameters.

a) Prior: While the NRI prior assumes the relations to be static across the entire trajectory and is chosen based on the expected sparsity of the relations of the system, the dNRI prior is learned and predicts relations between entities at each time step t given the previously predicted relations and the trajectories up to time t. By computing a series of message passing operations, a GNN produces a K-dimensional edge embedding $\mathbf{h}^t_{(i,j),\text{emb}}$ for each pair of entities (i, j) that captures the state of the relations at time t. These embeddings are fed into an LSTM that models the evolution of the relations over time and an MLP finally transforms the hidden state into the logits of the prior distribution:

$$\mathbf{h}^t_{(i,j),\text{prior}} = \text{LSTM}_{\text{prior}}\left(\mathbf{h}^t_{(i,j),\text{emb}}, \mathbf{h}^{t-1}_{(i,j),\text{prior}}\right) \quad (2)$$

$$p_\phi\left(\mathbf{z}^t \mid \mathbf{x}^{1:t}, \mathbf{z}^{1:t-1}\right) = \operatorname{softmax}\left(f_{\text{prior}}\left(\mathbf{h}^t_{(i,j),\text{ prior}}\right)\right) \quad (3)$$

b) Encoder: In NRI, the encoder's role is limited to predicting a single edge configuration covering an entire set of input trajectories. In contrast, the dNRI encoder is tasked with approximating the distribution of relations at each time step as a function of the entire input. A key component is an LSTM that processes the relation embeddings $\mathbf{h}^t_{(i,j),\text{emb}}$ in reverse. The approximate posterior is obtained by concatenating the reverse state and the forward state provided by the LSTMs and passing the result into an MLP:

$$\mathbf{h}^t_{(i,j),\text{enc}} = \text{LSTM}_{\text{enc}}\left(\mathbf{h}^t_{(i,j),\text{emb}}, \mathbf{h}^{t+1}_{(i,j),\text{enc}}\right) \quad (4)$$

$$q_\phi\left(\mathbf{z}^t_{(i,j)} \mid \mathbf{x}\right) = \operatorname{softmax}\left(f_{\text{enc}}\left(\left[\mathbf{h}^t_{(i,j),\text{enc}}, \mathbf{h}^t_{(i,j),\text{ prior}}\right]\right)\right) \quad (5)$$

The edge-type vectors $\mathbf{z}^t_{(i,j)}$ are sampled from the concrete distribution [10], [11]. It is a continuous approximation of the discrete categorical distribution and the reparametrization trick is used such that the sampling operation is differentiable and backpropagation is possible. Note that prior and encoder share some parameters.

c) Decoder: The task of the decoder is to predict the dynamics of the system given the sampled latent interaction graph $\mathbf{z}^t$. The decoder factorises as follows:

$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \prod_{t=1}^{T-1} p_\theta\left(\mathbf{x}^{t+1} \mid \mathbf{x}^{1:t}, \mathbf{z}^{1:t}\right) \quad (6)$$

The primary difference to NRI is that the the relation variable inputs $\mathbf{z}$ vary per time step. The latent graph encoding $\mathbf{z}$ is used as weight in the first step of the decoder message passing section:

$$v \rightarrow e : \tilde{\mathbf{h}}^t_{(i,j)} = \sum_{k=1}^{K} z^t_{ij,k} \tilde{f}^k_e\left(\left[\tilde{\mathbf{h}}^t_i, \tilde{\mathbf{h}}^t_j\right]\right), \quad (7)$$

where $z^t_{ij,k}$ denotes the k-th element of $\mathbf{z}^t_{ij}$. If $\mathbf{z}^t_{ij}$ is a one-hot sample, the edge model $\tilde{f}^k_e$ is used. When samples are taken from the continuous approximation, $\tilde{\mathbf{h}}^t_{(i,j)}$ becomes a weighted sum. If desired, the first edge-type can be hard-coded as non-edge, in which case the first summand in eq. (7) is skipped and the number of parameters of the decoder is decreased. A second message passing operation $e \rightarrow v$ produces a vector $\boldsymbol{\mu}^{t+1}_j$ from which the prediction for the next time step $\mathbf{x}^{t+1}_j$ of particle j is sampled as $p_\theta\left(\mathbf{x}^{t+1}_j \mid \mathbf{x}^{1:t}, \mathbf{z}^{1:t}\right) = \mathcal{N}\left(\boldsymbol{\mu}^{t+1}_j, \sigma^2\mathbf{I}\right)$.

