2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Autonomous Exploration Under Uncertainty via Deep Reinforcement Learning on Graphs

Fanfei Chen, John D. Martin, Yewei Huang, Jinkun Wang and Brendan Englot

***Abstract*— We consider an autonomous exploration problem in which a range-sensing mobile robot is tasked with accurately mapping the landmarks in an *a priori* unknown environment efficiently in real-time; it must choose sensing actions that both curb localization uncertainty and achieve information gain. For this problem, belief space planning methods that forward-simulate robot sensing and estimation may often fail in real-time implementation, scaling poorly with increasing size of the state, belief and action spaces. We propose a novel approach that uses graph neural networks (GNNs) in conjunction with deep reinforcement learning (DRL), enabling decision-making over graphs containing exploration information to predict a robot's optimal sensing action in belief space. The policy, which is trained in different random environments without human intervention, offers a real-time, scalable decision-making process whose high-performance exploratory sensing actions yield accurate maps and high rates of information gain.**

I. Introduction

It is challenging to solve an autonomous mobile robot exploration problem [1] in an *a priori* unknown environment when localization uncertainty is a factor, which may compromise the accuracy of the resulting map. Due to our limited ability to plan ahead in an unknown environment, a key approach to solving the problem is often to estimate and select the immediate next best viewpoint. The next-view candidates may be generated by enumerating frontier locations or using sampling-based methods to plan beyond frontiers. A utility function may be used to evaluate the next-view candidates and select the optimal candidate. It is common to forward-simulate the actions and measurements of each candidate and choose the best one as the next-view position [2], [3]. However, as the size of the robot's state and action space increases, the computation time of this decision-making process grows substantially. Although learning-based exploration algorithms can accurately predict optimal next-view positions with nearly constant computation time, these algorithms may suffer when transferring a learned policy to a new unknown environment due to poor generalizability. Our recent prior work [4] proposed a generalized graph representation for mobile robot exploration under localization uncertainty, compatible with graph neural networks (GNNs). However, the hyperparameters are difficult to tune and the performance in test is unstable because the size of the graph is always changing throughout the exploration process.

In this paper, we aim to learn a more robust and generalizable policy for exploration using deep reinforcement learning (DRL), where the optimal view can be estimated by the DRL policy instead of explicitly computing the expected utility of each candidate sensor view. Hence our method achieves a nearly constant-time, real-time viable decision-making process. Our *exploration graph* is a generalized state representation provided as input. Instead of learning from camera images or metric maps, such a graph reduces the dimensionality of our state space, so the policy can be exposed to and tested in a wide diversity of environments. Fig. 1 summarizes our general approach. The policy is trained in environments with randomly distributed landmarks, and while exploring, predictions of the next best view are provided by the learned policy. The policy is generalizable to environments of different sizes, and with different quantities of landmarks, whose graph topologies are well-represented despite our framework being trained in simpler environments.

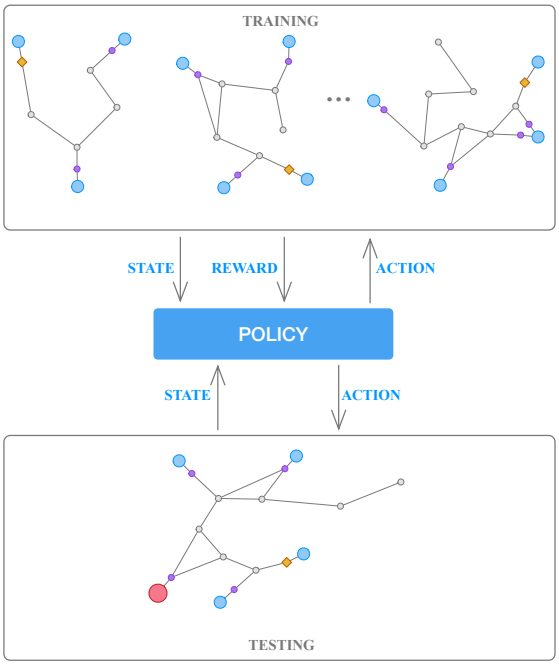


Fig. 1: **An illustration of our framework.** In the *training phase*, the robot is trained in different random environments with a random initial robot location and distribution of landmarks. The current state is represented by an *exploration graph*. The robot pose history, observed landmarks, the current pose and candidate frontier waypoints are indicated by gray circles, purple circles, an orange square and blue circles respectively. The parameters are optimized in the policy network and/or the value network according to the reward given by the environment. Then in the *testing phase*, we generate a different set of random environments. The optimal action is estimated by the trained policy, which is indicated in red.

F. Chen, J. D. Martin, Y. Huang, J. Wang and B. Englot are with the Department of Mechanical Engineering, Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ, 07030. {fchen7, jmarti3, yhuang85, jwang92, benglot}@stevens.edu

In this paper, we present the first framework to perform reinforcement learning with graph neural networks for the purpose of teaching a mobile robot how to efficiently explore unknown environments under localization uncertainty, while building an accurate map. We demonstrate that this allows our *exploration graph* abstraction to support robust and generalizable learning, which also scales favorably in problem size and performance against state-of-the-art approaches.

A. Related Work

Information-theoretic exploration methods often use a prediction of the mutual information (MI) between a robot's future sensor observations and the cells of an occupancy map to evaluate the next-view candidates being considered [2], [3]. A similar approach is applied specifically to the frontiers of Gaussian process occupancy maps [5] to efficiently predict the most informative sensing action. However, robot localization uncertainty is not considered in these approaches.

Active simultaneous localization and mapping (SLAM) exploration methods consider both map entropy and localization and/or map uncertainty. The active pose SLAM approach of [6] uses the sum of map and robot trajectory entropy to guide a robot's exploration of an unknown environment. Particle filtering is employed in [7] to manage uncertainty during exploration by transforming trajectory uncertainty into particle weights. *Virtual landmarks* are proposed in the Expectation-Maximization (EM) exploration algorithm [8] to model the impact of a robot's pose uncertainty on the accuracy of an occupancy map, employing a utility function favoring sensing actions that drive down the collective uncertainty of the virtual landmarks residing in all map cells.

