Explaining Graph-based Decision Learning for Autonomous Exploration in Unknown Environments*

1st Liping Chen
Computer science and technology
University of Science and Technology Beijing (USTB)
Beijing, China
M202210631@xs.ustb.edu.cn

2nd Shihong Duan
Computer science and technology
University of Science and Technology Beijing (USTB)
Beijing, China
duansh@ustb.edu.cn

3rd Cheng Xu
Computer science and technology
University of Science and Technology Beijing (USTB)
Beijing, China
xucheng@ustb.edu.cn

4th Heng Zhang
Computer science and technology
University of Science and Technology Beijing (USTB)
Beijing, China

5th Ran Wang
Computer science and technology
University of Science and Technology Beijing (USTB)
Beijing, China

Abstract—In unknown emergency environments, agents use ranging sensors for localization and mapping, needing to choose optimal decisions among numerous uncertain candidate target positions. Using exploration graphs to represent environmental information can significantly reduce the state space dimension. By employing deep reinforcement learning models with graph neural networks to predict actions, the agents can autonomously explore unknown environments with varying scales of landmarks. However, the strategy learning and prediction processes are opaque, making it difficult to understand and explain the temporal and spatial correlation of state transitions and the state-action relationships. This paper addresses this issue by proposing a multidimensional view visualization framework to analyze the correlation of high-dimensional state-action vector sequences during agent training and decision-making processes. A post-hoc explanation tool based on feature attribution is designed to create importance subgraphs, describing the causal effects and dependencies among nodes in the exploration graph, thereby explaining the agent's decision-making reasoning and behavior intentions. The visualization framework's effectiveness is validated in terms of high-dimensional vector sequence correlation, node uncertainty entropy reduction metrics, and performance factors of ranging navigation strategies.

Index Terms—Deep Reinforcement Learning, Graph Convolutional Network, Visualization, Explainability

I. Introduction

Localization and mapping in unknown environments are critical capabilities of autonomous mobile agents. The simultaneous localization and mapping (SLAM) algorithm is a widely used approach for building a map of an environment and estimating the agent's position within it, which is especially useful in dynamic and unstructured environments [1] [2]. However, the SLAM algorithm typically uses forward simulation to predict and select the best action based on the respective utility function. Therefore, the computation time is often costly, and may grow exponentially with the increasing dimension of the state space and action space, prohibiting real-time implementation [3]. The Exploration Graph (EG) provides a topological representation of environmental information, effectively reducing the dimensions of the state space and the navigation action space. To address the uncertainty of the agent's pose, a method combining Deep Reinforcement Learning (DRL) with Graph Neural Networks (GNN) is employed. This approach predicts the optimal exploration actions of the agent within a belief space [4], thereby offering a real-time and scalable decision-making process.

The goal of DRL is to train an autonomous agent to interact with a specific environment and make decisions using DNN as powerful function approximators to learn high-dimensional models from a large amount of data [5]. Although DRL demonstrates excellent performance, it is considered a black-box model due to its complex decision-making process and structure [6] [7]. In numerous fields, including medical and military, the inferences derived from these high-dimensional models are frequently challenging for users to comprehend and rely upon [8] [9]. Consequently, the explainability of DRL is a fundamental prerequisite for users to employ it in a safe and effective manner.

Regarding the autonomous navigation of agents, the baseline of DRL mainly adopts a visual game platform. Mnih et al. [10] employed the DQN (Deep Q-Network, DQN) algorithm and attained a favorable outcome in Atari

games. However, due to the lack of explainability, other DRL designers find it challenging to choose autonomous navigation hyperparameters for different scenarios [11]. Wang et al. [12] proposed a visualization analysis tool called DQNViz, which visualizes the dynamic decision-making process in DQN as a two-dimensional graph based on an Atari game. This tool is designed to help users understand the internal structure and decision-making process of DQN. However, further experimental support is needed to demonstrate the effectiveness of DQNViz in navigation tasks with continuous action spaces. Jaunet et al. [13] proposed a visual analysis tool, DRLViz, to assist users in understanding the decision-making and memory mechanisms in DRL. However, this memory mechanism is limited in its scope, targeting only specific network structures and lacking generalization. In order to advance beyond the current limitations of visualization, it is imperative to establish a general visualization platform.

In recent years, the GNN has been widely used in many fields for graph representation learning due to its excellent performance [14] [15], including natural language processing, biochemistry [16], traffic and weather prediction [17] and knowledge structure modeling [18]. The combination of DRL and Graph Convolutional Network (GCN) has become the choice of more and more researchers [4] [19]. However, due to the irregularity of the graph structure, traditional methods of interpreting Convolutional Neural Network (CNN) are difficult to apply to GCN. These methods include: i. Saliency-based methods. These methods highlight important regions in the input image that influence the model's predictions [20] [21]. ii. Perturbation-based Methods. These methods involve altering parts of the input and observing the changes in the output to infer the importance of the perturbed regions [22] [23]. iii. Feature Attribution Methods. These methods assign an importance score to each input feature [24] [25]. However, these interpretability methods struggle to elucidate the GCN prediction mechanisms during exploration in uncertain environments and to explain the temporal and spatial correlations between states and actions in DRL.