d) Training/Inference: In order to train the parameters ϕ and θ of the encoder/prior and decoder, the input trajectories $\mathbf{x}$ are passed through the GNN model to produce relation embeddings $\mathbf{h}^t_{(i,j),\text{emb}}$ for every time t and every entity pair (i, j). These embeddings are input into the forward/backward LSTMs, the prior $p_\phi(\mathbf{z}|\mathbf{x})$ and encoder $p_\phi(\mathbf{z}|\mathbf{x})$ are computed and the relations $\mathbf{z}$ are sampled from the encoder. Given these relations, the decoder then predicts the trajectory distribution $p_\theta(\mathbf{x}|\mathbf{z})$.

At test time, we cannot use the encoder to predict edges since we cannot feed the backward LSTM with any future unobserved states. Therefore, we sample the relations $\mathbf{z}^t$ from the prior.

B. Dynamic Factorised NRI (dfNRI)

As mentioned in the introduction, we combine dNRI and fNRI. Both NRI and dNRI use a one-hot latent encoding for the edge types $\mathbf{z}_{ij}$. This means that for each combination of multiple independent interaction types, for example spring and charge interactions in a physical setting, there needs to be a single edge type (see figure 3). fNRI in contrast uses a multi-categorical encoding, which allows the encoder

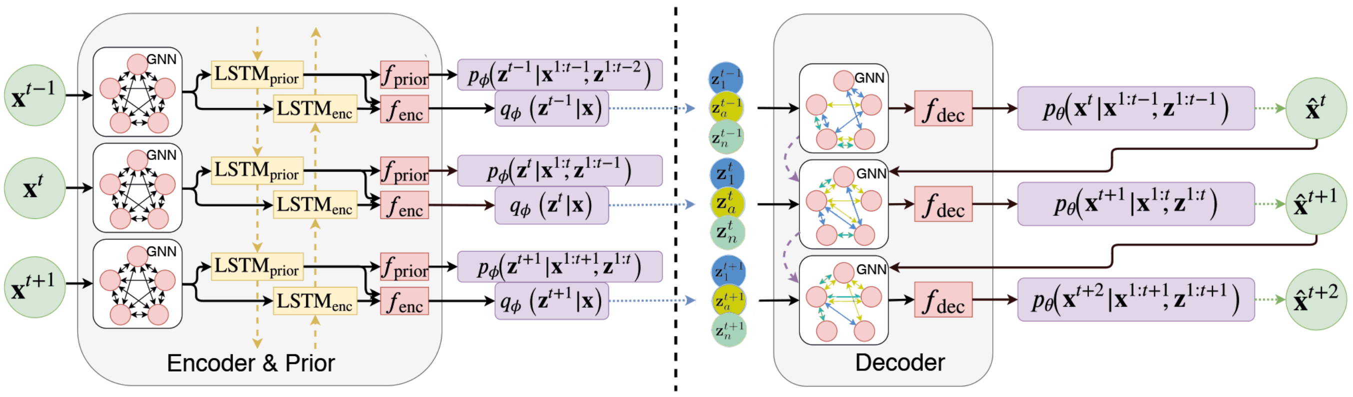


Fig. 2: The dfNRI model. The inputs x are fed to a GNN to produce an embedding of the relations for every pair of entities at every point in time. A forward LSTM computes a representation of the past which is used for the prior, and both the past and the future are used to compute the encoding (approximate posterior). A set of edge variables is sampled *for each layer graph* (factorization). The edge variables are used to choose a set of edge types for each time step. The decoder predicts the next time step using the multiplex GNN and the previous predictions.