The above methods use forward simulation to evaluate the mapping outcomes and uncertainty of next-view candidates, and so their computational cost increases substantially with an increasing number of next-view candidates. Learning-based exploration methods offer the prospect of improved scalability for selecting the next best view, with and without localization uncertainty. Learning-aided information-theoretic approaches [9], [10] are proposed to reduce the cost of predicting MI. Deep neural networks in [11], [12] and [13] can be trained to predict the optimal next-view candidate through supervised learning and reinforcement learning, by taking occupancy grid maps as input data. However, the state space of such maps is very large, and a learned policy may fail when tested in a completely new environment.

Graphs can offer generalized topological representations of robot navigation, permitting a much smaller state space than metric maps. GNNs incorporate graphs into neural network models, permitting a variety of queries to be posed over them [14]. Sanchez-Gonzalez et al. [15] proposed to use graph models of dynamical systems to solve control problems with Graph Nets [16]. Exploration under localization uncertainty is achieved in our prior work [4] through supervised learning over graphs, which gives an unstable result because the number of nodes in each graph differs, and the hyperparameters of the loss function are hard to tune. A novel graph localization network is proposed in [17] to guide a robot through visual navigation tasks. Wang et al. [18] introduces GNNs as policy networks and value networks, through their integration into DRL, to solve control problems.

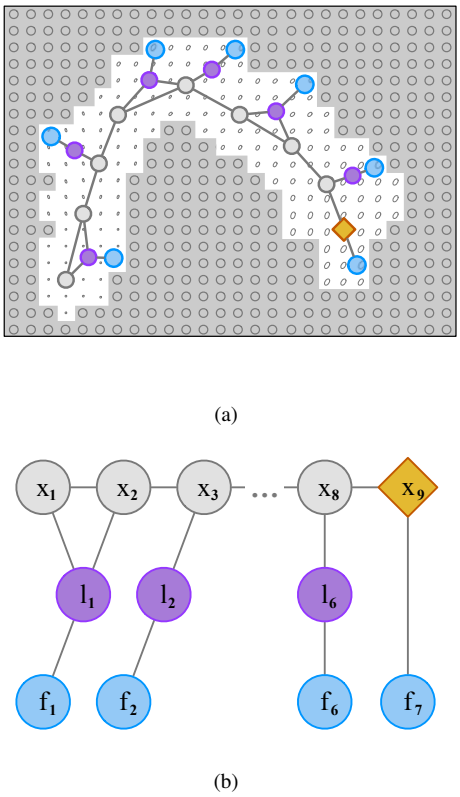


Fig. 2: **Formulating and extracting the exploration graph.** Top: the grayscale color represents the probability of occupancy of a given map cell (assumed to be 0.5 for unobserved cells). The ellipse in each cell represents the estimation error covariance of each *virtual landmark*, with true landmarks shown in purple. Past robot poses, the current pose, and candidate frontiers are indicated by gray circles, an orange diamond, and blue circles respectively. Landmark nodes and the current pose node are connected with their nearest frontiers. Bottom: An input exploration graph is extracted from the current exploration state. Each edge in this graph is weighted with the Euclidean distance between the two vertices connected.

We combine the exploration graphs proposed in [4] with DRL algorithms to learn robot exploration policies. The policy can be trained without human intervention. Our proposed approach offers a generalized representation of a SLAM-dependent mobile robot's state and environment, and a self-learning mechanism offering wide applicability.

B. Paper Organization

The definition of our mobile robot active SLAM exploration problem, and the corresponding exploration graph abstraction, is provided in Section II. In Section III, a framework for DRL with GNNs to address robot exploration under uncertainty is presented. Experimental results are given in Section IV, with conclusions in Section V.

II. Problem Formulation and Approach

A. Simultaneous Localization and Mapping Framework

We adopt a graph-based approach in the SLAM framework supporting our robot's exploration. We then solve the SLAM problem as a least-squares smoothing problem. The robot motion model and measurement models are defined as:

$$\mathbf{x}_i = h_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{w}_i, \quad \mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, Q_i), \tag{1}$$

$$\mathbf{z}_k = g_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, R_k), \tag{2}$$

where $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{t}$ are poses along the trajectory, $\mathcal{L} = \{\mathbf{l}_j\}_{j=1}^{m}$ are $m \in \mathbb{N}$ landmarks, and $\mathcal{U} = \{\mathbf{u}_\mathbf{i}\}_{i=1}^{t}$ is a given sequence of low-level actions. Then we solve the SLAM problem as a least-squares problem:

$$\begin{aligned}\mathcal{X}^*, \mathcal{L}^* = \arg\min_{\mathcal{X},\mathcal{L}} &\sum_i \left\|\mathbf{x_i} - h_i(\mathbf{x_{i-1}}, \mathbf{u_i})\right\|^2_{Q_i} \\ &+ \sum_k \left\|\mathbf{z_k} - g_k(\mathbf{x_{i_k}}, \mathbf{l_{j_k}})\right\|^2_{R_k}.\end{aligned} \tag{3}$$

$\mathcal{X}^*$ and $\mathcal{L}^*$ are obtained using the GTSAM [19] implementation of iSAM2 [20] to perform nonlinear least-squares smoothing over a factor graph. The Gaussian marginal distributions and Gaussian joint marginal distributions are produced from this graph-based inference procedure.