In this paper, we propose a platform for visualizing algorithmic process data, which enables researchers to store and analyze historical experimental data. The explainability of graph-based DRL algorithms and its application to mapping tasks is a rapidly evolving area of research. Consequently, we have initially applied the data of a mapping task to this platform. Furthermore, an explanation model was established to learn explanation subgraphs, which are utilized to answer questions like "Why the GCN model makes a certain prediction?". The primary contributions of our research are as follows:

1) A general algorithmic process data management platform: This platform enables the storage and visualization of historical experiments, supporting a wide range of algorithms. By providing robust data handling capabilities, it enhances the accessibility and usability of valuable historical information. In order to explain the agent's decision-making process, we use the method of Post-hoc analysis [26].
2) Explain the temporal and spatial correlation between states and actions: Active navigation in unknown environments requires the consideration of various contextual information, such as observed landmarks, candidate frontiers, the current and historical poses [27]. The information is presented in time dimension to show the temporal and spatial correlation between states and actions.
3) An explanation model applied to graph data: We applied our explanation model to the data from a mapping task, with the objective of identifying the critical subgraphs that effect decisions of the agent. The result revealed that the exploration strategy learned is consistent with that of most mammals.

The following content of this paper is organized as follows: Section II describes the design of platform, the temporal and spatial correlation theory and the explanation model. Section III shows experimental results and discussions. Section IV concludes the paper with a comprehensive summary.

II. Deep Reinforcement Learning Strategies Based on GCN

A. Formalization of Strategies

Graph-based deep reinforcement learning (G-DRL) consists of GCN and deep reinforcement learning algorithm. GCN predicts real-time actions in the belief space, making G-DRL suitable for autonomous mapping in unknown environments.

The Markov Decision Process (MDP) serves as a formal framework for DRL [28]. G-DRL, leveraging GCNs as the policy and value networks for navigating in unknown environments, is formally represented as G-MDP=$\{G, A, P_G, R, \gamma\}$. Here, G, A, and R respectively correspond to the state space, action space, and reward function. The state is defined as an exploration graph [4]. The exploration graph, representing the current state of the agent, contains observed landmarks, candidate frontiers, the current and historical poses. It is an efficient alternative to time-consuming simulations for measurements and map predictions, especially with increasing state and action dimensions. It is denoted as $g = (V, E)$, where V and E are the sets of vertices and edges. At each time step t, the agent observes a non-global state $g_t \in G$, adopts a policy π to select an action $a_t \in A$, receives a scalar reward r_t, and observes a new state $g_{t+1} \in G$. The cumulative reward R_t is computed as $R_t = \sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau$, with $\gamma \in [0, 1]$ as a discount factor. To facilitate state transitions, the GCN performs convolution operations on the exploration graph by considering information from neighbors. Specifically, at time step t, the state g_t can

be represented as a feature matrix, and GCN extracts features $g_t^{'}$ using Formula (1).

$$g_t^{'} = \sigma(D^{-\frac{1}{2}}\widetilde{A}D^{-\frac{1}{2}}g_t W) \tag{1}$$

where $\widetilde{A} = A+I$, I is the identity matrix, A is the weighted adjacency matrix, D is the degree matrix, W is a learnable weight matrix, $\sigma(\cdot)$ is the activation function Rectified Linear Unit (ReLU).

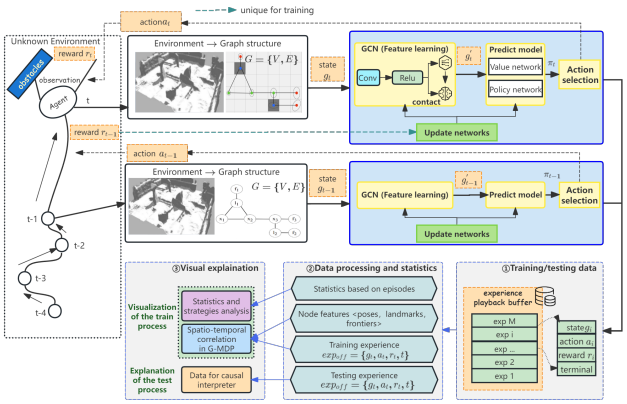

Fig. 1. Framework of Visualization Platform: The upper part shows the processing flow of G-DRL. The lower part shows how the data is presented, ①. An experience playback buffer is established for the purpose of storing the data generated by the algorithm. ②. The data is preprocessed and stored in a database, where exp_{off} stands for an offline experience. ③. The training data is employed for the purpose of visualisation, while the testing data is utilised to investigate subgraphs that influence decision-making once the algorithm has converged.