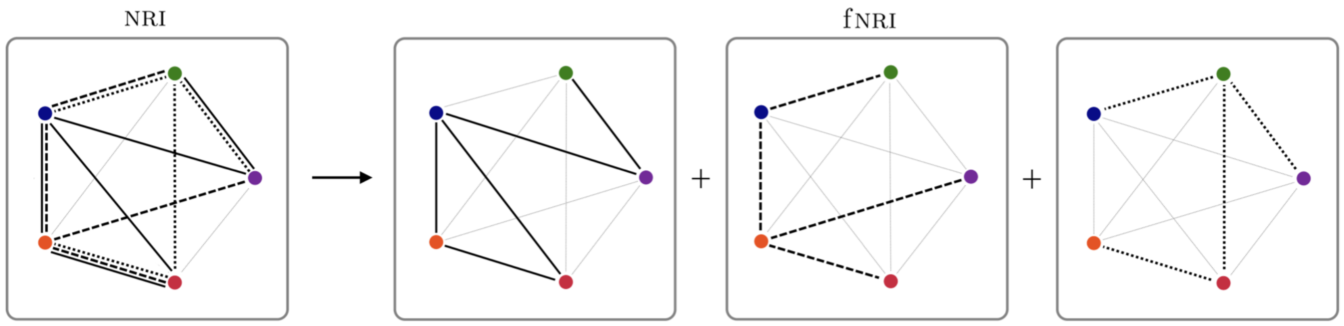


Fig. 3: Visualization of the factorised interaction graph proposed by fNRI. While NRI requires a different edge type for all possible combinations of interactions, fNRI requires only two edge types for each of the layer graphs.

to choose one edge type per layer-graph. The compression benefit grows exponentially with the number of independent interaction types. Moreover, the edge type decoder networks $\tilde{f}_e^k$ in eq. (7) now have a larger training set since $\tilde{f}_e^k$ gets used every time its corresponding interaction is present. In addition, the factorised decoder model has fewer parameters. We expect this representational change to have benefits even beyond purely physical systems that were considered in [5] and where the underlying independent interactions and their combinations are clearly observable.

The necessary adaptions in order to incorporate fNRI into dNRI and get dfNRI are described in the following. Formally, the single interaction graph with K edge types is factorised into an n-layer multiplex graph, where the a-th layer-graph has K_a edge types. For dfNRI, the encoder logits in eq. (5) are split into n segments:

$$\boldsymbol{\ell}^t_{(i,j),\text{enc}} := f_{\text{enc}}([\mathbf{h}^t_{(i,j),\text{enc}}, \mathbf{h}^t_{(i,j),\text{prior}}]) \tag{8}$$

$$\boldsymbol{\ell}^t_{(i,j),\text{enc}} = [\boldsymbol{\ell}^{t,1}_{(i,j),\text{enc}}, \dots, \boldsymbol{\ell}^{t,n}_{(i,j),\text{enc}}] \tag{9}$$

where each segment $\boldsymbol{\ell}^{t,a}_{(i,j),\text{enc}}$ is a K_a-dimensional vector and $K = \sum_{a=1}^n K_a$ is the total number of edge types. The edge type posterior distribution for each layer-graph a is then given by

$$q_\phi(\mathbf{z}^{t,a}_{(i,j)} \mid \mathbf{x}) = \text{softmax}\left(\boldsymbol{\ell}^{t,a}_{(i,j),\text{enc}}\right) \tag{10}$$

An analagous change is applied to the prior logits in eq. (3)

$$\boldsymbol{\ell}^t_{(i,j),\text{prior}} := f_{\text{prior}}\left(\mathbf{h}^t_{(i,j),\text{enc}}\right) \tag{11}$$

An alternative way to incorporate fNRI into dNRI would be to do the segmentation earlier in the network, namely for the edge embedding $\mathbf{h}^t_{(i,j),\text{emb}}$ that is produced by the prior/encoder GNN. We chose not do so because that requires separate forward/backward LSTMs for each segment and the hidden states would only contain information about the segment itself and not about the other segments.

