IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 July 2023, accepted 6 August 2023, date of publication 17 August 2023, date of current version 24 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3305927

RESEARCH ARTICLE

Dealing With Sparse Rewards Using Graph Neural Networks

MATVEY GERASYOV[1,2] AND ILYA MAKAROV[2,3,4]

[1] School of Data Analysis and Artificial Intelligence, HSE University, 101000 Moscow, Russia
[2] Laboratory of Algorithms and Technologies for Network Analysis, HSE University, 603155 Nizhny Novgorod, Russia
[3] AI Center, National University of Science and Technology (NUST) MISiS, 119049 Moscow, Russia
[4] Artificial Intelligence Research Institute (AIRI), 105064 Moscow, Russia

Corresponding author: Ilya Makarov (makarov@airi.net)

This work was supported in part on Section 2 by the Strategic Project "Digital Business" within the framework of the Strategic Academic Leadership Program "Priority 2030" at the National University of Science and Technology (NUST) MISiS, in part by the Basic Research Program at the National Research University Higher School of Economics (HSE University), and in part by the Computational Resources of HPC Facilities at HSE University.

ABSTRACT Deep reinforcement learning in partially observable environments is a difficult task in itself and can be further complicated by a sparse reward signal. Most tasks involving navigation in three-dimensional environments provide the agent with minimal information. Typically, the agent receives a visual observation input from the environment and is rewarded once at the end of the episode. A good reward function could substantially improve the convergence of reinforcement learning algorithms for such tasks. The classic approach to increasing the density of the reward signal is to augment it with supplementary rewards. This technique is called reward shaping. In this study, we propose two modifications of one of the recent reward shaping methods based on graph convolutional networks: the first involving advanced aggregation functions, and the second utilizing the attention mechanism. We empirically validate the effectiveness of our solutions for the task of navigation in a 3D environment with sparse rewards. For the solution featuring the attention mechanism, we can also show that the learned attention is concentrated on edges corresponding to important transitions in the 3D environment.

INDEX TERMS Deep reinforcement learning (DRL), graph neural networks (GNNs), partially observable Markov decision process (POMDP), reward shaping.

I. INTRODUCTION

Reinforcement learning is a machine learning paradigm where an artificial agent learns the optimal behavior through interactions with a dynamic environment. Goals and purposes are explained to the agent via a scalar reward signal it receives after each interaction. Throughout the training process, the agent infers the behavior that maximizes cumulative reward, also called the return. To succeed in this task, the agent needs to explore the environment to understand which states and actions yield high rewards. On the other hand, the agent also has to exploit the rewards it has already received to adapt its behavior. This problem is known as the exploration and exploitation trade-off.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

Deep reinforcement learning (DRL) algorithms use neural networks to process large or continuous state spaces. The deep reinforcement learning approach has proven worthy in many dynamic tasks, such as machine translation [1], [2], [3], robotics [4], [5], [6], playing videogames [7], [8], [9], [10], [11], [12], [13], [14], and performing navigation in complex environments [15], [16], [17], [18], [19], [20]. In addition to these domains, deep reinforcement learning has demonstrated significant potential for solving real-world control problems, such as predictive aircraft maintenance [21] and traffic signal control [22], [23].

Navigating in three-dimensional environments can present a challenging problem for agents due to the sparsity of rewards. This problem arises when a scant number of states in the state space return a meaningful reward signal. A typical situation is when the agent must find a specific item or place

TABLE 1. Summary of notations used in the paper.

Notation	Description
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{R}$	Reward function
$s, s_t \in \mathcal{S}$	Current state
$s' \in \mathcal{S}$	Next state
$a, a_t \in \mathcal{A}$	Current action
r	Immediate reward
$\pi(a\|s)$	Policy
π^*	Optimal policy
γ	Discount factor
G_t	Discounted return at timestep t
$V(s)$	State-value function
$A(s,a)$	Advantage function
J_{actor}	Objective of the actor
$r_t(\theta)$	Probability ratio between the policies
$clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$	Clip $r_t(\theta)$ between $1-\epsilon$ and $1+\epsilon$
$\mathcal{R}'(s,a,s')$	Shaped reward function
$F(s,a,s')$	Shaping function
Φ	Scalar potential function defined on states
h	Node embeddings
$\mathcal{N}(i)$	Set of neighbors of node i
$\mathbb{S}$	Set of base case states
A	Adjacency matrix of the graph
X	Matrix of node features
$\mathcal{L}_0$	Cross entropy component of the loss
$\mathcal{L}_{prop}$	Message-passing component of the loss

in the environment and receives a positive reward only after reaching the destination. From the RL training procedure formulation, it naturally follows that one wants to reward the agent as often as possible. Hence, sparse rewards are detrimental to learning efficiency.