We next introduce the *virtual map* of the EM algorithm for exploration under localization uncertainty [8], which is a uniformly discretized grid map comprised of *virtual landmarks*, $\tilde{\mathbf{l}}_k \in \tilde{\mathcal{L}}$, to represent the uncertainty and occupancy probability of every map cell. Each cell of the virtual map contains a virtual landmark with a large initial covariance (illustrated in Figures 2(a) and 3(a)). We use A-optimality [21] for our uncertainty criterion:

$$\phi_\mathrm{A}(\Sigma) = \mathrm{tr}(\Sigma), \tag{4}$$

where Σ is the error covariance matrix for all virtual landmarks in the map. Accordingly, the uncertainty of the current state is quantified by the trace of the covariance of all virtual landmarks. The definition of the utility function is as follows:

$$U(\tilde{\mathcal{L}}) = \sum_{\tilde{\mathbf{l}}_k \in \tilde{\mathcal{L}}} \phi_\mathrm{A}(\Sigma_{\tilde{\mathbf{l}}_k}). \tag{5}$$

B. Exploration Graph

The definition of the *exploration graph* is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{X} \cup \mathcal{L} \cup \mathcal{F}$, and where $\mathcal{X}, \mathcal{L}$, and $\mathcal{F}$ are sets of previous poses, landmarks, and candidate frontier nodes respectively. The edges $\mathcal{E}$ connecting pose to pose $\mathbf{x}_i$ — $\mathbf{x}_{i+1}$, pose to landmark $\mathbf{x}_{i_k}$ — $\mathbf{l}_{j_k}$, landmark to frontier $\mathbf{l}_{j_k}$— $\mathbf{f}_{n_k}$, and the current pose to its nearest frontier $\mathbf{x}_t$— $\mathbf{f}_{n_t}$ are weighted with the Euclidean distances between those nodes. Each node $\mathbf{n_i} \in \mathcal{V}$ has a feature vector

$$\mathbf{s}_i = [s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4}, s_{i_5}],$$

$$s_{i_1} = \phi_\mathrm{A}(\Sigma_i), \tag{6}$$

$$s_{i_2} = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2}, \tag{7}$$

$$s_{i_3} = \arctan2(y_i - y_t, x_i - x_t), \tag{8}$$

$$s_{i_4} = p(m_i = 1), \tag{9}$$

$$s_{i_5} = \begin{cases} 0 & \mathbf{n_i} = \mathbf{x_t} \\ 1 & \mathbf{n_i} \in \{\mathbf{f_n}\} \\ -1 & \text{otherwise} \end{cases}. \tag{10}$$

The contents of the feature vector are as follows. In Eq. (6), the A-Optimality criterion, derived from our virtual map $\tilde{\mathcal{L}}$, is used to quantify a node's uncertainty. Eqs. (7) and (8) provide relative pose information; the Euclidean distance and the relative orientation between the current robot pose and each node in the exploration graph. Occupancy information is extracted from an occupancy grid map, where m_i is the occupancy of the map cell associated with node n_i in Eq. (9). Eq. (10) provides an indicator variable, denoting the current pose to be 0, all frontiers to be 1, and all other nodes -1.

We note that frontier nodes are sampled from the boundary cells between free and unexplored space in the occupancy map. The landmarks and the current robot pose are the only nodes connected to frontiers, and each connects only to its nearest frontier node. It is possible for multiple landmark nodes to be connected to the same frontier node, but not for a landmark or pose node to connect to multiple frontier nodes; only the nearest frontiers are connected into the graph. The composition of an exploration graph is shown in Fig. 2.

III. Algorithms

A. Graph Neural Networks

We explore three different graph neural networks in this paper: Graph Convolutional Networks (GCNs) [22], Gated Graph Neural Networks (GG-NNs) [23] and Graph U-Nets (g-U-Nets) [24]. The GCN model performs convolution operations on graphs using the information of neighbors around each node. GG-NNs adopt a gated sequential unit to update the node feature vectors of graphs. Finally, g-U-Nets include pooling and unpooling layers to achieve an encoder-decoder mechanism similar to U-Net [25]. Each GNN model has three hidden layers and a multilayer perceptron (MLP) output model. We add a dropout layer between each GNN layer to prevent overfitting. These GNNs are used in our RL framework as policy networks and/or value networks.

B. Deep Reinforcement Learning

Reinforcement learning describes a sequential decision making problem, whereby an agent learns to act optimally from rewards collected after taking actions. In our setting, we consider changes to the exploration graph that result from our selection of frontier nodes, which represent waypoints lying on the boundaries between mapped and unmapped areas. At each step $k \in \mathbb{N}$ the environment state is captured

Algorithm 1: Reward Function

input: Exploration graph $\mathcal{G}$, Frontier node $\mathbf{f}$
Normalize the raw reward Alg. 2
$\mathcal{R}_\mathcal{G} = \{r_{\mathbf{f}'} = \texttt{raw\_reward}(\mathcal{G}, \mathbf{f}') \; \forall \; \mathbf{f}' \in \mathcal{A}_\mathcal{G}\}$
$l = \min \mathcal{R}_\mathcal{G}$, $u = \max \mathcal{R}_\mathcal{G}$
$r_\mathbf{f} \leftarrow (r_\mathbf{f} - l)/(u - l)$
Compute projection based on nearest frontier
$\mathbf{f}_t = \texttt{nearest\_frontier}(\mathbf{x}_t)$
if $u = \texttt{raw\_reward}(\mathbf{f}_t)$ **then**
  return $r_\mathbf{f} - 1$ # $r(\mathcal{G}, \mathbf{f}) \in [-1, 0]$
end
return $2r_\mathbf{f} - 1$ # $r(\mathcal{G}, \mathbf{f}) \in [-1, 1]$

Algorithm 2: Raw reward

input: Exploration graph $\mathcal{G}$, Frontier node $\mathbf{f}$
$\mathcal{U} \leftarrow \texttt{path\_planner}(\mathcal{G}, \mathbf{f})$
Compute raw reward with (5), Alg 3 and cost-to-go
return $U(\tilde{\mathcal{L}}) - \texttt{compute\_utility}_{\tilde{\mathcal{L}}}(\mathcal{U}) - \alpha C(\mathcal{U})$

by the exploration graph $\mathcal{G}_k \in \mathsf{G}$ [1]. The robot chooses a frontier node to visit based on the graph topology: $\mathbf{f}_k \in \mathsf{F}_{\mathcal{G}_k}$. This causes a transition to $\mathcal{G}_{k+1} \in \mathsf{G}$ and a scalar reward $R \in \mathbb{R}$ to be emitted. The interaction is modeled as a Markov Decision Process $\langle \mathsf{G}, \mathsf{F}, \Pr, \gamma \rangle$ [26], associated with the transition kernel $\Pr\colon \mathsf{G}\times\mathsf{F} \to \mathscr{P}(\mathsf{G}\times\mathbb{R})$ that defines a joint distribution over the reward and next exploration graph, given the current graph and frontier node pair. Here, $\gamma \in [0, 1)$ is a discount factor used to control the relative importance of future rewards in the agent's learning objective.

