Learning the graph structure of interaction networks for inference of complex object-centric and relational dynamics

Swapnil Pande, Saumya Saxena, Alvin Shek, Kevin Wang
{swapnilp, saumyas, ashek, kwang2}@andrew.cmu.edu

1 Introduction

Learned dynamics models have been extensively applied in planning and control for a wide variety of robotics tasks. These models are useful because they allow us to model environment state transition dynamics simply from a dataset of interactions, instead of manually programming a first-principles simulator, which can be very expensive and may not accurately capture the true environment dynamics. Recently, there has been growing interest in parametrizing dynamics models as graphical models, as this class of models naturally capture the dependencies in the behavior of complex mechanical systems. The inductive biases that these graphical models provide have the potential to reduce model size and improve generalization performance over the standard multi-layer perceptron. However, most implementations have assumed that the graphical structure of the mechanical system is known a priori. While this may be a safe assumption for very simple systems, this assumption does not hold for complex systems with many links and actuators. Furthermore, it is possible that even for simple systems, a structure that captures all of the relationships may not be evident a priori. For example, for a legged robot, we could naïvely describe links as nodes, and add edges for every joint. However, we may find that due to the nature of the gaits that we could also add edges between opposite legs, due to the dependencies imposed by the gait. Therefore, we aim to address the challenge of learning the graph structure of a given dynamical system for the inference of the system's dynamics.

We propose a new model that combines ideas from two different approaches: we combine the graph learning architecture from [1] to learn the underlying graphical model of a system and the "Encode-Process-Decode" architecture presented in [2] as our forward model.

2 Dataset and Task

We consider the problem of learning the structure of a graphical model underlying the graph network architecture presented in [2]. [2] uses *predefined* graphs to represent the dynamics of complex, dynamical systems such that nodes represent objects and edges represent relations between pairs of objects. This network architecture is then used for inference of object-centric and relational dynamic parameters from random excitation trajectories. In this work, our task is to learn the underlying graph structure governing a dynamical system given these trajectories. In doing so, we capture relationships and find interactions between the joints of an agent. Specifically, given a dynamical system defined as a set of nodes $N = \{n_i\}_{i=1}^{N}$, we seek to learn the edge set E for a graph $G = (N, E)$ that minimizes the prediction error of the dynamics function $s_{t+1} = f_d(s_t)$, where f_d represents the forward dynamics prediction function.

We will be testing this approach on two MuJoCo simulation agents, Half Cheetah and Walker2D. Both of these systems consists of multiple links connected at joints, so they are amenable to a graphical representation. We collect the trajectories of a node/joint state and action for each MuJoCo agent over multiple timesteps as our input. We will evaluate the performance of our method using the 1-step and 5-step rollout performance. For each expriment, we quantitatively measure our results using the mean squared error between the true states and predicted states. Furthermore, for the graph learning methods, we visualize our generated graphs to see the links between joints in a MuJoCo agent to qualitatively measure the predicted graph structures.

3 Methods

3.1 Proposed method

Our method is briefly summarized in Fig. 1. Our learning architecture is composed of two main modules: the graph inference module and the dynamics module. We use Causal Discovery Network (CDN) [1] as our graph inference module that takes as input observed trajectories and outputs the graph structure. Specifically, it outputs the edge set and the corresponding edge parameters. Our dynamics module, Encode-Process-Decode (EPD) [2], then takes as input this graph structure and the current state and predicts the state at the next time step.