Throughout recent years several papers have addressed the sparse reward problem. Some notable approaches include reward shaping, Curiosity-Driven Methods [24], [25], [26], Curriculum Learning [27], [28], [29], Adaptive Skill Acquisition [30], [31], and learning with Auxiliary Tasks [32], [33], [34]. This study focuses on the potential-based reward shaping technique, as it is the most straightforward and intuitive way to deal with the sparse reward problem. This method is very flexible since it can be combined with most general-purpose RL algorithms.

This paper proposes a novel modification to a recently developed reward shaping technique based on the message-passing mechanism of graph convolutional networks [35]. Over recent years, graph neural networks have become increasingly popular and have found their application across various domains, including reinforcement learning [36], [37]. As a result, numerous graph neural network architectures have emerged, offering different benefits [38]. We show how selecting the appropriate architecture can notably increase the quality of the learned shaping function. For this purpose, we conduct several experiments using environments with sparse rewards from MiniWorld [16].

II. BACKGROUND AND MOTIVATION

Table 1 summarizes the parameters, variables, and functions used throughout this paper.

A. DEEP REINFORCEMENT LEARNING OVERVIEW

Markov decision process (MDP) is a standard model of agent-environment interaction. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a finite state space and $\mathcal{A}$ is a finite action space. $\mathcal{P}$ denotes a state transition function, giving the transition probability $p(s_{t+1} \mid s_t, a_t)$. Finally, $\mathcal{R}$ is a scalar reward function. A fundamental property of MDP is that the conditional probability distribution given by $\mathcal{P}$ depends only on the current state and does not depend on the history of the process. Discounted return is the sum of all rewards starting from state s_t multiplied by a discount factor $\gamma \in [0, 1)$:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{1}$$

where r_{t+k+1} is the reward recieved at timestep $t+k+1$.

In the partially observable MDP setting, the states are not entirely observable by the agent, introducing additional challenges to reinforcement learning algorithms.

At each step, an agent takes a decision according to a policy $\pi(a|s)$. The main goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected discounted return:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi[G_0] \tag{2}$$

Value function $V_\pi(s)$ is the expected discounted return conditional on the state of the environment:

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a)[r + \gamma V_\pi(s')] \tag{3}$$

where s' is the next state after state s, r and a are the reward and action at the current step respectively. $v_\pi(s)$ represents the value of following policy π from state s.

Policy-based DRL methods approximate the agent's policy with a neural network. One of the most famous policy-based methods is called advantage actor-critic [39]. The underlying neural network has two heads called actor and critic, respectively. The actor learns an optimal policy, and the critic learns the value function, which represents the quality of a given state. Thus, the information provided by the critic helps train the actor. The update rule for the actor reflects this fact:

$$\nabla_\theta J_{actor} = \frac{1}{N} \sum_{t=0}^{N} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A(s_t, a_t) \tag{4}$$

where θ denotes the parameters of the network, N is the number of steps in the trajectory, and A is the advantage of action a in state s. The advantage function represents the quality of a chosen action compared to the expected baseline, given by the value function. Thus, following the update rule given by (4), we aim to increase the probability of choosing beneficial actions with positive advantage values. One can estimate the advantage from a part of a trajectory (s, a, r, s') of an agent as follows:

$$\hat{A}(s,a) = r + \gamma V(s') - V(s) \tag{5}$$

Training the critic can be formulated as a regression problem with the following loss function:

$$L_{critic} = \frac{1}{N}\sum_{i=0}^{N}\sum_{s,a}(V_\theta(s) - [r + \gamma V(s')])^2 \quad (6)$$

One problem with this approach is that the value of the learning rate does not guarantee the degree of policy change. Small changes in the network parameters can lead to abrupt changes in the quality of a policy. The Proximal Policy Optimization (PPO) algorithm [40] addresses this issue by minimizing the following objective:

$$\hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \quad (7)$$

where $r_t(\theta)$ is the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and clip denotes clipping r_t, which removes the benefit of moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$.