In this paper we consider value-based methods and policy-based methods for model-free control. Value methods strive to maximize the expected sum of future rewards, or the *value*:

$$Q_{\boldsymbol{\theta}}(\mathcal{G}, \mathbf{f}) = \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^k R(\mathcal{G}_k, \mathbf{f}_k) \;\middle|\; \mathcal{G}_0 = \mathcal{G}, \mathbf{f}_0 = \mathbf{f}\right]. \quad (11)$$

For large-scale problems, the value function is represented with a parameterized function, such as a convolutional neural network or, in our study, a graph neural network.

To train $Q_{\boldsymbol{\theta}}$, we gather transition samples of the form $(\mathcal{G}, \mathbf{f}, r, \mathcal{G}')$ using an action sampling strategy that chooses the most uncertain action, which has the largest value at the output. According to [27], the uncertainty of the actions can be represented by the output of the dropout layer. During training, we first dropout 90% of our data before the MLP output model, and as the training phase runs, we gradually decrease the dropout rate until it reaches 0%, which means the policy provides greedy action selection. The parameters $\boldsymbol{\theta}$ are adjusted using stochastic gradient descent to minimize the loss function $L\colon \mathcal{B} \to \mathbb{R}$ over sampled minibatches $\mathcal{B} \sim \mathcal{D} = \{(\mathcal{G}, \mathbf{f}, r, \mathcal{G}')_k\}_{k\in\mathbb{N}}$. In this paper we consider the

[1]The exploration graph $\mathcal{G}$ encodes the full history of robot poses.

Algorithm 3: Compute Utility

global: Virtual landmarks $\tilde{\mathcal{L}}$
input: Sequence of low-level actions $\mathcal{U}$
Execute low-level actions with [8]
$\tilde{\mathcal{L}} \leftarrow \texttt{update\_virtual\_map}(\mathcal{U})$
Compute uncertainty estimate with (5)
return $U(\tilde{\mathcal{L}})$

Algorithm 4: Exploration Training with RL

initialize: $\mathcal{G}$, $\boldsymbol{\theta}$, $step$
$\mathcal{D} \leftarrow \emptyset$
while $step < max\_training\_steps$ **do**
  if *RL="DQN"* **then**
    # Gather experience visiting frontier nodes
    $\mathbf{f} \leftarrow \texttt{action\_sampling}(\mathcal{G})$
    $\mathcal{G}', r \leftarrow \texttt{visit\_frontier}(\mathcal{G}, \mathbf{f})$
    $step \leftarrow step + 1$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{G}, \mathbf{f}, r, \mathcal{G}'\}$
    $\mathcal{G} \leftarrow \mathcal{G}'$
    # Train policy with DQN algorithm
    $\pi_{\boldsymbol{\theta}} \leftarrow \texttt{dqn}(\mathcal{D})$
  else if *RL="A2C"* **then**
    # Gather experience visiting frontier nodes
    $\mathbf{f} \leftarrow \pi_{\boldsymbol{\theta}}(\mathcal{G})$
    $\mathcal{G}', r \leftarrow \texttt{visit\_frontier}(\mathcal{G}, \mathbf{f})$
    $step \leftarrow step + 1$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{G}, \mathbf{f}, r, \mathcal{G}'\}$
    $\mathcal{G} \leftarrow \mathcal{G}'$
    # Train policy with A2C algorithm
    if $step \mod policy\_update\_steps = 0$ **then**
      $\pi_{\boldsymbol{\theta}} \leftarrow \texttt{a2c}(\mathcal{D})$
      $\mathcal{D} \leftarrow \emptyset$
end
return: $\pi_{\boldsymbol{\theta}}$

DQN [28] loss function:

$$L_{\text{DQN}}(\mathcal{D}) = \mathbf{E}_{\mathcal{B}\sim\mathcal{D}}[(r + \gamma \max_{\mathbf{f}'\in\mathsf{F}_{\mathcal{G}'}} Q(\mathcal{G}', \mathbf{f}') - Q(\mathcal{G}, \mathbf{f}))^2], \quad (12)$$

which encodes the expected squared TD-error of one-step predictions over $\mathcal{B}$.

We also consider the policy-based method A2C [29], that directly trains a parameterized policy $\pi_{\boldsymbol{\theta}}\colon \mathsf{G} \to \mathscr{P}(\mathsf{F})$ from data gathered following that policy. We use two separate GNN models to serve as the policy network and the value network, and train it with the loss function:

$$L_{\text{A2C}}(\mathcal{D}) = \mathbf{E}_{\mathcal{B}\sim\mathcal{D}}[L^{(1)}_{\text{A2C}} + \eta L^{(2)}_{\text{A2C}}], \quad (13)$$
$$L^{(1)}_{\text{A2C}} = [A(\mathcal{G}, \mathbf{f}) - V(\mathcal{G}))\log\pi(\mathbf{f}|\mathcal{G}) + \beta(A(\mathcal{G}, \mathbf{f})]^2,$$
$$L^{(2)}_{\text{A2C}} = \sum_{\mathbf{f}\in\mathsf{F}_\mathcal{G}} \pi(\mathbf{f}|\mathcal{G})\log\pi(\mathbf{f}|\mathcal{G}).$$

Here, $L^{(1)}_{\text{A2C}}$ and $L^{(2)}_{\text{A2C}}$ denote the loss terms for a single transition sample. The function $A(\mathcal{G}, \mathbf{f}) = Q(\mathcal{G}, \mathbf{f}) - V(\mathcal{G})$ is called the *advantage*; it computes the difference between the state-action value function Q and the state value function

$V(\mathcal{G}) = \max_{\mathbf{f} \in \mathrm{F}_\mathcal{G}} Q(\mathcal{G}, \mathbf{f})$. We use $\beta \in \mathbb{R}$ as a coefficient for the value loss, and we use $\eta \in \mathbb{R}^+$ as a coefficient for the entropy of the output, to encourage exploration within the RL solution space.