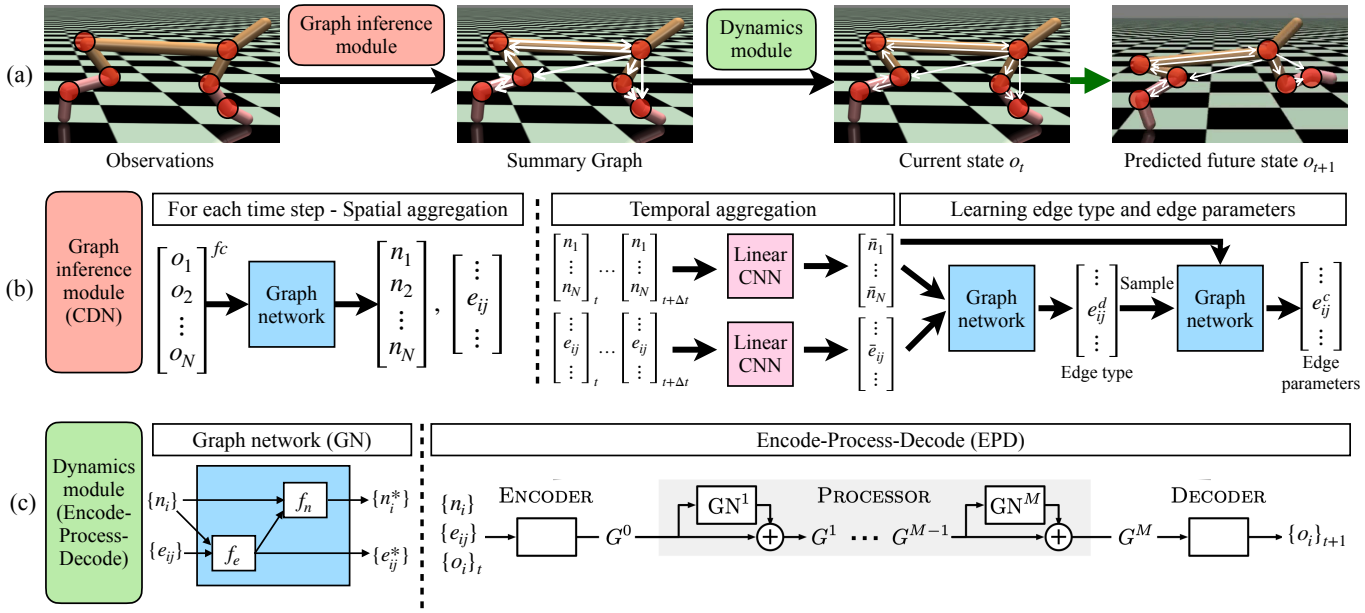

Figure 1: (a) Learning architecture: Using observed trajectories the causal summary graph is inferred using the Inference module. The dynamics of the system is simultaneously learned using the Dynamics module that operates on the identified graph. (b) Inference module discovers the edge set and the associated parameters. First, we propagate information spatially at each time step using a graph network which inputs a fully connected graph over all nodes. The output node and edge embeddings are then aggregated temporally and fed into a 1D-CNN to output temporal aggregations which are again be passed though another graph network that predicts a discrete distribution over edge types. Sampling from this distribution using the Gumbel-Softmax technique and passing through another graph network then gives the continuous edge parameters. The edge types and edge parameters together constitute the causal graph. (c) Dynamics module: The left half of this subplot shows the graph network building block which acts as a single message passing step. The encoder encodes the node and edge parameters and observations and feeds them to the processor which acts like a sequence of message passing steps with skip connections. The output is then passed through the decoder which predicts the next state.

In particular, CDN first aggregates the state information spatially at every time step by feeding it to a graph network (GN) (more details about GNs are provided in section 3.2) assuming a fully connected graph. The output node and edge emdeddings for each time step are then aggregated temporally by feeding these embeddings over a small time window to a linear CNN. These temporally aggregated embeddings are then fed to another graph network that outputs a categorical distribution over the edge types where edge type can represent whether an edge is present in the graph or not. We sample from this categorical distribution using the Gumbel-Softmax sampling technique (to ensure differentiability) and feed it to another graph network that outputs the associated edge parameters.