B. REWARD SHAPING

The reward shaping framework aims to solve the sparse reward problem by augmenting the original reward function with a shaping function $F(s, a, s')$:

$$\mathcal{R}'(s, a, s') = \mathcal{R}(s, a, s') + F(s, a, s') \quad (8)$$

Shaping functions can be hand-crafted based on expert knowledge of the problem (e.g., Euclidean distance to the goal) or inferred during the training procedure [35], [41]. The necessary and sufficient condition for preserving the set of optimal policies of an MDP is for the shaping function to take the following form [42]:

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s) \quad (9)$$

where Φ is the scalar potential function defined on states. Designing a good potential-based shaping by hand can be a challenging task for some problems. Moreover, hand-crafted reward shapings often lack performance compared to automatically learned ones [35].

C. GRAPH NEURAL NETWORKS

A graph is a data structure representing a set of objects and relations between them. A graph G can be defined as a collection of nodes and edges connecting them $G = \{V, E\}$. Each node and edge of the graph can store additional information in the form of feature vectors. Graphs are an essential tool for modeling heterogeneously structured data, such as MDPs in reinforcement learning problems. Graph neural networks (GNNs) are deep learning models that allow for inference on graphs by leveraging local graph structure and node-level features. Graph convolutional networks (GCNs) are a special kind of GNN that implement a message-passing mechanism through the aggregation of neighboring nodes' features. In this section, we discuss commonly-used graph convolutional models.

The original Graph Convolutional Network [43] performs aggregation of neighboring nodes' features normalized by node degrees. The output of one convolutional layer of such a network can is defined as follows:

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j\in\mathcal{N}(i)}\frac{1}{c_{ji}}h_j^{(l)}W^{(l)}), \quad (10)$$

where $\mathcal{N}(i)$ is the set of neighbors of node i, $c_{ji} = \sqrt{|\mathcal{N}(j)|}\sqrt{|\mathcal{N}(i)|}$, W is a learnable weight matrix, l is the number of layer, and σ is a non-linear activation function.

Graph Attention Network (GAT) [44] adds the attention mechanism to GCN. GAT convolution aggregates node features of neighbors proportional to attention scores $a_{i,j}$:

$$h_i^{(l+1)} = \sigma(\sum_{j\in\mathcal{N}(i)} a_{i,j}W^{(l)}h_j^{(l)}); \quad (11)$$

$$a_{i,j}^{(l)} = softmax_i(e_{i,j}^{(l)}); \quad (12)$$

$$e_{i,j}^{(l)} = LeakyReLU(\vec{a}^{(l)^T}[Wh_i^{(l)}||Wh_j^{(l)}]), \quad (13)$$

where $||$ denotes concatenation and $\vec{a}^{(l)}$ is a learnable weight vector.

Finally, the GraphSAGE model [45] allows using different aggregators, such as mean, pooling, and LSTM [46]:

$$h_{\mathcal{N}(i)}^{(l+1)} = aggregate(\{h_j^l, \forall j \in \mathcal{N}(i)\}); \quad (14)$$

$$h_i^{(l+1)} = \sigma\left(W \cdot concat(h_i^l, h_{\mathcal{N}(i)}^{(l+1)})\right), \quad (15)$$

where *aggregate* is one of the aggregators from the list mentioned above and *concat* stays for the concatenation of node embeddings.

The key distinction among the outlined models lies in how they propagate messages between nodes to update their embeddings. Choosing an appropriate aggregating procedure can strongly affect the performance of graph convolutional networks. Next, we discuss learning an optimal potential-based reward shaping using graph convolutional networks.

D. REWARD PROPAGATION USING GRAPH CONVOLUTIONAL NETWORKS

In [35], the authors propose applying GCNs to a graph in which each state is a node and edges represent a possible transition between two states. Since there is no access to the complete underlying graph, it is approximated through sampled trajectories. The key idea of this approach is to propagate information about rewarding states through the message-passing mechanism implemented by GCNs. The probability distribution Φ_{GCN} at the output of the GCN is used as a potential function for potential-based reward shaping. To train the GCN, the authors use the following loss function:

$$\mathcal{L} = \mathcal{L}_0 + \eta\mathcal{L}_{prop} \quad (16)$$

$$\mathcal{L}_0 = \sum_{s\in\mathbb{S}} p(O \mid s)\log(\Phi_{GCN}(s)) \quad (17)$$

$$\mathcal{L}_{prop} = \sum_{v,w} A_{vw}\|\Phi_{GCN}(X_w) - \Phi_{GCN}(X_v)\|^2 \quad (18)$$

