Multi-Agent Systems Control Using Graph Neural Networks with Model-Based Reinforcement Learning

Abstract

Multi-agent systems (MAS) play a crucial role in the advancement of machine intelligence and its applications. To explore complex interactions within MAS settings, we introduce a novel "GNN for MBRL" model. This model employs a state-space Graph Neural Network alongside Model-based Reinforcement Learning to tackle MAS tasks, such as Billiard-Avoidance and Autonomous Driving. The process involves using a GNN model to predict the future states and paths of several agents. Subsequently, a Model Predictive Control, enhanced by the Cross-Entropy Method (CEM), is used to guide the ego-agent's action planning, facilitating successful completion of MAS tasks.

1 Introduction

1.1 Purpose

Vision-based approaches have been extensively researched in various reinforcement learning (RL) areas, including mastering video games directly from raw pixels, managing simulated autonomous vehicles using complex image observations, and carrying out robotic tasks like grasping using state representations derived from complicated visual data. However, it has been shown that RL from complex observations such as raw pixels is time-consuming and needs a lot of samples. Additionally, it is widely acknowledged that learning policies from physical state-based characteristics is significantly more effective and straightforward than learning from visual pixels. Therefore, this research focused on learning control policies from states and exploring the use of a graph neural network (GNN) dynamics model to predict future states in multi-agent systems. We then utilized a Cross-Entropy Method (CEM)-optimized model-based controller for motion planning of the ego-agent, which enabled successful execution of specific MAS missions. These include multi-billiard avoidance and self-driving car scenarios.

1.2 Background

Inspired by a state-space model for videos that reasons about multiple objects and their positions, velocities, and interactions, our project seeks to develop a "GNN for MBRL" model. This model is based on a multi-billiard simulator for sample-efficient model-based control in MAS tasks involving many interacting agents. Autonomous driving, a complicated multi-agent system, requires the ego-agent to consider the situations of surrounding agents when conducting motion planning. The gym-carla can be used for further study in this context. We begin by developing and testing our "GNN for MBRL" model on a MAS billiard avoidance scenario to investigate the possibilities of GNNs and model-based RL. We aim to transfer the framework to real-world self-driving applications.

Graph Neural Networks. GNNs have been proposed to create node and edge representations in graph data, achieving remarkable success in applications such as recommendation systems, social network prediction, and natural language processing. Recognizing the capabilities of GNNs in physical systems, we can utilize GNN-based reasoning to represent objects as nodes and relations as edges, which allows for an effective approach to analyzing objects and relations. A successful

.

example of a GNN is the state-space model, STOVE, which combines a GNN dynamics model for inference with an image reconstruction model to accelerate training and improve the unsupervised learning of physical interactions. Thus, the state-space predictive model is the result of combining these two components:

$z_p(x_t|z_{t-1}) = z_p(z_0)z_p(z_1|z_0)\prod_t z_p(z_t|z_{t-1})$

where x is the image observation and z represents object states. The latent positions and velocities of multiple agents act as the connection between the two components. The model uses simple uniform and Gaussian distributions to initialize the states. The STOVE model is trained on video sequences by maximizing the evidence lower bound (ELBO). STOVE has also extended their video model into reinforcement learning (RL) tasks for planning. Empirical evidence demonstrates that an actor using Monte-Carlo tree search (MCTS) on top of STOVE is comparable to model-free techniques, such as Proximal Policy Optimization (PPO), while needing a fraction of the samples. Inspired by these RL experiments, we apply the GNN model directly to states rather than complex visual data to improve sample efficiency and predict agents' future states. This is then combined with another model-based RL approach, such as Model Predictive Control (MPC). In the experiment, we train the GNN dynamics model using ground truth states of video sequence data for multi-agent systems instead of visual data.

Model-based Reinforcement Learning. Model-based RL is considered a solution to the high sample complexity of model-free RL. This method typically includes two primary steps: (1) creating a dynamics model that predicts future states based on present states and actions, and (2) using a planning method to learn a global policy and act in the environment effectively. The STOVE model uses Monte-Carlo tree search (MCTS) to develop a policy based on the world model, which acts as a planning simulator, and found that MCTS combined with STOVE could outperform the model-free PPO algorithm in a multi-billiards avoidance task.