B. Algorithms

In this paper, we adopt both value-based method Deep Q-network (DQN) and policy-based method Advantage Actor-Critic (A2C). Value-based method strive to maximize the expectation or value of future returns.

$$Q_\theta(g_t, a_t) = E[\sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau | g = g_t, a = a_t] \tag{2}$$

To train Q_θ in Formula (2), the action sampling strategy is used to collect the transition samples of the form (g_t, a_t, r_t, g_{t+1}). The parameters θ are adjusted using stochastic gradient descent to minimize the loss function over sampled minibatch
$B \sim D = \{(g_t, a_t, r_t, g_{t+1})_t\}_{t \in T}$ (Formula (3)):

$$L_{DQN}(D) = E_{B\sim D}[(r_t + \gamma max Q(g_{t+1}, a_{t+1}) - Q(g_t, a_t))^2] \tag{3}$$

In addition to the value-based method, we also consider the policy-based algorithm A2C, which directly trains a parameterized strategy π_θ. We use two independent GCNs as the policy network and the value network. The loss function is defined as Formula (4):

$$L_{A2C}(D) = E_{B\sim D}[L_{A2C}^{(1)} + \eta L_{A2C}^{(2)}] \tag{4}$$

$$L_{A2C}^{(1)} = A(g_t, a_t)\log\pi(a_t|g_t) + \beta(V(g_t) - r_t - \gamma V(g_{t+1}))^2 \tag{5}$$

$$L_{A2C}^{(2)} = \sum \pi(a_t|g_t)\log\pi(a_t|g_t) \tag{6}$$

In Formula (4), Formula (5) and Formula (6), $L_{A2C}^{(1)}$ and $L_{A2C}^{(2)}$ represent the loss functions of a single transition sample. The function $A(g_t, a_t) = Q(g_t, a_t) - V(g_t)$ is called the advantage function, which calculates the difference between the state-action value function $Q(g_t, a_t)$ and the state value function $V(g_t)$. $\beta \in \mathbb{R}$ is a coefficient of the value loss and $\eta \in \mathbb{R}^+$ is the coefficient for the entropy of the output.

C. Visualization platform and explanatory model

As shown in the upper part of Figure 1, the processing flow of G-DRL mainly includes:

i. State extraction (environment → graph structure): Extract the observed environmental information of the agent, including node information and distance information, that is, the current state g_t.

ii. Feature extraction and transformation (graph structure → feature vectors): Extract graph feature vectors through a multi-layer neural network, that is, input the current state g_t into GCN.

iii. Action selection (feature vectors → action): Output the probability distribution of actions using the fully connected layer and softmax activation function or output the Q value using the fully connected layer.

The lower part of Figure 1 shows how the data obtained from the G-DRL process is presented: ①. An experience playback buffer is established for the purpose of storing the data generated by the algorithm. ②. The data is preprocessed and stored in a database, where exp_{off} stands for an offline experience. ③. The training data is employed for the purpose of visualisation, while the test data is utilised to investigate subgraphs that influence decision-making once the algorithm has converged.

Subsequently, we proceed to introduce the concepts of Spatio-temporal Correlation.

Spatio-temporal Correlation: There is a spatio-temporal correlation between the states and actions in G-MDP. This paper relates the state and action and explain the internal reasons for decision-making. The spatial correlation aims to reveal the relevant semantic features that affect the agent's decision-making, that is, where to see what kind of important information; The temporal correlation indicates the potential temporal dependence between adjacent observations and how the importance of features changes over time.

In order to confirm the subgraphs that affect the decision-making, the feature attribution method is adopted [29]. The testing data should be employed to identify explanatory subgraphs.

Figure 2 illustrates the relationship between the two models. The mapping task model is designated as the

target model, while the model for identifying subgraphs is referred to as the explanatory model.

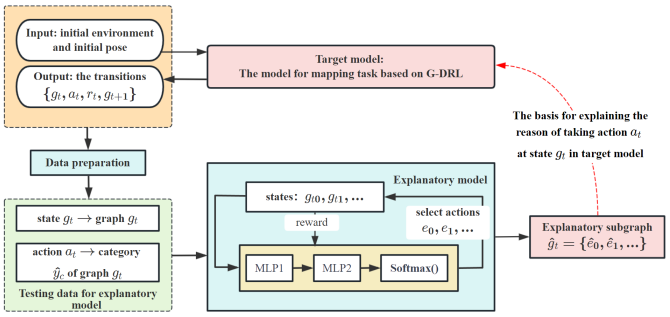

Fig. 2. Framework of the explanatory model and its Relationship with the target model