Here, $\mathbb{S}$ is the set of base case states, which consists of the first and last states of a trajectory and the states with non-zero rewards. A is the adjacency matrix, and X is the matrix of node features. $\mathcal{L}_0$ is the cross entropy loss between the labels of states from $\mathbb{S}$ and predictions of the GCN model. $\mathcal{L}_{prop}$ combines the neighboring messages through the graph Laplacian. η is the hyperparameter controlling the contribution of $\mathcal{L}_{prop}$ component to the whole loss. Lesser values of η lead to a more simple and biased model. Following the original paper, η is set to be equal to 10 in all the experiments.

III. RELATED WORK

A. APPLICATIONS OF DEEP REINFORCEMENT LEARNING

Deep reinforcement learning has been successfully applied to real-world problems in various domains. For example, in [47], the authors employed Deep Q-Learning [48] to predict lithium-ion battery capacity based on the permutation entropy of battery voltage sequences. Similarly, deep reinforcement learning was utilized for predictive aircraft maintenance [21]. The authors used a Soft-Actor-Critic [49] agent to decide when to schedule an engine replacement based on the estimates of Remaining-Useful-Life. Another domain where reinforcement learning has shown promising results is traffic signal control [22]. For instance, in [50], the authors adopted the distributed framework of Ape-X DQN [51] to learn a generalizable policy for operating a signalized intersection. In addition, deep reinforcement learning has been effectively applied to mobile robot navigation in indoor environments [6]. The authors trained an A3C [39] agent using only data from a 2D laser scanner and an RGB-D camera. These examples demonstrate the potential of deep reinforcement learning to improve decision-making in complex systems.

B. GRAPH CONVOLUTIONAL NETWORKS

Graph convolutional networks have been a significant development in the field of representation learning on graphs. Since their introduction [43], multiple modifications have emerged to enhance their performance. One area of research has focused on developing more effective methods of neighborhood aggregation. For instance, GAT [44] has added an attention mechanism to GCN, while GraphSAGE [45] adapted the aggregation process to incorporate advanced aggregators such as Long Short-Term Memory (LSTM) [46]. However, the choice of the optimal architecture heavily depends on the task at hand, and it remains an active research topic [52]. Model quality can be affected by various design choices, including the style of message passing, the number of message-passing layers, the dimensionality of embeddings, layer connectivity, and others [53]. In addition, different architectures vary in their expressive power [54]. Therefore, further research is necessary to discover better architectures suitable for emerging tasks, such as applying graph neural networks in reinforcement learning.

TABLE 2. CNN encoder architecture.

Layer	Number of filters	Kernel size	Stride	Activation
1	32	8	4	ReLU
2	64	4	2	ReLU
3	32	3	1	ReLU

IV. PROPOSED APPROACH

Following the reward propagation framework discussed previously, we propose using GAT and GraphSAGE models to propagate information about rewarding states. We use two-layered implementations of these models with 64 hidden units. The first layer of our GAT model has four attention heads and LeakyReLU activation function. As for the GraphSAGE model, we use mean aggregation and ReLU activation in the first layer. We chose the LSTM aggregator for the second layer since it is more appropriate considering the sequential nature of the data in reinforcement learning. We use an actor-critic network with a three-layered CNN encoder to model the policy of all agents. The architecture of the encoder is provided in Table 2.

The number of attention heads in the GAT model was tuned by comparing the results of multiple experiments. Other design choices highlighted in the previous section are consistent with the prior work [35] to make a fair comparison. We argue that the proposed models have an inductive bias which can help the models leverage the specific structure of the data in the problem at hand. The GraphSAGE model with the LSTM Aggregator can take advantage of the sequential nature of the data in the agent's trajectory. The GAT model can learn which transitions in the underlying MDP are relevant to the agent's task due to the attention mechanism. In addition, the GAT model provides interpretability as we can directly evaluate the learned attention scores. The training data for the models are samples of the underlying MDP graph represented by linear graphs of the agent's trajectories. This sampling strategy has been demonstrated to be a valid technique for training graph convolutional networks, as it does not result in a significant deterioration of model performance [35], [55].