2 Method

2.1 Framework

The "GNN for MBRL" method consists of two primary stages: (1) a GNN dynamics model training phase, using offline recorded video sequences or low-dimensional states for video prediction, and (2) a motion planning phase using CEM-based Model Predictive Control (MPC). This involves a feedback control algorithm with a Cross-Entropy Method optimizer to interact with the billiard environment. The aim is to plan effective actions for the ego-agent in order to avoid collisions.

There are two different cases in the GNN dynamics training stage. The "Action-conditioned case" follows the STOVE model-based control approach, training GNN with an object reconstruction model on visual data. The "Supervised RL case" is designed for RL tasks directly on low-level states. Both cases involve training GNN dynamics models for predicting future multi-agent states. Subsequently, the trained model is integrated into the model-based RL section to control the ego agent for motion planning.

After training, MPC uses a model to predict future outputs of a process. It handles multi-input multi-output (MIMO) systems with constraints and incorporates future reference information to improve performance. Therefore, we established a continuous version of the multi-billiard environment for data collection. It is possible to combine the previously trained GNN model with MPC to assess if this method can successfully address MAS tasks.

2.2 Data Generation

STOVE proposed an object-aware physics prediction model based on billiard simulations, including avoidance, billiards, gravity, and multi-billiards modes. We wrapped them into a gym environment style, "gym-billiard," which can be easily used by Python API, aiding researchers in understanding this system and creating efficient algorithms.

Our project focuses on the avoidance billiard scenario, where the red ball is the ego-object and the RL agent controls it to avoid collisions. In STOVE, the ego-ball has nine actions: movement in eight directions and staying at rest. A negative reward is given when the red ball hits another. We obtained the avoidance sequences datasets using the "generate_billiards_w_actions" function. 1000 sequences

of length 100 for training and 300 sequences of length 100 for testing were generated using a random action selection policy. The pixel resolution was 32*32 with the ball mass set to 2.0.

We changed the environment to use continuous actions for agents, where the red ball is controlled by 2-dimensional numpy values ranging in (-2, 2), representing the acceleration in x and y directions. Similar to the discrete setting, continuous datasets were produced with random actions from a uniform distribution within (-2, 2). These datasets included the image observations, actions, states, dones, and rewards. The average rewards for the continuous mode are lower, indicating more frequent interactions between the balls.

Table 1: Basic comparisons of the continuous and discrete datasets.

Data Mode	Action space	Actions dtype	Average Rewards of training data	Average Rewards of testing data
Discrete	9	One-hot mode	-17.276	-16.383
Continuous	2	Numpy array	-18.93	-18.71

2.3 GNN Dynamics Model Training

We used supervised learning to train on ground-truth states rather than high-dimensional image data. The aim is to improve sample efficiency, then combine the trained model with CEM-optimized MPC for predicting future states. Two cases were trained on both Discrete and Continuous datasets: (1) the Action-conditioned case, which makes predictions based on state and action and predicts reward, and (2) the Supervised RL case, where real states including positions and velocities were used as the input for GNN dynamics model. The model can learn to predict future states of multiple agents instead of first extracting the states from visual data with a separate model. Training was performed for 500 epochs, and the model parameters were saved. Training time for the Supervised condition was less than the Action-conditioned case. The GNN model could work on both action space 2 and 9 discrete actions without changing the GNN network architecture, resulting in a unified training framework.

2.4 GNN with MBRL

Following traditional Model-based RL, the trained GNN model was combined with CEM-optimized MPC to assess performance on the continuous gym-billiard avoidance task. The saved GNN model predicts future states, and the MPC searches for optimal actions for the ego-agent. The search is repeated after each step to account for prediction errors and rewards, incorporating feedback from the environment. We also conducted experiments on discrete datasets using MCTS and saved videos. For the continuous case, the GNN dynamics model was combined with CEM optimized MPC and compared with random and ground truth scenarios.