The edge type vectors $\mathbf{z}^{t,a}_{ij}$ are sampled as in dNRI and then concatenated to get the edge type vector $\mathbf{z}^t_{ij} = [\mathbf{z}^{t,1}_{ij}, \dots, \mathbf{z}^{t,n}_{ij}]$ of the multiplex interaction graph. The $\mathbf{z}^t_{ij}$ are now multi-categoric and supplied to the decoder.

The KL divergence is adapted as follows

$$D_{\mathrm{KL}}\left[q_\theta(\mathbf{z} \mid \mathbf{x}) \| p_\phi(\mathbf{z} \mid \mathbf{x})\right] = \sum_{t=1}^{T} \sum_{a=1}^{n} D_{\mathrm{KL}}\left[q_\theta\left(\mathbf{z}^{t,a} \mid \mathbf{x}\right) \| p_\phi\left(\mathbf{z}^{t,a} \mid \mathbf{x}\right)\right] \quad (12)$$

The training and inference routine is not modified and stays the same as in dNRI.

C. Implementation Details

In order to isolate the impact of the factorised interaction graph used in dfNRI over the single interaction graph used in dNRI, we use the exact same hyperparameters as detailed in [4] unless otherwise stated.

The only hyperparameter we change is the number of edge types. We always use at least four edge types because only then there is a difference between our dfNRI model and the dNRI model. As was done in [5], we always use two edge types per layer-graph, i.e. $K_a = 2$. In dNRI and fNRI, all experiments were done with the first edge type hard-coded as non-edge. In the case of fNRI, this means that the first edge type in each layer-graph is hard-coded as non-edge. Using this approach, the first edge type decoder network is skipped in the message passing operation. In our experiments however, we observed that sometimes better results are achieved if we do not hard-code the first edge type to be non-edge, whenever that is the case we denote the model with an asterisk*, for example dfNRI $(2,2)^*$.

III. Results

We provide the results on experiments with two subjects, #35 for walking and #118 for jumping, from the CMU motion capture database[1] [12].

Our evaluation procedure is the same as [1] and [4]: For subject #35, we train using the first 50 steps of the sequence and evaluate the mean squared error (MSE) in predicting following 49 steps of the sequence. For #118, due to the ”unforeseen” jump of the subject, the model is always restored to ground truth after making a prediction for the next 40 timesteps. The errors are then averaged, which is why the MSE curve is smoother compared to subject #35. The results are shown in figure 5a and 5b. The original dNRI model was evaluated only for up to 4 edge types, however, we scaled up the number to 16 edge types to have a comparison to a 4-layer dfNRI model (i.e. dfNRI $(2,2,2,2)$). For subject #35 the best performance was achieved with the dNRI model with 16 interaction types (dNRI 16), closely followed by dfNRI $(2,2,2)^*$. However, the standard deviation for dfNRI $(2,2,2)^*$ is high and the result for one single seed achieved in fact the by far lowest MSE. For subject #118, the best dfNRI model only reached an MSE similar to the best dNRI model. Again, a high standard deviation suggests that with some modifications better results might be achieved for dfNRI.

[1]Due to time constraints, we did not perform experiments on the basketball dataset used in [4].