The feature attribution method confirms the explanatory subgraphs [29]. We frame the process as a Markov Decision Process (MDP) M $=\{S, A, P, R\}$. Herein, $S = s_k$ is the set of states abstracting edge sequences, and $A = e_k$ is the set of actions, which adds an edge to the current edge sequences at each time step. $P(s_k|s_{k-1}, e_k)$ is the transition dynamics about the new state s_k after edge addition e_k to the state s_{k-1}. $R(s_{k-1}, e_k)$ is used to calculate the reward after performing action e_k from state s_{k-1}. $(\emptyset, e_1, r_1, s_1, ..., e_k, r_k, s_k)$ describes the generation of edge sequences, where reward r_k reflects the causal effect of action e_k and coalition effect with the added edges previously [30]. For an exploration graph g_t and its predicted actions a_t, the detailed description and the relationship between G-MDP and M are as follows:

1) State space. At time step k, state s_k represents the subgraph g_{tk} composed of salient edges, where the initial state $s_0 = g_{t0} = \emptyset$.
2) The dynamic transition of states [31]. After the action e_k is made at time step k, the transition of state s_k is to merge the salient edge $\hat{e}_k$ into the previous state $s_{k-1} : g_{tk} = g_{t(k-1)} \cup \hat{e}_k$.
3) Action space. For state $s_{k-1} = g_{t(k-1)}$, the available action space A_k is the complement of $g_{t(k-1)}$, formally $A_K = g_t \backslash g_{t(k-1)}$. We select action e_k from A_k, choose the prominent edge $\hat{e}_k$, and connect it to the previous state $g_{t(k-1)}$ by adding an edge.
4) Reward design. As demonstrated in Formula (7), $\hat{y}_c$ is derived from the agent's action a_t in the context of autonomous exploration. $I(\cdot)$ computes the individual causal effect of the edge $\hat{e}_k$. The causal interpreter for feature attribution builds the explanatory model f rooted in reinforcement learning, depicted in Figure 2.

$$R(g_{t(k-1)}, e_k) = \begin{cases} I(e_k|g_{t(k-1)}, \hat{y}_c) + 1, if \ f_\theta(g_{t(k-1)} \cup \hat{e}_k) = \hat{y}_c \\ I(e_k|g_{t(k-1)}, \hat{y}_c) - 1, otherwise \end{cases} \tag{7}$$

TABLE I
HyperParameters of value-based algorithm DQN

Parameter	Value
EPISODE	$5e^3$
REPLAY_MEMORY	$1e^3$
OBSERVE	$5e^2$
BATCH	64
GAMMA	0.99
LEARN_RATE	$1e^{-5}$
EPSILON	$0.9 \sim 0$

III. Experiments and Analysis

A. Experiment Setup

The exploration strategy of this paper is trained in a 2D simulation environment, and the agent's task is to draw landmarks in a prior unknown environment in real-time accurately. The horizontal field of view of the agent is 360°(±0.5°), and the measurement range is 5m(±0.02m). During the exploration process, the agent can rotate from -180° to 180°(±0.2°), and each time step can move a distance of 2m(±0.1m). All simulated noise is Gaussian. Given the target frontier node, the agent will first turn to it and then drive directly toward it along a straight path.

We use an occupancy grid map to describe the environment. Each occupancy map is 40m×40m with randomly sampled landmarks and random initial agent location. In the experiment, the feature density of landmarks was $0.005/m^2$. To simplify the problem, the landmark is assumed to be passable, so obstacle avoidance is unnecessary. A virtual map consists of 2m×2m square map units, and each map unit contains a virtual landmark. Each virtual landmark's initial error covariance in the x-axis and y-axis is $1m^2$. This means that before starting the exploration, there may be a 1m error in the position of a virtual landmark on both the x-axis and y-axis. The initial error will be continuously corrected and updated in the subsequent exploration process. Once the agent observes 85% of the map, the task will be terminated.

Provide the following Hyperparameters setting for the DRL algorithms used in this paper:

In table I, EPISODE represents the complete training episodes. REPLAY_MEMORY represents the length of the experience playback buffer. OBSERVE describes the number of time steps between two updates of the target network. The minimum sampling strategy is used when updating the target network and the loss function. BATCH represents the number of sampled minibatch. GAMMA represents the discount factor. LEARN_RATE indicates the learning rate. EPSILON gradually changes during the training process, changing from the initial 0.9 to the final 0. This means that before MLP output, 90% of the data is first discarded, and as the training progresses, it gradually decreases to 0%. This means that the strategy provides greedy action selection.