C. Reward Function

In Algorithm 1, we use linear normalization functions to map the range of the raw reward. If the optimal frontier is the one associated with the current pose, the maximum of the reward is 0, otherwise, it is 1 in all other cases. Algorithm 2 is designed based on the utility function of the EM exploration algorithm, Eq. (5), with an additional term penalizing the travel distance associated with an exploratory action. The raw reward is the difference in utility between the current state and the subsequent state by taking the actions $\mathcal{U}$. The cost-to-go $C(\mathcal{U})$ with factor α encourages short travel distances in the raw reward function. The goal is to minimize this weighted combination of map uncertainty and travel distance with every given visit to a frontier node.

D. Exploration Training with RL

As shown in Algorithm 4, using an action sampling strategy, the robot chooses a next-view frontier based on an *exploration graph*. A reward and a new exploration graph are assigned after reaching the frontier most recently selected. The policy is trained using recorded experience data.

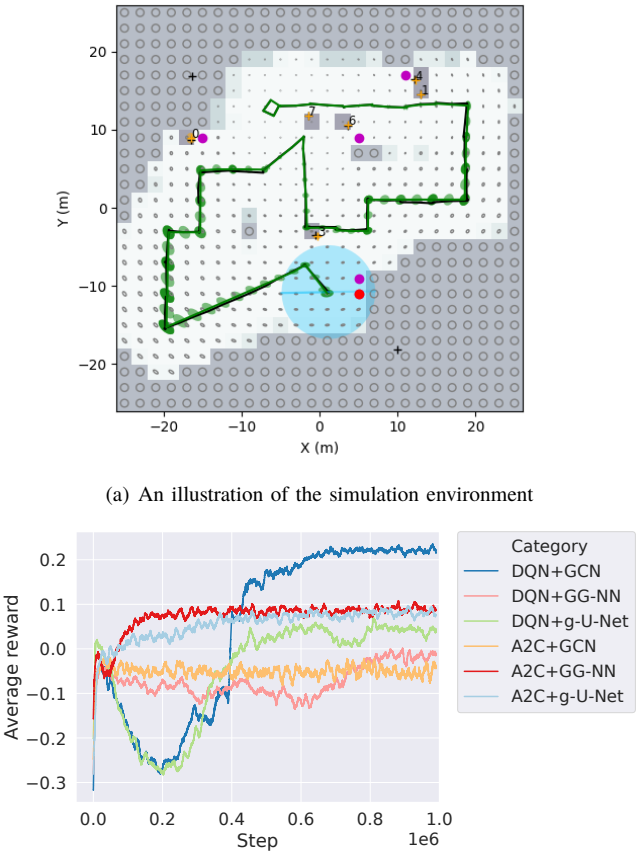
IV. Experiments and Results

A. Experimental Setup

Our exploration policy is trained in a 2D simulation environment. The simulated robot has a horizontal field of view of 360° (±0.5°). The measurement range of the simulated robot is $5m$ ($\pm 0.02m$). While exploring, the robot can rotate from $-180°$ to 180° (±0.2°), and travel from $0m$ to $2m$ ($\pm 0.1m$) at each timestep. All simulated noise is Gaussian, and we give its standard deviation in the intervals above. Given the goal frontier location, the robot will first turn to the goal and then drive directly to the goal following a straight-line path. We use an occupancy grid map to describe the environment. Each occupancy map is $40m \times 40m$ with randomly sampled landmarks and random initial robot locations. The *feature density* of landmarks is 0.005 per m^2 in our experiment. To simplify the problem, landmarks are assumed passable, so obstacle avoidance is not needed.

The virtual map is made up of $2m \times 2m$ square map cells, with a $1m^2$ initial error covariance in each dimension for the virtual landmarks. We note that virtual landmarks are only used to evaluate the reward of Alg. 2, which trades map accuracy against travel expense. Meanwhile, the virtual landmarks are excluded from both SLAM factor graphs and the exploration graph. The exploration task will be terminated once 85% of a map has been observed.

The simulation environment is written in Python and C++, and our graph neural network models are trained using PyTorch Geometric [30]. Our code is freely available on Github[2] for use by others.



(a) An illustration of the simulation environment

(b) Average reward during the training process

Fig. 3: The training performance of different methods on randomly generated simulation environments.

B. Policy Training

The training environments are generated with uniformly randomly sampled landmarks and initial robot locations. A training environment example is shown in Fig. 3(a); the uncertainty of virtual landmarks is represented by the error ellipses in each cell of the map. The blue circle represents the current field of view and the center point is the current robot location. The green error ellipses represent the uncertainty of past robot poses. The magenta points are frontier candidates, which comprise the current action space, and the red point is the selected next-view frontier. Landmarks are indicated by plus signs, which we plot in yellow once observed by the robot. Cells containing true landmarks are denoted as occupied in our map to mark their locations (ensuring a unique and non-trivial occupancy map is obtained from every exploration trial), but the landmarks are infinetesimal in size, and assumed not to present occlusions or collision hazards.