The forward path in GCNs involves an aggregation of neighboring nodes' features. A simple approach to implementing the aggregation step is to use a matrix multiplication between the adjacency and feature matrices. However, this method has a high time complexity of $O(N^2F)$, where N represents the number of nodes and F is the dimensionality of node embeddings. It is possible to take advantage of the sparse nature of adjacency matrices in the problem at hand. By utilizing sparse operators, the time complexity of the aggregation can be reduced to linear with respect to the number of nodes [56]. This approach offers significant improvements, enabling efficient processing of large graphs. Thus, the time complexity of the forward path in GCNs is linear with respect to the number of nodes in the framework of the considered problem. The proposed models don't involve any expensive matrix operations such as inversion. They

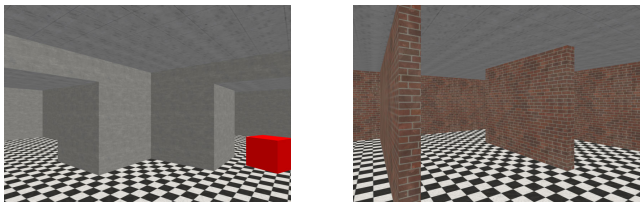

FIGURE 1. Screenshots of the FourRooms environment (left) and Maze environment (right).

only require a constant number of additional matrix multiplications per layer when compared to GCN used in prior work [35]. Given that all models in our experiments have the same number of layers and hidden units, the proposed approach can be considered comparable in computational time to the baseline architecture.

The PPO algorithm is used to update the policy in all experiments. The node features provided to GNNs come from the output of the CNN encoder of the actor-critic network. Finally, we compare our agents, denoted Φ_{GAT} and $\Phi_{GraphSAGE}$ respectively, with two baselines: Φ_{GCN} introduced in the original paper [35] and basic PPO without any reward shaping.

V. EXPERIMENT DESIGN AND RESULTS

We perform a series of experiments in MiniWorld [16] to test our approach. MiniWorld has several challenging three-dimensional POMDP environments. For our experiments, we select two environments with sparse rewards: FourRooms and Maze. Screenshots of both environments are shown in Figure 1. In the next two sections, we describe them in detail as well as the training procedure. We state our results in the final section of this chapter.

A. FourRooms

The player spawns at a random position inside four rooms connected by four openings. In order to get a reward, the player must reach the goal, represented by a red box. Furthermore, the position of the goal is also random for each episode. Also, there is a time limit to perform this task which is 250 steps. The player chooses one of three actions at each step: move forward, turn left, and turn right. The environment provides a positive reward only when the player succeeds. In this case, the reward is scaled down proportional to how long it took the player to reach the goal. The rest of the original rewards are zeros.

We train all neural networks on 16 parallel instances of the environment. We organize training based on the algorithm outlined in [35]. The node features used by the GNN come from the output of the CNN encoder of the actor-critic network. Each agent interacts with its environment for 128 steps. During this process, we record hidden states at the output of the CNN encoder and add all transitions (s_t, s_{t+1}) to the graph G_i, where i is the number of the environment. Then we apply reward shaping using the current potential function Φ and split the resulting sequence into four mini-batches. Finally, we use these mini-batches and PPO to update the policy. When environment i reaches the end of an episode, we use the recorded hidden states, the set of the base case states $\mathbb{S}_i$, and the graph G_i to update the potential function Φ at the output of the GNN model. Since an agent does not receive non-zero rewards until the end of an episode, $\mathbb{S}_i$ only consists of the first and last states of a trajectory. We repeat this update procedure until the total number of steps made by agents in all 16 environments exceeds 5 million.

B. MAZE

The player has to navigate to a goal through a procedurally generated maze. The player and the goal spawn randomly inside this maze, and the action space is the same as in the previous environment. The critical difference that makes this environment much harder than FourRooms is that the map is generated randomly at the beginning of each episode. The maze generation procedure begins at the top-left corner and utilizes a recursive backtracking algorithm to construct the maze. At each step, the algorithm randomly selects a neighboring room that hasn't been visited before and connects it to the current room. If there are no available unexplored rooms, the algorithm backtracks until it finds an unvisited room or returns to the starting position. This process generates a connected acyclic graph, ensuring that every room is reachable from any other room and that there is only one path between any two rooms. Subsequently, walls are placed between neighboring rooms that are found to be disconnected after completing the generation procedure. All walls inside the maze have the same color and texture. After the maze is generated, the goal and the agent are placed in random locations within randomly chosen rooms. The time limit for this environment is 216 steps, and rewards are assigned according to the same rule as in FourRooms. Altogether this makes the Maze environment a very challenging POMDP with sparse rewards.