The inferred edge types and parameters together constitute the graph structure which forms the input to our dynamics module. EPD first encodes the state-based graph representation as a latent graph G^0. This latent graph is then processed via M message passing steps, by passing it though a sequence of multiple graph networks, to generate an output latent graph G^M. The decoder then finally converts the latent graph back to a state-based graph representation which is the predicted next state. The whole model is trained end-to-end using the following objective function:

$$\mathcal{L} = \mathbb{E}_{p_\phi(\tilde{E}|s_{1:T})}\left[logp_\psi(\hat{s}_{t+1}|s_t, \tilde{E})\right] - \mathcal{D}_{KL}\left(p_\phi(\tilde{E}|s_{1:T})\|p(\tilde{E})\right)$$

where p_ϕ is the edge prediction model, p_ψ is the dynamics model, and $p(\tilde{E})$ is a prior that encourages a sparse graph.

3.2 Baselines

Our baselines are designed to explore the two main components of our approach: the graph inference module and the dynamics module. Our first three baselines assume that we know the graph structure a priori, considering an edge at every physical link in the system, and thus the inference module is not required. For these baselines we explore three types of dynamics modules 1) MLP 2) Graph network 3) Encode-Process-Decode.

For baseline 1 we consider a multi-layer perceptron with 2 hidden layers as the forward model. For baseline 2, a graph network module [3] is considered, which represents one iteration

	Inference module	Dynamics module
Baseline 1	Graph known	MLP
Baseline 2	Graph known	Graph Network
Baseline 3	Graph known	Encode-Process-Decode
Baseline 4	Causal Discovery Network	Graph Network
Proposed method	Causal Discovery Network	Encode-Process-Decode

Figure 2: Our baselines explore two main components of our proposed method: The graph inference module and the dynamics module

of message passing similar to belief propagation Fig. 1 (c, left).
Given an graph with node and edge embeddings $\{n_i\}$ and $\{e_{ij}\}$, the GN module first updates each edge using its neighboring nodes, $e^*_{ij} = f_e(n_i, n_j, e_{ij})$ where e^*_j is the updated edge embedding and f_e is a MLP. Once each edge is updated, each node is also updated using the aggregated messages from all its incoming edges, $n^*_i = f_n(n_i, \sum_j e_{ij})$ where n^*_i is the updated node embedding and f_n is a MLP.

In baseline 4, we assume that the model is not known a priori and use the CDN architecture to learn the graph structure (details provided in section 3.1) and use a single graph network (GN), which acts like a single message passing step, as the forward model in the dynamics module.

4 Related Work

Our approach builds upon the interaction networks architecture explored in [2], [3], [4] and [5]. In our work we first build upon the method presented in [2] because of its capability of making accurate long horizon predictions for very large environments eg. Water-3D and Sand-3D. All of these approaches assume an existing graph of nodes and edges and attempt to learn the features representing the nodes and edges. Here, the key joints and limbs of a model or object are labeled as nodes, while the edges between these nodes are added based on intuition. However, a priori knowledge of the graph is not always available or necessarily optimal for certain tasks. In this case, either the nodes, edges, or both may be generated by a learned model, which is the focus of Deep Graph Generation (DGG). Given a dataset of graphs from an underlying distribution $p(\mathcal{G})$, these methods attempt to either explicitly model this distribution, or learn to sample from this distribution [6]. [7] [8] sequentially generate new nodes or edges using LSTMs so that new generations depend on previous ones along with the existing graph. [9] [10] take existing, locally-valid substructures of nodes and edges, and builds a graph from these subgraphs, while other approaches generate the entire graph of nodes and edges using Variational AutoEncoders (VAE) [11] [12]. In [13] and [14], edges are selected for a given set of nodes in a scene graph using a Relation Proposal Network, which is optimized using the ground truth edges.

We also take inspiration from the method proposed in [15] as this method clearly relates to our task of learning object-centric and relational dynamics parameters while using interaction networks as the base architecture. This method proposes a Visual Causal Discovery Network (V-CDN), that takes as input visual representations of a scene at various timesteps, converts them into relevant keypoints, and uses these keypoints to learn the graph architecture as well as the forward model. In our task, keypoints/nodes are already known and do not need to be inferred from images. We plan to adapt this graph learning method to the graph architecture presented in [2] and exploit the strength of each these methods in terms of learning a more accurate graph representation for our data as well as ensuring accurate long-horizon predictions.