The exploration policy is trained by DQN and A2C with three different GNN models each. The performance of each

[2] https://github.com/RobustFieldAutonomyLab/DRL_graph_exploration

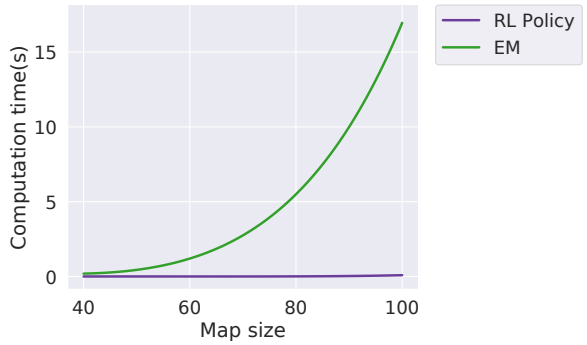


Fig. 4: Computation time for exploration decision-making on different map sizes, which are square in the dimension indicated. Timing was evaluated on a computer equipped with an Intel i9 8-core 3.6Ghz CPU and an Nvidia GeForce Titan RTX GPU. The average computation time is 0.04427s for the RL policy.

approach is shown in Fig. 3(b). For further use and study in exploration experiments, we select the policies that achieve the highest average reward for each RL framework, which are DQN+GCN and A2C+GG-NN.

C. Computation Time

The decision-making process is the most time-consuming component of exploration. The EM exploration algorithm uses forward simulation of a robot's SLAM process (including the propagation of uncertainty across virtual landmarks) to select the best next-view position from the available candidates. Hence the time complexity of the EM algorithm is $\mathcal{O}(N_{action}(C_1+C_2))$, where N_{action} is the number of the candidate actions, C_1 is the cost of each iSAM2 predictive estimate over a candidate trajectory (a function of the number of mapped true landmarks and prior poses) and C_2 is the cost of the covariance update for virtual landmarks (a function of the number of prior poses and the number of virtual landmarks). We compare computation time between the EM algorithm and the RL policy on four different sizes of maps, namely $40m\times40m$, $60m\times60m$, $80m\times80m$ and $100m\times100m$. Each size has the same feature density, which is 0.005 landmarks per m^2. As shown in Fig. 4, the average computation time of the EM algorithm grows dramatically with increasing size of the map, which leads to larger action, state and belief spaces. The EM algorithm will ultimately prove unsuitable for real-time exploration in large, complex environments. On the other hand, the RL policies use graph neural networks to predict the best action so that the computation time of the RL policy remains low, which meets the requirements for real-time application in high-dimensional state and action spaces.

D. Exploration Comparison

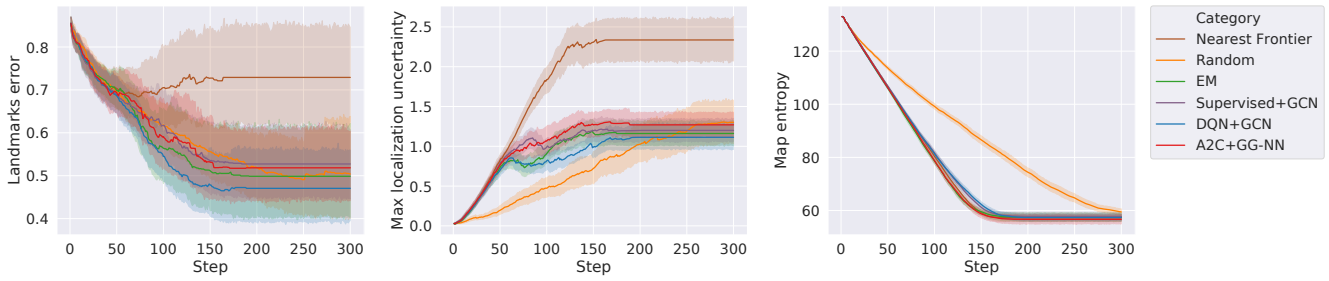
We compared the learned policies with (1) a nearest frontier approach, (2) the selection of a random frontier, (3) the EM approach, and (4) the GCN model trained by supervised learning in [4] over 50 exploration trials. For each trial, every approach uses the same random seed to generate the same environment. Results of this comparison are shown in Figure 5, where we evaluate three metrics: the average landmark uncertainty (of real landmarks only), the maximum localization uncertainty along the robot's trajectory, and occupancy map entropy reduction. The EM algorithm offers the best performance, but is not real-time viable in high-dimensional state-action spaces. Both the Nearest Frontier method and the Random method are real-time viable methods. However, the Nearest Frontier method has the highest landmark and localization uncertainty, and the Random method has the lowest exploration efficiency. The learning-based methods can address real-time applications and offer low landmark and localization uncertainty. The Supervised+GCN method and DQN+GCN method have similar map entropy reduction rates, but DQN+GCN offers lower landmark and localization uncertainty because it travels longer distances from one iteration to the next by selecting high-value actions. In Fig. 5(c), both supervised+GCN and DQN+GCN have slower entropy reduction rates than EM and the A2C+GG-NN policy. When supervised+GCN does not perform with high accuracy, it will tend toward random action selection, because it is trained using binary labels.

Unlike supervised learning, the RL policies are seeking the largest value for each step. Thus, when DQN+GCN does not have high accuracy, it will choose the highest-value action based on predicted values. Although the DQN+GCN policy has relatively low accuracy in predicting the highest-value action, it still selects high-value actions instead of random ones, which leads to more accurate mapping. The A2C+GG-NN policy has the highest exploration efficiency among learning approaches, and offers lower landmark uncertainty than the Supervised+GCN method, but not lower than the DQN+GCN method. Representative exploration trials for all of the algorithms compared in Figure 5 are provided in our video attachment.