TABLE II
Hyperparameters of policy-based algorithm A2C

Hyperparameter	Value
EPISODE	$5e^3$
NSTEP	40
GAMMA	0.99
LEARN_RATE	$1e^{-5}$
ENT_COEF	0.01
VF_COEF	0.25

Table II lists the Hyperparameters for the policy-based algorithm A2C. NSTEP means the number of time steps between two updates of the loss function. Generally speaking, A2C algorithm can be updated in a single time step, but updating every 40 time steps is adopted due to the significant value of EPISODE. In addition, ENT_COEF and VF_COEF are the coefficient for the entropy of the output and the coefficient of the value loss, respectively, corresponding to η in Formula (4) and β in Formula (5).

B. Multi-Views Visualization of Training Process

In G-MDP, there is a spatio-temporal correlation between the agent's states and actions. G-MDP generates M trajectories $exp_1, ..., exp_M$. Each trajectory is represented as $exp_i = \{g_t, a_t, r_t, t\}$, where g_t contains information such as h_t, x_t, l_t, f_t, and E_t. We can visualize the agent's exploration trajectory using every current pose $x_1, x_2, ..., x_t$. When focusing on a specific pose, like x_t, we can determine spatial associations, including landmarks l_t and frontiers f_t. Additionally, we can identify the corresponding action a_t, indicating the agent's movement towards the selected frontier. This paper introduces a trajectory view that captures temporal and spatial correlations within G-MDP, providing insights into the trajectory and state changes during exploration. Each node within g_t is represented as a vector containing $\{pos_t, ori_t, uc_t, inc_t, occ_t\}$, where pos_t and ori_t describe the node's position and orientation relative to the current pose, uc_t represents location uncertainty, inc_t is an indicator providing the type of the node, and occ_t is the occupancy of the map cell associated with the node.

The information on the map changes dynamically throughout the exploration process. Two performance indicators for the agent's exploration in the unknown environment are the exploration rate (RE) and the uncertainty matrix (U) of the map. The exploration rate RE is defined as the progress of exploring the unknown environment, denoted as $RE = \frac{S_{epd}}{S_{epd}+S_{upd}}$, where S_{epd} represents the set of explored cells, and S_{upd} represents the set of unexplored cells. Assuming the exploration map consists of $p \times q$ cells, the uncertainty matrix U of the map is defined by Formula (8).

$$U = \begin{bmatrix} u_{11} & \dots & u_{1q} \\ \vdots & \ddots & \vdots \\ u_{p1} & \dots & u_{pq} \end{bmatrix}_{p \times q} \tag{8}$$

The uncertainty of the cell in row m and column n can be defined as: $u_{mn} = \{cell_{mn} \in S_{epd} ? uc_{mn} : 0.5\}$. Cell uncertainty is determined by the uncertainty of the corresponding node. When cells are unexplored, their uncertainty is set to 0.5. An overview view is designed in this paper to visualize how the RE and U change with action selection. The following presents the design of multi-dimensional views.

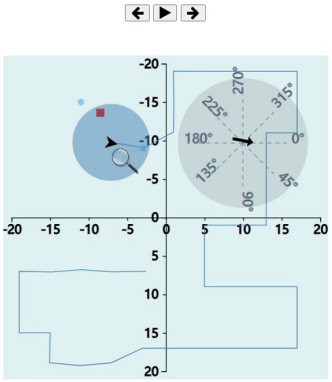

Fig. 3. Trajectory View

In Figure 3, the trajectory view consists of upper and lower parts. The three buttons in the upper part enable interaction with users, which can control the playback of the trajectory; The lower part of the trajectory view displays the mapping process, where the black arrow can be considered as the agent itself, the dark blue circle is the field of view of the agent. The blue dots are the frontier nodes. The agent selects one of the frontier nodes and follows it to draw the mapping trajectory (a blue line); the red squares are randomly generated landmark nodes on the map during initialization. At each identified landmark, the agent records its location on the map. Users can zoom in to see the agent's direction with a magnifying glass.

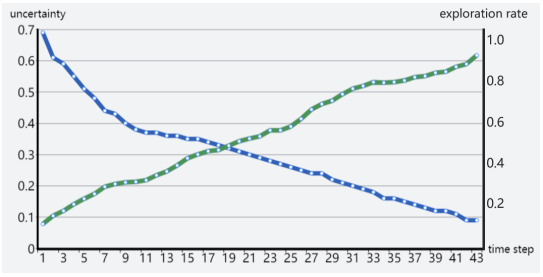

Fig. 4. Overview View

As shown in Figure 4, The overview view illustrates the evolution of the map through the uncertainty of cells

at each time step (the blue line). As time passes, it can be seen that the agent is exploring more and more area. The gradual decrease in uncertainty from left to right can be conceptualised as a process of exploring progress, corresponding to the trajectory view (Figure 3). The green line in Figure 4 displays the exploration rate of the map over time. The horizontal axis represents time and the vertical axis represents exploration rate. A mouse hover event will be triggered if the mouse hovers over the dot.