3 Results

3.1 GNN Training Results

The datasets generated from the gym-billiard API environment, including image observations, actions, states, dones, and rewards, were stored in pickle files. GNN dynamics models were trained in two conditions: (1) Action-conditioned case, which used video sequences with a visual reconstruction model, and (2) Supervised RL case, which used real states as input for the GNN dynamics model. Both conditions were trained for 500 epochs. The Supervised condition took less time to train than the Action-conditioned case. A notable finding is that the GNN model worked equally well for both action space 2 and 9 discrete actions without changing the original GNN architecture. This allows for unified training for both the Discrete and Continuous billiard avoidance environments. After training, model parameters were saved in the "checkpoints" folder. "gifs" folder stores the videos, and "states" contains state and reward files.

In the Action-conditioned case, the reward MSE loss decreases for both continuous and discrete conditions. However, the continuous reward error decreased from 0.48 to 0, while the discrete one dropped from 0.16 to 0. The ELBO increased significantly from 450 to 3600. Position and velocity prediction errors decreased during training. The continuous position error was close to the discrete,

but the velocity error showed a greater difference. The continuous V_error dropped from 0.65 to 0.05, while the discrete one decreased from 0.07 to 0.01. The four metrics met the criteria for a reasonable GNN dynamics model for the subsequent RL task. In the Supervised RL case, the model directly inputs the ground truth states and actions for GNN training to predict future states. Reconstruction errors were always zero since no image reconstruction was used on the true states. The discrete case showed a better performance compared to the continuous case with respect to the "Prediction_error". The continuous loss remained stable for "Total_error", while the discrete loss showed a downtrend before stabilizing. Generated rollout videos indicated that the ego-red ball performed reasonably well in avoiding collisions. Thus, the trained Supervised RL model can be used for the following model-based RL phase.

3.2 GNN with MBRL

In the "Model-based Control" framework, we used MCTS on discrete datasets to generate qualitative videos. We changed the mass of the ball agents to 1.0 and 2.0 and trained two GNN dynamics models. During MCTS, the GNN model predicted future sequences for 100 parallel environments with the length of 100, using a maximal rollout depth of 10. We then calculated the mean collision rate and saved 100 videos to show the ego-ball's interaction with other agents, which demonstrates improved collision avoidance and lower collision rates.

For the continuous datasets, we combined the trained GNN model into the CEM optimized MPC method and compared it with random and ground truth cases. The GNN model made accurate predictions based on the current states by checking the code, changing the cuda device and data type. We computed the "reward," the average collisions per epoch, for each method.

Table 2: Reward (Collision Rate) for two envs with different baselines.

Envs	Epochs	Horizons	GNN_MPC	Random	Ground_truth
m=1	100	50	0.0558+0.0012	0.2790+0.025	0.0707+0.066
m=1	50	100	0.0565+0.0008	0.3543+0.0445	0.0408+0.0392
m=2	100	50	0.0648+0.001	0.2420+0.0178	0.0505+0.0480
m=2	50	100	0.0455+0.0008	0.2690+0.0350	0.0612+0.0575

The "random" case used randomly generated actions, while "ground_truth" used the true interaction environment for generating next states. The "m=1" version task differed slightly from "m=2" as the "m=1" model was trained on the old continuous datasets, making the red ball movement less flexible. The collision rates in "GNN_MPC" were lower than "Random" and close to "ground_truth". The performance of our proposed method was better than random cases, and the results of "GNN_MPC" were close to the "Ground_truth" case, which indicated that the trained GNN dynamics model predicts the future states of multi-object systems as well as the ground truth interactive environment.

4 Conclusions

We introduced the "GNN for MBRL" concept, combining a graph neural network (GNN) dynamics model with CEM-optimized Model Predictive Control (MPC) on a gym-billiard avoidance MAS task. We also conducted experiments on the "Action-conditioned" case with MCTS using discrete datasets and explored the "Supervised RL" GNN dynamics model with CEM-optimized MPC on continuous datasets. The proposed model predicted video sequences well and controlled the ego-agent to address RL tasks, which may be applied to complex multi-agent systems like the gym-carla autonomous driving environment.