The MiniWorld developers provide four versions of this environment, each with distinct size and movement characteristics. These versions include MazeS2, which is a small 2 by 2 maze, MazeS3, which is a medium-sized 3 by 3 maze, MazeS3Fast, which has increased turning and moving motion per action, making navigation easier for the agent, and Maze, the largest version with an 8 by 8 size. In this study, we use the MazeS3 version, which has standard movement and turning speeds and consists of 9 interconnected rooms. The training procedure is the same as described in the previous section, but since this environment is much more challenging, we extend the training duration to 20 million steps.

C. RESULTS

The average rewards of the agents during training are shown in Figure 2.

Here, we can immediately see that agents augmented with reward shaping learn faster than baseline PPO. In FourRooms experiments, after 5 million steps, all models converge to a

FIGURE 2. Learning curves for the FourRooms environment (top) and MazeS3 environment (bottom). Results are in the form of $mean(R) \pm std(R)$.

TABLE 3. Final performance for both environments. We convert all rewards to [0, 100] scale for better visibility. All results are shown in the form of $mean(R) \pm std(R)$.

Model	FourRooms	MazeS3
Φ_{GAT}	69.93 ± 5.72	76.56 ± 5.27
$\Phi_{GraphSAGE}$	69.91 ± 5.64	68.96 ± 10.32
Φ_{GCN}	65.28 ± 5.07	66.96 ± 9.55
PPO	66.79 ± 6.31	59.75 ± 4.42

very similar final performance, with Φ_{GAT} and $\Phi_{GraphSAGE}$ being marginally better. However, in the case of the MazeS3 environment, the difference between the models is more explicit. Also, the Φ_{GAT} model kept improving even at the end of the training, indicating that the result may be refined.

The final performance of all models is provided in Table 3. It is worth noting that Φ_{GAT} has the best mean final performance in both environments. Although, the difference is significant only in the case of a more challenging MazeS3.

Finally, we assess the quality of attention learned by the GAT model. Figure 3 demonstrates the distribution of attention values for the FourRooms environment.

We observe that the learned attention is largely focused on the edges at the beginning of the trajectory and before reaching the goal. Also, high attention is given to the transitions at the point in time when the goal (the red cube) enters the

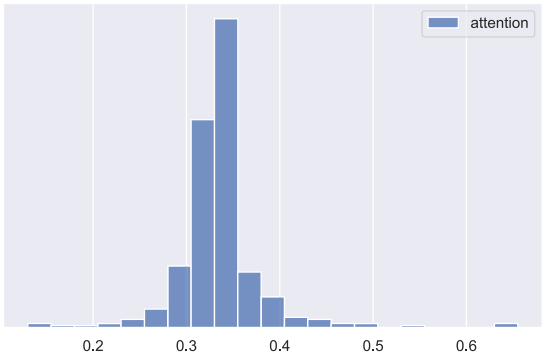

FIGURE 3. Histogram of the learned attention for the FourRooms environment.

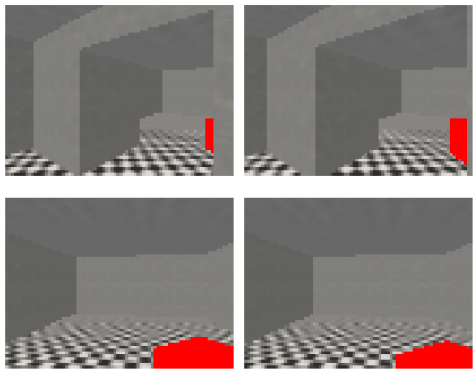

FIGURE 4. Pairs of screenshots corresponding to the transitions (left-to-right) that received high attention (samples from the top 5 percent of transitions).

agent's field of view (see Figure 4). This result, in particular, is unusual since the model did not receive any additional supervision to highlight such edges.

Hence, we can conclude that the GAT model learns to focus its attention on the transitions that are important for the agent's task.

VI. CONCLUSION

This study presented two modifications of one of the novel reward shaping techniques. Both our agents demonstrated better convergence speed and final results compared to the baselines. We also showed that the GAT model, which achieved the best final performance for both environments, was also able to learn meaningful attention relevant to the task performed by the agent.