5 Experiments

To evaluate the performance of our model, we train the baselines and our model to make forwards dynamics predictions for two standard Mujoco simulator environments: Half-Cheetah and Walker2D. Our models are each optimized on 1-step predictions from a dataset containing a mix of data generated by a random policy and expert policy. We additionally evaluate the performance of our model for 5-step rollouts, which are performed by inputting the predicted state as the input at the next time step. The results from these experiments are presented in Figure 3. Finally, we perform a qualitative analysis on the graph generated by the model to gain an intuitive understanding of the learned relationships. We observe from Figure 3:a that the performance for both graph architectures for dynamics predictions exceeds the performance of the MLP, indicating that the inductive biases that the graph architecture imposes on the model are effective. However, we unexpectedly find that the GN model performs better than the EPD model.

Comparing plots 3:c, 3:d, 3:e, and 3:f, we notice three key trends. First, CDN-GN's mean squared error is much lower than the error of our proposed model. We believe this is true for several reasons explained at the end. Second, CDN-GN in fact performs better than the other baselines on Walker 2D, showing that a learned graph can yield better performance than a fixed graph. Lastly, comparing plots 3:e and 3:f (N-step prediction) with 3:c and 3:d (1-step prediction), we also observe that N-step error is higher. In N-step prediction, the model feeds its own output back as input N times, and initial errors propagate and will build up over time.

Figure 4 visualizes the learned edges (directed arrows) and a priori known nodes (circles) annotated on top of the simulation models. Figure 4:a shows the output of our CDN-EPD model, which resembles an almost fully-connected graph. This shows that the model was not able to identify the most meaningful edges, explaining the poor prediction performance. However, figure 4:b shows the output of CDN-GN, which is more sparse with dense connections only between nodes on the same leg. This matches our intuition since joints that are physically connected or spatially nearby should have edges. Only a few edges are learned across the two legs, and this possibly represents the relationship between the two legs in a gallop or walking motion since the legs follow a cyclic motion pattern. This shows the promise of learning a graph to discover useful relationships rather than assuming fixed priors. Figures 4:c and 4:d show similar trends.