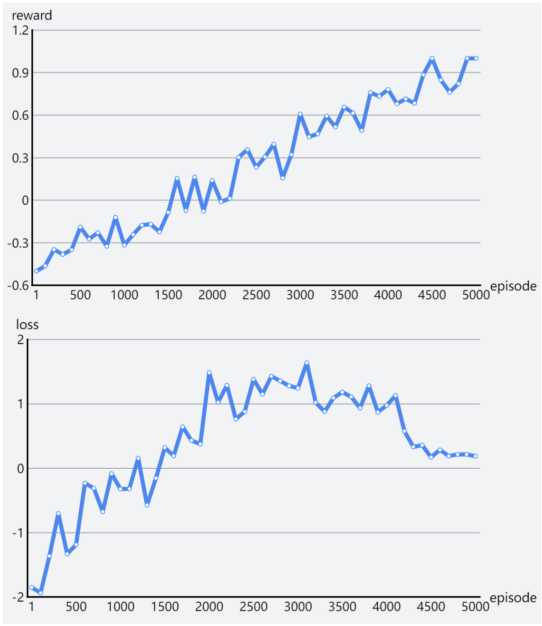

Fig. 5. Statistical View

The statistical view in Figure 5 shows two kinds of data, mainly including the statistics of reward and loss of an episode. The reward chart illustrates the reward of each episode. The loss chart shows the average loss of each episode.

Two evaluation indicators, the average error of landmarks and the entropy reduction of the map are employed to evaluate the algorithms [4]. The average error is calculated according to the actual locations and the predicted locations of the landmarks. Assuming that the actual location of a landmark is $[l_{x0}, l_{y0}]$ and the predicted location is $[l_{x1}, l_{y1}]$, then the error is defined as Formula (9):

$$error = \sqrt{(l_{x0} - l_{x1})^2 + (l_{y0} - l_{y1})^2} \tag{9}$$

$all\_error = error + (actual\ number\ of\ landmarks) - (number\ of\ observed\ landmarks)$. Calculating entropy reduction requires obtaining the uncertainty matrix of the map. The initial entropy of the map is a fixed value, $init = -(0.5 \times \ln 0.5) \times p \times q$, where 0.5 represents the initial uncertainty of a map cell. Update the entropy at each time step in the later. The entropy is defined as Formula (10):

$$entropy = -\sum_{m=1}^{p}\sum_{n=1}^{q}(u_{mn} \ln u_{mn}) \tag{10}$$

Finally, the difference with the initial entropy $entropy - init$ is calculated, which is the entropy reduction of the map.

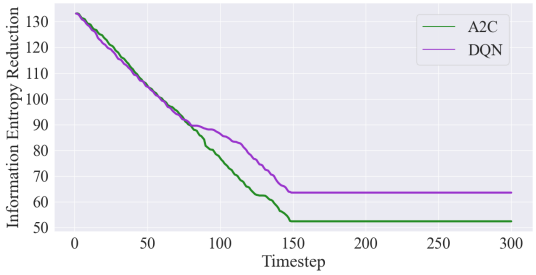

Fig. 6. Entropy Reduction of Map

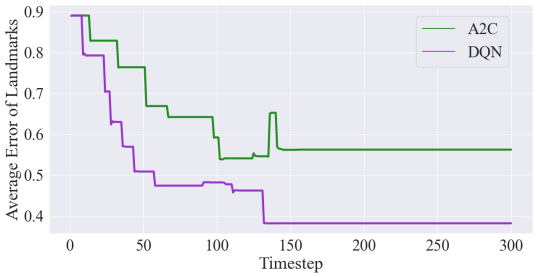

Fig. 7. Average Error of Landmarks

The green lines in Figure 6 and Figure 7 represent the test result of A2C, while the purple line represents the test result of DQN. It can be seen that the test result of A2C is better from Figure 6. What causes the poor test of DQN? It is not difficult to find that the agent still does not observe enough landmarks even when the training is near the end. Figure 9 shows the process data of episode4901 randomly selected in the visualization platform. It can be seen that the agent only finds six landmarks. However, in the final stage of the A2C algorithm, the agent found eight landmarks, as shown in Figure 8, which is the reason of entropy reduction of the map built by DQN is higher.

The average error of landmarks in Figure 7 shows that the DQN has a better test result. Since the calculation of the average error is related to the actual number of landmarks and the error between the actual and predicted positions of landmarks, it can also be seen that the number of landmarks observed in the DQN is less. Therefore, the rationale for this conclusion is that DQN is more accurate in predicting landmarks. So DQN is more suitable for the mapping task, even if not enough landmarks are observed.

