Final report

1st Vitalii Rusinov
Faculty of Informatics
Technical University of Munich
Munich, Germany
vitaliy.rusinov.cv@gmail.com

***Abstract*—This report describes the results of the project "Generalization performance of GNNs for Reinforcement Learning in Robotics" carried out by team 10.**

***Index Terms*—graph neural networks, reinforcement learning, generalization**

I. Key results

- The recurrent GNN model successfully learns robust policies using the on-policy algorithm PPO.
- Transfer learning succeeds for size and disability transfer.
- The generalization performance has further improved after disabling per-element normalization of the observation vector.
- This is the first successful application of a recurrent GNN model for learning robot motions in an off-policy setting. Training with the off-policy algorithm TD3 yields stable high rewards after principled modifications.
- The developed software facilitates future research in the area.

II. Choice of environment

The recent state-of-the-art distributed reinforcement learning framework RlLib has been chosen for on-policy learning. It supports both Tensorflow and PyTorch, provides for stable training using PyTorch. The original author's code for the TD3 publication has been chosen for off-policy learning. The GNN model is based on PyTorch Geometric.

III. Graph structure

The nodes of the robot graph are represented by its joints. In addition, the root node is added to encode the global features of the robot (orientation, z-coordinate, external force onto the robot body). The features of the joints are given by their joint space coordinates, velocities and the external forces applied to them.

IV. GNN model

This work uses the Gated Graph Sequence Neural Network model. The hidden layer size is 64. The input node features consist of the embedded observations vector (of size 32) and the embedded node annotation. The observations vector is embedded using one linear layer with a TanH activation. The node annotation vector consists of 32 learnt features, the nodes with the same function have the same annotation vector. The output stage of the network consists of a two-layer MLP with TanH activations. Its weights are shared: the same network is applied to the final hidden states of each node.

V. Improvements for Off-policy learning

The following modifications were required for successful training using the off-policy algorithm TD3.

First, the Markov assumption has to hold. In the stable-baselines3 framework, the Markov assumption is often not fulfilled because the environment sets the done signal after the episode timeout. The next state no longer depends only on the previous state and action. Training the GGNN model with the additional root node feature indicating the current timestep number has not been successful, the training diverged.

Secondly, training the GGNN requires a lower learning rate of 3e-4. Finally, the important modification is the new output stage that makes the off-policy learning feasible. After the earlier described two-layer MLP with TanH activations, another TanH activation is applied, and the result is multiplied by the maximal allowed motor torque value. Using ReLu activations instead of TanH activations broke the training.

As the result of these changes, the GGNN model features stable training using the off-policy algorithm TD3.

VI. Description of key experiments

1) Policy training for a 6-legged centipede using PPO. Comparison between the learnt GNN and MLP policies. The GGNN and the two two-layer MLP policy models result in similar rewards.
2) Policy training for a 8-legged centipede using PPO. Transfer learning from the 6-legged centipede to the 8-legged centipede. Transfer learning from the 8-legged centipede to the 6-legged centipede.
   The policy brings in high rewards only after a few retraining iterations.
3) Additional study of dependence of the used observations filtering method on the network generalization performance.
   The model has required considerably less retraining iterations after the input observation normalization element-by-element was disabled.
4) Training of the GGNN model for the six-legged centipede using the off-policy algorithm TD3.
   The training of the modified model has resulted in consistent high rewards.

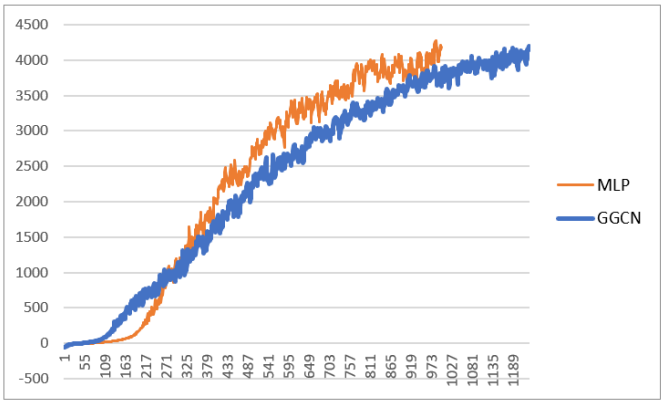

Fig. 1. Rewards for the learnt policies for the Centipede-6 environment.