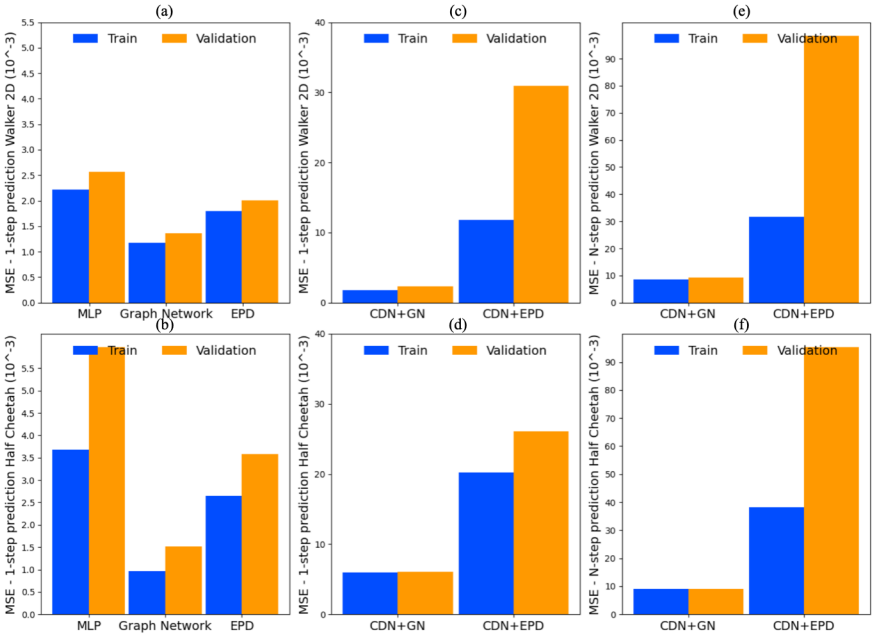

Figure 3: (a), (b) Baselines 1,2,3 for the walker2D and half cheetah models: Assuming a known graph, comparison of performance using 1-step dynamic prediction as metric using three dynamics modules - MLP, Graph Network and Encode-Process-Decode (c), (d) Baseline 4 (CDN+GN) vs Proposed method (CDN+EPD): Comparison of performance using 1-step dynamic prediction as metric (e), (f) Baseline 4 (CDN+GN) vs Proposed method (CDN+EPD): Comparison of performance using 5-step dynamic prediction as metric

In all experiments, we find that the performance of the EPD dynamics model is worse than that of the GN model. We hypothesize that this is a result of the additional hyperparameters that the EPD model introduced, which may not be tuned correctly. Specifically, we believe that varying the number of message passing steps EPD performs (currently performing 10), may have a significant influence on performance, as GN performs better with a single step. Additionally, the added complexity of the model may be causing the joint optimization between the graph inference network and the dynamics prediction to settle into a local minimum, resulting in the mostly fully-connected graphs we see in Figures 4:c,d. A potential fix for avoiding local optima could be pretraining the dynamics model based on a heuristic a priori graph.

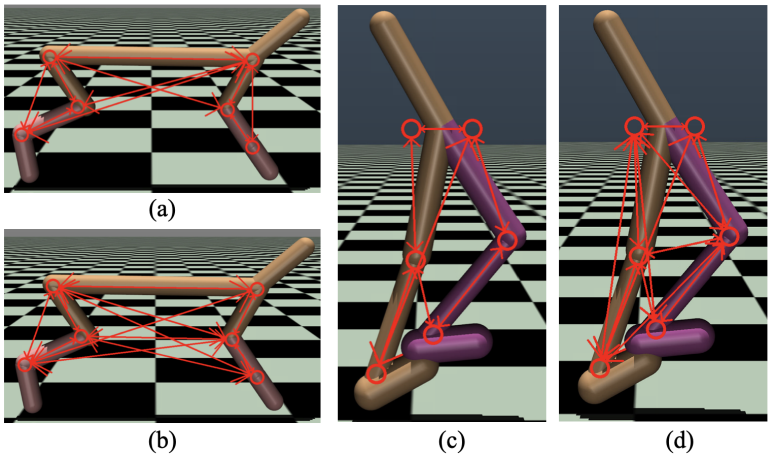

Figure 4: (a), (c) Learned graph structure for the walker2D and half cheetah models using CDN+GN learning architecture. (b), (d) Learned graph structure for the walker2D and half cheetah models using CDN+EPD learning architecture.

6 Conclusion

We propose a method for learning a graphical structure to build a dynamics model for mechanical systems with multiple links. We compare the performance of multiple models, and find that the CDN model combined with the Graph Network dynamics model learns effective graph representations for the two systems and demonstrates low prediction error. We also find that the EPD model performs worse than the GN model, both for a learned graph and an a priori graph. In the future, we hope to study the effect of the number of message passing steps performed in the EPD model on the model's performance. To remediate the issue of local minima in the CDN+EPD model, we also plan to try pretraining the dynamics model on a heuristic graph.

References

[1] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos, 2020.

[2] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[3] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[4] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics, 2016.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, and Hamid R. Rabiee. Deep graph generators: A survey, 2020.

[7] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model, 2018.

[8] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models, 2018.

[9] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation, 2019.

[10] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs, 2020.

[11] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders, 2018.

[12] Daniel Flam-Shepherd, Tony Wu, and Alan Aspuru-Guzik. Graph deconvolutional generation, 2020.

[13] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation, 2018.

[14] J. Zhang, M. Elhoseiny, S. Cohen, W. Chang, and A. Elgammal. Relationship proposal networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5226–5234, 2017.

[15] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos supplementary material.