At the same time, it was noticed that the average loss during the training process of DQN did not converge, and

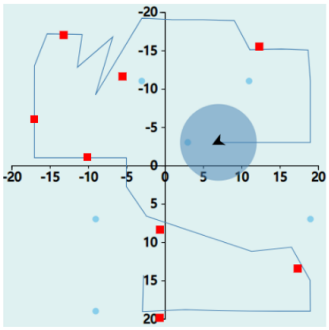

Fig. 8. Episode4882 (A2C)

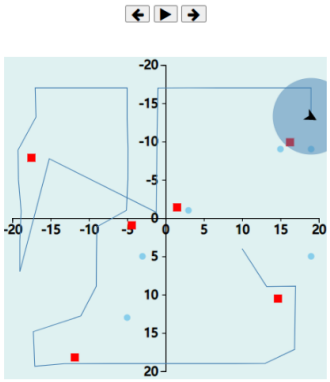

Fig. 9. Episode4901 (DQN)

the loss of episode5000 was even higher than episode1. The failure to converge the model probably led to the agent not observing enough landmarks, so we increased the training episodes of DQN and record the change of loss.

Increase the number of episodes to twelve thousand. The result is shown in Figure 10. It can be seen that the loss is in the increasing stage at episode5000 and converges around episode10000. This proves that increasing the number of training episodes can improve the result of DQN.

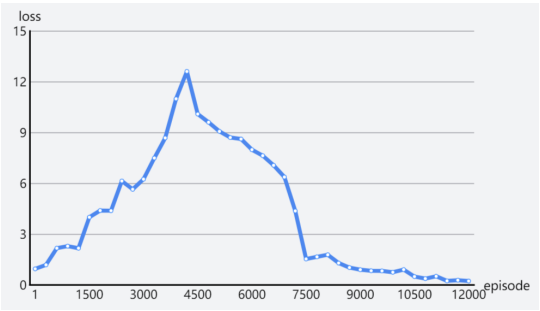

Fig. 10. Loss of DQN with Increased Episodes

In summary, this visualization platform can provide partial explanations for the testing results of the algorithms and identify the reasons.

C. Explanatory Subgraph

The explanatory model is applied to an episode of exploring the unknown environment to explain decision-making.

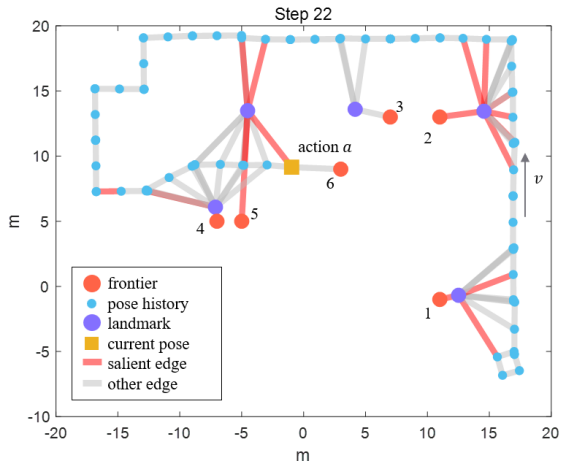

Fig. 11. Explanatory Subgraph

Applying the test process data oriented in unknown environment to the explanatory model, it is not difficult to find that the explanatory model can capture the edges that cause decision-making. For example, in the 22nd time step of the episode, as shown in Figure 11, when explaining the action (going to the frontier node with the number 6) selected by agent in this state, it assigns the most convincing edges (the salient red edges) with the most significant importance, including the edges connected to landmarks and the edges connected to its own position. The result shows the agent believes that the edges near the explored landmarks and its own position are more important for the current action selection, which is also in line with the habit of most mammals when exploring unknown environment. The explored area will not be considered in the current time step, and the agent is more inclined to go to the unexplored area. [32].

IV. Conclusion and Prospect

Agents commonly employ DRL methods to establish path planning decisions when exploring unknown environments. Given the black-box nature of the G-DRL model, this paper first examines the significance of visualization and interpretability techniques in achieving active path planning based on DRL method with exploration graph representation. Next, a visualization paltform is established, and an explanatory model for G-DRL is implemented. For the training process of G-DRL, multi-dimensional views are established to display the spatio-temporal correlated data in optimal navigation policy learning based on the exploration graph. For the testing

process, performance comparison is established to compare model accuracy and efficiency, also demonstrate the key parameters to optimize the model. An explanatory model is established for the G-DRL, showing explanatory subgraphs of feature attribution to explain decision-making rationale. Experimental results substantiate the effectiveness of visualization platform in explaining the decision-making, including process data correlation, models comparison, and explanatory subgraph to attribute decision-making rationale. In future research, we will further enhance the practicality of this visualization platform.

References

[1] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics 31, 5 (2015), 1147–1163.

[2] Junfu Qiao, Jinqin Guo, and Yongwei Li. 2024. Simultaneous localization and mapping (SLAM)-based robot localization and navigation algorithm. Applied Water Science 14, 7 (2024), 1–8.