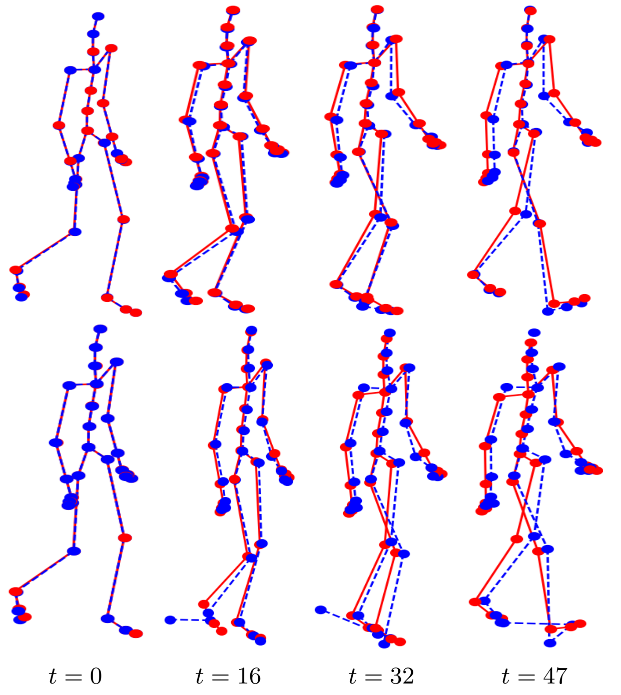


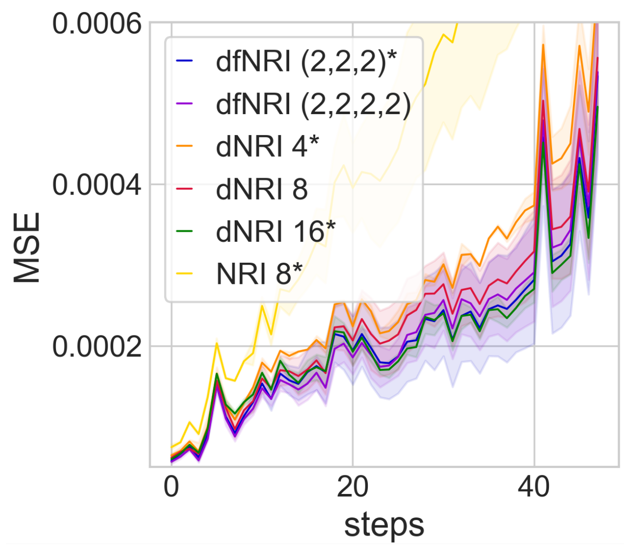
Fig. 4: The overall best variant of both model types dfNRI and dNRI, dfNRI $(2,2,2)^*$ (top) and dNRI (4) (bottom), used to predict 48 steps of future movement of sample 20 of subject #35. Ground truth is displayed in solid-red, while the prediction is dashed-blue.

IV. Discussion

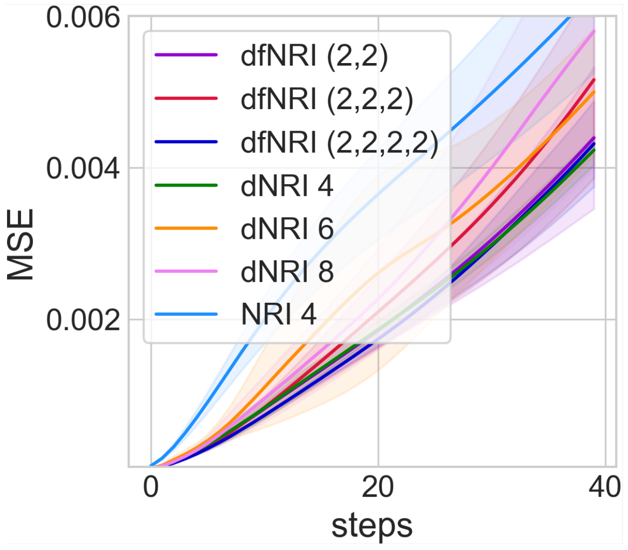
The executed experiments are not yet sufficient to determine one single best model-type, as different models outperform others on different datasets. Overall, factorising the latent graph has slightly improved prediction errors in most cases, thus we argue that the combined dfNRI model does indeed provide a small expressive advantage over the dNRI model.