E. Scalability

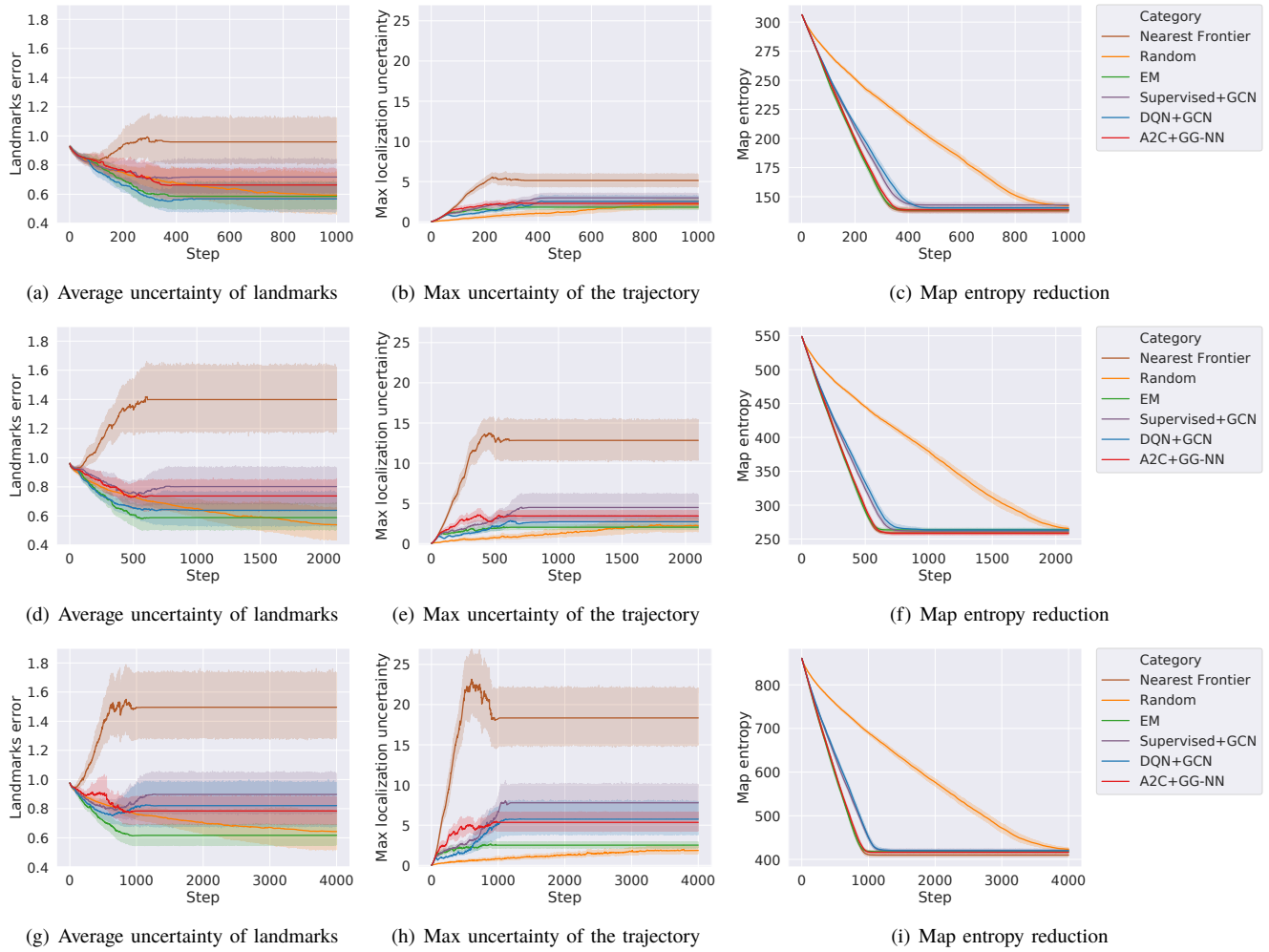
During testing, it is possible to encounter large-scale environments that induce a lengthier pose history, more landmarks, and more frontiers, generating larger exploration graphs over longer-duration operations than a robot may have been exposed to during training. We demonstrate here that our RL policies can be trained in small environments, and scale to large environments to perform exploration tasks. In this experiment, our RL policies are trained on $40m\times40m$ maps with 0.005 landmarks per m^2 feature density. The size of the testing environments are $60m\times60m$, $80m\times80m$ and $100m\times100m$ with the same feature density.

The exploration results are shown in Fig. 6. The A2C+GG-NN method has nearly the same exploration efficiency as the EM algorithm and the Nearest Frontier approach, which all explore faster than the Supervised+GCN and DQN+GCN methods as shown in Figs. 6(c), 6(f), and 6(i). The landmark uncertainty (Figs. 6(a), 6(d), 6(g)) and localization uncertainty (Fig. 6(b), 6(e), 6(h)) of our learning-based methods gradually degrade in performance with increasing map size. The Supervised+GCN method accumulates the largest landmark and localization uncertainty among learning-based



(a) Average uncertainty of landmarks (b) Max uncertainty of the trajectory (c) Map entropy reduction

Fig. 5: The result of 50 exploration trials of each method, with the same randomly initialized landmarks and robot start locations, on $40m \times 40m$ maps (the first three metrics shown are plotted per time-step of the simulation).



(a) Average uncertainty of landmarks (b) Max uncertainty of the trajectory (c) Map entropy reduction

(d) Average uncertainty of landmarks (e) Max uncertainty of the trajectory (f) Map entropy reduction

(g) Average uncertainty of landmarks (h) Max uncertainty of the trajectory (i) Map entropy reduction

Fig. 6: The results of 50 exploration trials on large-size maps (each of which is larger than the $40m \times 40m$ maps used for training); (a), (b), (c) represent $60m \times 60m$ maps, (d), (e), (f) represent $80m \times 80m$ maps, and (g), (h), (i) represent $100m \times 100m$ maps.

methods for each map size. DQN+GCN achieves better results over $60m \times 60m$ and $80m \times 80m$ maps than A2C+GG-NN, but the performance of A2C+GG-NN is better than DQN+GCN over $100m \times 100m$ maps because of the overfitting of DQN+GCN. Overall, A2C+GG-NN demonstrates the best scalability performance based on its high exploration efficiency and relatively low landmark and localization uncertainty across all trials. Representative exploration trials showing the performance of A2C+GG-NN across all of the map sizes examined are provided in our video attachment.

