Project Proposal: “Generalization Performance of Graph Neural Networks for Reinforcement Learning in Robotics”

Oliver Borchert Vitalii Rusinov

Advanced Deep Learning for Robotics, Summer 2020

Advisor: Johannes Pitz

1 Objective

The aim of the project is exploring the use of Graph Neural Networks for reinforcement learning tasks. The robot can be modeled as a connected graph, and various ways to extract the features from this graph and use them together with MLP features could be explored. We believe that it can potentially help the agents train faster and better generalize to new robot structures. When robots are trained in simulated environments, the trained agents very often have problems generalizing to the real environment, and training the agents with real robots is expensive. We hope that Graph neural networks would also help bridge this gap making better use of the experience collected in the simulated environment.

2 Related Work

Notable previous work on the generalization performance of RL algorithms includes Packer et al. (2018). They presented a benchmark protocol evaluating the generalization performance of a variety of algorithms when augmenting the training environment. They found that, in most cases, “vanilla” RL algorithms outperform specialized approaches [9]. This finding allows us to focus our analysis of the effect of graph neural networks on well-known algorithms such as PPO [10], TD3 [4], and SAC [5]. While often, the usage of GNNs in the context of RL is focused on multi-agent environments [6, 8], there also exists prior work on representing an environment’s state in a relational fashion [13]. Interestingly, Wang et al. already considered modeling robot morphology as a graph but we believe that in our project, we must explore more neural network types, feature embeddings and look more closely into the generalization performance of the resulting algorithms.

3 Proposed approach

In our project, a robot is to be represented by an attributed graph where the nodes are given as the robot’s joints and the edges represent the connections between them. Naturally, nodes are attributed with their coordinates while edges may carry geometric information such as the length of the connection or its angle. As a result, we can model the relational structure that is otherwise lost or obscured, at a minimum, when representing a robot’s state as a fixed-size vector. Applying different kinds of graph convolutions, we expect to model the robot more accurately. It is proposed to compare the results when we use different edge features, but also try using graph Laplacian or deep walk embeddings with node features. We would also like to explore different ways to combine GNN and MLP features. We propose using the TD3 and SAC algorithms in the beginning. In addition, we would like to see how the weights of the GNN can be used in a different environment and whether we need to make them trainable. Applying GNNs to imitation learning is another possible approach.

For our experiments, we would use the PyBullet environments [1] to model the robot and its environment and either the Spinning Up or stable-baseline3 framework. Furthermore, we would leverage PyTorch Geometric [3] to efficiently perform deep learning on graphs.

References

[1] Coumans, Erwin and Bai, Yunfei. *PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning*. `http://pybullet.org`. 2016–2019.

[2] Fey, Matthias, Eric Lenssen, Jan, Weichert, Frank, and Müller, Heinrich. "SplineCNN: Fast geometric deep learning with continuous B-spline kernels". In: *IEEE CVPR*. 2018, pp. 869–877.

[3] Fey, Matthias and Lenssen, Jan E. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop*. 2019.

[4] Fujimoto, Scott, Van Hoof, Herke, and Meger, David. "Addressing Function Approximation Error in Actor-Critic Methods". In: *arXiv preprint arXiv:1802.09477* (2018).

[5] Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *arXiv preprint arXiv:1801.01290* (2018).

[6] Jiang, Jiechuan, Dun, Chen, and Lu, Zongqing. "Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation". In: *arXiv preprint arXiv:1810.09202* (2018).

[7] Kipf, Thomas N and Welling, Max. "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[8] Liu, Yong et al. "Multi-Agent Game Abstraction via Graph Attention Neural Network". In: *arXiv preprint arXiv:1911.10715* (2019).

[9] Packer, Charles et al. "Assessing Generalization in Deep Reinforcement Learning". In: *arXiv preprint arXiv:1810.12282* (2018).

[10] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[11] Wang, Tingwu, Liao, Renjie, Ba, Jimmy, and Fidler, Sanja. "NerveNet: Learning Structured Policy with Graph Neural Networks". In: *ICLR*. 2018.

[12] Ying, Zhitao et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *NeurIPS*. 2018, pp. 4800–4810.

[13] Zambaldi, Vinicius et al. "Relational Deep Reinforcement Learning". In: *arXiv preprint arXiv:1806.01830* (2018).

Milestones

Week 1–3: Modeling Robots as a Graph

- We want to modify existing PyBullet [1] examples such that a robot can be described by a graph where the nodes model its joints and the edges the connections between them. It would be expected that such a description carries more information than a simple state vector describing joints and their velocities.
- We want to enhance the simple MLP that learns the policy given the observation vector by adding a graph neural network [7] that operates on the graph describing the robot. For the implementation of the GNN, we want to use PyTorch Geometric [3] and use different message passing layers such as SplineConv [2].
- We want to use well-known RL algorithms [10, 4, 5] as given by *Spinning Up in Deep RL*[1] to train the agent.

Week 4–6: Generalization to New Environments

- Building upon our work in the first weeks, we want to evaluate how well the robot's policy works in an environment that is slightly altered.
- Primarily, we will be focusing on how different representations of the robot as a graph influence generalization performance. For this, we consider different node features (e.g. one-hot vectors, features learnt by DeepWalk, or features obtained by the first eigenvectors of the Graph Laplacian) as well as edge features (e.g. length of connection, i.e. Euclidean distance between joints, angle, …). As far as node features are concerned, it is particularly interesting how well training works with features that were obtained solely from the graph's structure.
- For this, we will alter the environments provided by PyBullet.

Week 7–10: Transfer Learning

- Eventually, we want to investigate whether graph neural networks (GNNs) enable transfer learning. As GNNs are well-known to work on graphs of different sizes and there is the possibility to do hierarchical learning on the graph [12], it is interesting to see whether a trained GNN helps training faster for a completely different environment/agent.
- Weights of the GNN can either be fixed, trained with a slow learning rate, or just constitute a good starting point for training.

[1] `https://spinningup.openai.com/en/latest/`