The optimal number of possible latent edge types (dNRI), respectively the optimal number of subgraphs (dfNRI) is also not perfectly determined through our experiments. Although the lowest prediction error for dNRI was achieved with the maximal number of interaction types in our experiments, this does not reflect a trend towards better performance with increasing number of edge types. The same is true for dfNRI: A higher number of layer graphs did not necessarily improve the prediction, although we were indeed able to achieve a comparably low error with some of the models. The improvement due to factorising the latent graph is however small, which raises questions about the benefits of the dfNRI model. While one might not gain a lot of expressive power with the factorised neural network, we can definitely say, that while keeping at least the same level of expressive power, one has a lot less parameters to train. This in turn allows to use larger MLP-networks connecting the edges or for more different latent edge types.

While for the dNRI model the optimal number of edge types



(a) Subject #35



(b) Subject #118

Fig. 5: Trajectory prediction mean squared errors (MSE) on motion capture data. Results averaged across 2 - 5 initializations, shaded area representing standard deviation.

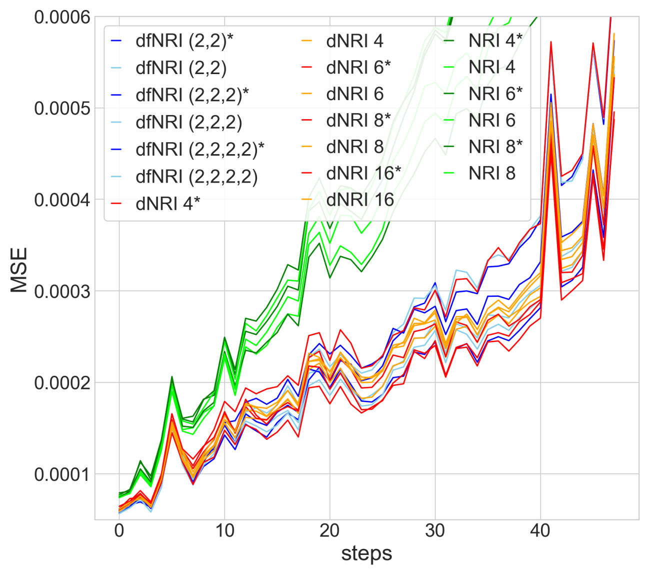


Fig. 6: Mean squared errors without standard deviations for all experiments run for subject #35.

seems to saturate very quickly (most of the time, 4 edge types outperformed the other dNRI models, as also the authors of [4] report), the optimal number of edge types for dfNRI is not that clear. Especially, dfNRI could improve by adding more supgraphs, and thus be able to handle more complex situations. This would require further investigation.

V. Summary

In this paper we show that it is possible to combine the contributions of both the dNRI and the fNRI paper and that they are well compatible. While factorising the latent graph of dNRI brought less improvement on the prediction errors than the original fNRI paper that factorised the static NRI approach, we were still able to get small improvements for most cases and were able to reduce the number of parameters that had to be trained.

Overall we argue that factorising the latent representation graph into multiple subgraphs with each a small number of different possible edge types is favourable, mainly for the number of parameters to train, but also for slight improvements of the predictions.

Acknowledgement

We would like to thank the authors of [1], [4] and [5] for making the codebase for their models publicly available. We would also like to thank the ETH Computer Science Department for granting us access to the Leonhard Cluster to run our experiments.

References

[1] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," 2018.

[2] A. Stanić, S. van Steenkiste, and J. Schmidhuber, "Hierarchical relational inference," 2020.

[3] J. Li, F. Yang, M. Tomizuka, and C. Choi, "Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning," in *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2020.

[4] C. Graber and A. G. Schwing, "Dynamic neural relational inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] E. Webb, B. Day, H. Andres-Terre, and P. Lió, "Factorised neural relational inference for multi-interaction systems," 2019.

[6] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[7] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[8] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017.

[10] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *CoRR*, vol. abs/1611.00712, 2016. [Online]. Available: http://arxiv.org/abs/1611.00712

[11] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017.

[12] C. G. Lab, "Cmu motion capture database," 2003, http://mocap.cs.cmu.edu.