V. Conclusions

We have presented a novel approach by which a mobile robot can learn, without human intervention, an effective

policy for exploring an unknown environment under localization uncertainty, via exploration graphs in conjunction with GNNs and reinforcement learning. The exploration graph is a generalized topological data structure that represents the states and the actions relevant to exploration under localization uncertainty, and we build upon our previous work that used them for supervised learning with GNNs.

Through our novel integration of this paradigm with reinforcement learning, a robot's exploration policy is shaped by the rewards it receives over time, and a designer does not need to tune hyperparameters manually, as is the case for supervised learning. Additionally, the policy learned from RL algorithms is non-myopic and robust across a variety of test environments. Our learned RL policies have exhibited the best performance among the real-time viable exploration methods examined in this paper. Moreover, we have shown that policies learned in small-scale, low-dimensional state-action spaces can be scalable to testing in larger, higher-dimensional spaces. The RL policies learned offered high exploration efficiency and relatively low map and localization uncertainty among the real-time viable exploration methods examined, across the various examples explored. In the future, we will embed obstacle information from metric maps into exploration graphs to train a robot in simulation environments and test in real-world environments.

Acknowledgments

This research has been supported by the National Science Foundation, grant numbers IIS-1652064 and IIS-1723996.

References

[1] S. Thrun, W. Burgard and D. Fox, "Exploration," *Probabilistic Robotics*, pp. 569–605, MIT Press, 2005.

[2] B. J. Julian, S. Karaman and D. Rus, "On Mutual Information-Based Control of Range Sensing Robots for Mapping Applications," *The International Journal of Robotics Research*, vol. 33(10), pp. 1375–1392, 2014.

[3] B. Charrow, S. Liu, V. Kumar and N. Michael, "Information-theoretic Mapping using Cauchy-Schwarz Quadratic Mutual Information," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4791–4798, 2015.

[4] F. Chen, J. Wang, T. Shan, and B. Englot, "Autonomous Exploration Under Uncertainty via Graph Convolutional Networks," *Proceedings of the International Symposium on Robotics Research*, 2019.

[5] M. G. Jadidi, J. V. Miró and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, vol. 42, pp. 273–290, 2018.

[6] R. Valencia, J. V. Miró, G. Dissanayake and J. Andrade-Cetto, "Active Pose SLAM," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1885–1891, 2012.

[7] C. Stachniss, G. Grisetti and W. Burgard, "Information Gain-based Exploration using Rao-Blackwellized Particle Filters," *Proceedings of Robotics: Science and Systems*, pp. 65–72, 2005.

[8] J. Wang and B. Englot, "Autonomous Exploration with Expectation-Maximization," *Proceedings of the International Symposium on Robotics Research*, pp. 759–774, 2017.

[9] S. Bai, J. Wang, K. Doherty and B. Englot, "Inference-Enabled Information-Theoretic Exploration of Continuous Action Spaces," *Proceedings of the International Symposium on Robotics Research*, pp. 419–433, 2015.

[10] S. Bai, J. Wang, F. Chen and B. Englot, "Information-theoretic Exploration with Bayesian Optimization," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1816–1822, 2016.

[11] S. Bai, F. Chen and B. Englot, "Toward Autonomous Mapping and Exploration for Mobile Robots through Deep Supervised Learning," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2379–2384, 2017.

[12] F. Chen, S. Bai, T. Shan and B. Englot, "Self-Learning Exploration and Mapping for Mobile Robots via Deep Reinforcement Learning," *Proceedings of the AIAA SciTech Forum*, 2019.

[13] F. Niroui, K. Zhang, Z. Kashino and G. Nejat, "Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 4(2), pp. 610–617, 2019.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20(1), pp. 61–80, 2009.

[15] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell and P. Battaglia, "Graph Networks as Learnable Physics Engines for Inference and Control," *Proceedings of the 35th International Conference on Machine Learning*, pp. 4470–4479, 2018.

[16] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner and C. Gulcehre, "Relational Inductive Biases, Deep Learning, and Graph Networks," *arXiv preprint*, arXiv:1806.01261 [cs.LG], 2018.

[17] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vazquez, S. Savarese "A Behavioral Approach to Visual Navigation with Graph Localization Networks," *Proceedings of Robotics: Science and Systems*, 2019.

[18] T. Wang, R. Liao, J. Ba and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," *Proceedings of the International Conference on Learning Representations*, 2018.

[19] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," *Technical Report, Georgia Institute of Technology*, GT-RIM-CP&R-2012-002, 2012.

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard and F. Dellaert, "iSAM2: Incremental Smoothing and Mapping using the Bayes Tree," *The International Journal of Robotics Research*, vol. 31 (2), pp. 216–235, 2012.

[21] M. Kaess and F. Dellaert, "Covariance Recovery from a Square Root Information Matrix for Data Association," *Robotics and Autonomous Systems*, vol. 57(12), pp. 1198-1210, 2009.

[22] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," *Proceedings of the International Conference on Learning Representations*, 2017.

[23] Y. Li, D. Tarlow, M. Brockschmidt and R. Zemel, "Gated Graph Sequence Neural Networks," *Proceedings of the International Conference on Learning Representations*, 2016.

[24] H. Gao and S. Ji, "Graph U-Nets," *Proceedings of the International Conference on Learning Representations*, 2019.

[25] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 234-241, 2015.

[26] M. L. Puterman, "Model Formulation," *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, pp. 17–32, John Wiley & Sons, 1994.

[27] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," *Proceedings of the International Conference on Machine Learning*, pp. 1050-1059, 2016.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski and S. Petersen, "Human-level Control through Deep Reinforcement Learning," *Nature*, vol. 518(7540), pp. 529–533, 2015.

[29] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. v. Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp and R. Tsing, "Starcraft II: A New Challenge for Reinforcement Learning," *arXiv preprint*, arXiv:1708.04782 [cs.LG], 2017.

[30] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.