[3] F. Chen, J. Wang, T. Shan, and B. Englot. 2019. Autonomous Exploration Under Uncertainty via Graph Convolutional Networks. (2019).

[4] Fanfei Chen, John D Martin, Yewei Huang, Jinkun Wang, and Brendan Englot. 2020. Autonomous exploration under uncertainty via deep reinforcement learning on graphs. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 6140–6147.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. nature 521, 7553 (2015), 436–444.

[6] Su Jiongming, Liu Hongfu, Xiang Fengtao, Wu Jianzhai, and Yuan Xingsheng. 2020. A survey of deep neural network interpretation methods. Computer Engineering 46, 9 (2020), 1–15.

[7] Buomsoo Kim, Jinsoo Park, and Jihae Suh. 2020. Transparency and accountability in AI decision support: Explaining and visualizing convolutional neural networks for text information. Decision Support Systems 134 (2020), 113302..

[8] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. 2017. Mdnet: A semantically and visually interpretable medical image diagnosis network. In Proceedings of the IEEE conference on computer vision and pattern recognition. 6428–6436.

[9] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. 2019. Causability and explainability of artificial intelligence in medicine. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9, 4 (2019), e1312.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. nature 518, 7540 (2015), 529–533.

[11] Thomas Hickling, Abdelhafid Zenati, Nabil Aouf, and Phillippa Spencer. 2023. Explainability in Deep Reinforcement Learning: A Review into Current Methods and Applications. Comput. Surveys 56, 5(2023), 1–35.

[12] Junpeng Wang, Liang Gou, Han-Wei Shen, and Hao Yang. 2018. Dqnviz: A visual analytics approach to understand deep q-networks. IEEE transactions on visualization and computer graphics 25, 1 (2018), 288–298.

[13] Theo Jaunet, Romain Vuillemot, and Christian Wolf. 2020. DRLViz: Understanding decisions and memory in deep reinforcement learning. In Computer Graphics Forum, Vol. 39. Wiley Online Library, 49–61.

[14] Han Zhang, Yu Hao, Xin Cao, Yixiang Fang, Won-Yong Shin, and Wei Wang. 2021. Relation prediction via graph neural network in heterogeneous information networks with missing type information. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2517–2526.

[15] Liyue Niu, Qiusheng Zheng, and Long Zhang. 2021. Enhance gated graph neural network with syntactic for sentiment analysis. In 2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA). IEEE, 1055–1060.

[16] Yang Li, Yang Yang, Qinghe Zheng, Yunxia Liu, Hongjun Wang, Shangling Song, and Penghui Zhao. 2024. Dynamical graph neural network with attention mechanism for epilepsy detection using single channel EEG. Medical & Biological Engineering & Computing 62, 1(2024), 307–326.

[17] Dan Xu. 2022. Research on Clustering Based on graph neural network. In CIBDA 2022; 3rd International Conference on Computer Information and Big Data Applications. VDE, 1–6.

[18] Weibo Gao, Qi Liu, Zhenya Huang, Yu Yin, Haoyang Bi, Mu-Chun Wang, Jianhui Ma, Shijin Wang, and Yu Su. 2021. RCD: Relation map driven cognitive diagnosis for intelligent education systems. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. 501–510.

[19] Lixiong Leng, Jingchen Li, Haobin Shi, and Yi'an Zhu. 2021. Graph convolutional network-based reinforcement learning for tasks offloading in multi-access edge computing. Multimedia Tools and Applications 80, 19 (2021), 29163–29175.

[20] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision. 618–626.

[21] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825 (2017).

[22] Smitha Milli, Pieter Abbeel, and Igor Mordatch. 2017. Interpretable and pedagogical examples. arXiv preprint arXiv:1711.00694 (2017).

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 1135–1144

[24] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In International conference on machine learning. PMLR, 3319–3328.

[25] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In International conference on machine learning. PMLR, 3145–3153.

[26] Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue 16, 3 (2018), 31–57.

[27] Jinkun Wang and Brendan Englot. 2020. Autonomous exploration with expectation-maximization. In Robotics Research: The 18th International Symposium ISRR. Springer, 759–774.

[28] Richard S Sutton and Andrew G Barto. 2018. Reinforcement learning: An introduction. MIT press.

[29] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2017. Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104 (2017).

[30] Xiang Wang, Yingxin Wu, An Zhang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2022. Reinforced causal explainer for graph neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[31] Wenjie Shi, Gao Huang, Shiji Song, and Cheng Wu. 2021. Temporalspatial causal interpretations for vision-based reinforcement learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 44, 12(2021), 10222–10235.

[32] Celeste Kidd and Benjamin Y Hayden. 2015. The psychology and neuroscience of curiosity. Neuron 88, 3 (2015), 449–460