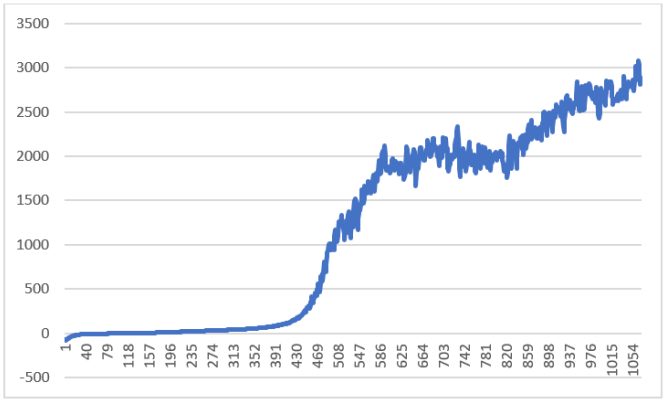

Fig. 2. Rewards for the MLP policy and the Centipede-8 environment.

VII. Conclusion

This is the first application of a recurrent graph neural network for learning robotic movements in an off-policy setting. It is feasible to train the robust recurrent graph convolutional policy model using the algorithm TD3. The result has important practical and research implications. The developed software facilitates future research in the area.

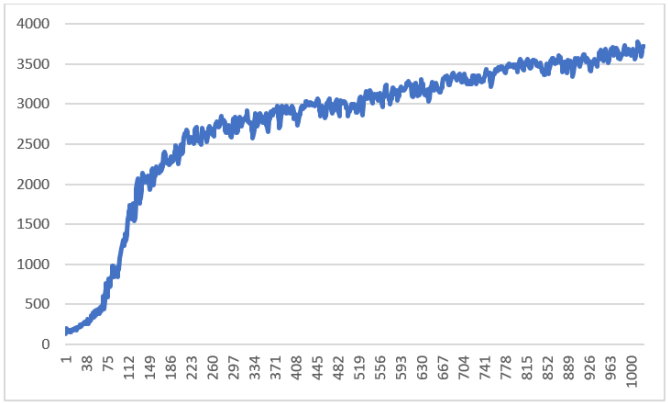

Fig. 3. Transfer learning from Centipede-6 to Centipede-8.

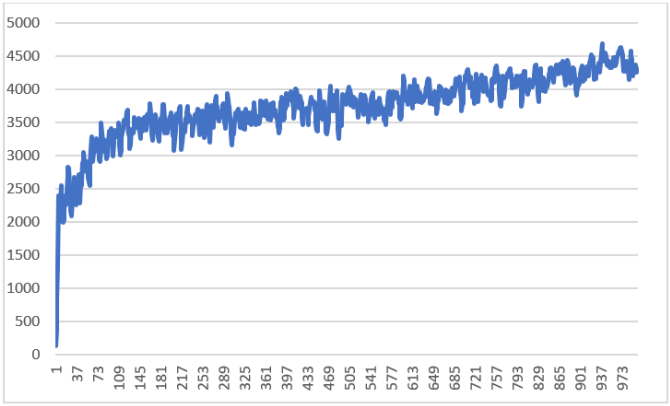

Fig. 4. Transfer learning from Centipede-6 to Centipede-8 with disabled elementwise observation filtering.

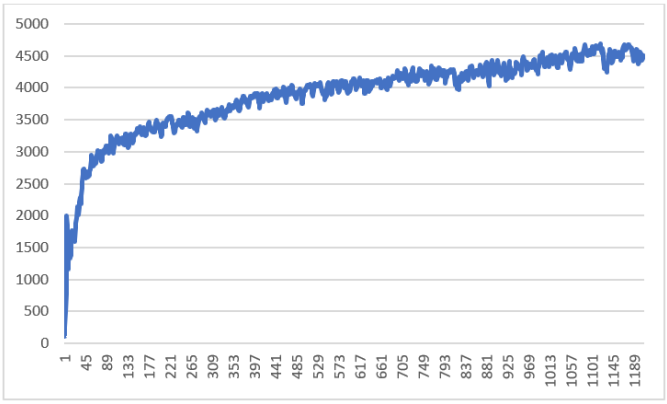

Fig. 5. Transfer learning from Centipede-8 to Centipede-6.

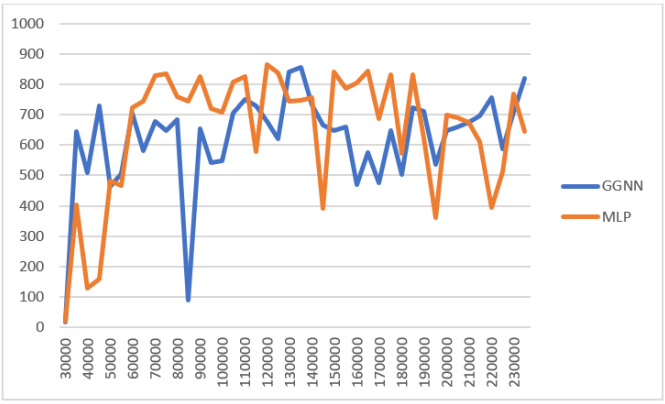

Fig. 6. Evaluation rewards for MLP and GGNN policies during TD3 training for the Centipede-6 environment.