In future work, it may be beneficial to incorporate edge-level or graph-level features. This would provide the graph neural network responsible for learning the potential function with additional information about the environment. Moreover, it may be valuable to explore a more complex design of a transition graph to better capture the structure of the underlying Markov decision process.

REFERENCES

[1] L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A study of reinforcement learning for neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Brussels, Belgium: Association for Computational Linguistics, Oct./Nov. 2018, pp. 3612–3621.

[2] H. Satija and J. Pineau, "Simultaneous machine translation using deep reinforcement learning," in *Proc. ICML Workshop Abstraction Reinforcement Learn.*, 2016.

[3] Y. Lee, J. Shin, and Y. Kim, "Simultaneous neural machine translation with a reinforced attention mechanism," *ETRI J.*, vol. 43, no. 5, pp. 775–786, Oct. 2021.

[4] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," 2018, *arXiv:1809.07731*.

[5] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile robot navigation based on deep reinforcement learning," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Jun. 2019, pp. 6174–6178.

[6] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments," 2020, *arXiv:2005.13857*.

[7] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," 2017, *arXiv:1710.02298*.

[8] M. Wydmuch, M. Kempka, and W. Jaśkowski, "ViZDoom competitions: Playing doom from pixels," 2018, *arXiv:1809.03470*.

[9] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[10] I. Makarov, A. Kashin, and A. Korinevskaya, "Learning to play pong video game via deep reinforcement learning: Tweaking deep q-networks versus episodic control," in *Proc. 6th Int. Conf. Anal. Images, Social Netw. Texts (AIST)*. Cham, Switzerland: Polytechnic University, Jul. 2017, pp. 236–241.

[11] I. Kamaldinov and I. Makarov, "Deep reinforcement learning methods in match-3 game," in *Proc. 8th Int. Conf. Anal. Images, Social Netw. Texts (AIST)*, in Lecture Notes in Computer Science, Kazan Federal University. Berlin, Germany: Springer, Jul. 2019, pp. 51–62.

[12] I. Kamaldinov and I. Makarov, "Deep reinforcement learning in match-3 game," in *Proc. IEEE Conf. Games (CoG)*. London, U.K.: Queen Mary Univ. of London. New York, NY, USA: IEEE, Aug. 2019, pp. 1–4.

[13] D. Akimov and I. Makarov, "Deep reinforcement learning with vizdoomfirst-person shooter," in *Proc. 5th Workshop Exp. Econ. Mach. Learn. (EEML)*. Cham, Switzerland: National Research University Higher School of Economics, Sep. 2019, pp. 3–17.

[14] M. Bakhanova and I. Makarov, "Deep reinforcement learning in VizDoom via DQN and actor-critic agents," in *Proc. 16th Int. Work-Conf. Artif. Neural Netw. (IWANN)*. Barcelona, Spain: Universitat Politecnica de Catalunya. Berlin, Germany: Springer, Jun. 2021, pp. 138–150.

[15] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen, "Deepmind lab," 2016, *arXiv:1612.03801*.

[16] M. Chevalier-Boisvert, "Miniworld: Minimalistic 3D environment for RL & robotics research," Farama Found., 2018. [Online]. Available: https://github.com/Farama-Foundation/Miniworld

[17] I. Makarov, M. Tokmakov, and L. Tokmakova, "Imitation of human behavior in 3D-shooter game," in *Proc. 4th Int. Conf. Anal. Images, Social Netw. Texts (AIST)*. Yekaterinburg, Russia: Ural Federal University. Cham, Switzerland: CEUR Workshop Proceedings, Apr. 2015, pp. 64–77.

[18] I. Makarov and P. Polyakov, "Smoothing Voronoi-based path with minimized length and visibility using composite Bezier curves," in *Proc. 5th Int. Conf. Anal. Images, Social Netw. Texts (AIST)*. Yekaterinburg, Russia: Ural Federal Univ. Cham, Switzerland: CEUR Workshop Proceedings, Apr. 2016, pp. 191–202.

[19] I. Makarov, P. Zyuzin, P. Polyakov, M. Tokmakov, O. Gerasimova, I. Guschenko-Cheverda, and M. Uriev, "Modelling human-like behavior through reward-based approach in a first-person shooter game," in *Proc. 3rd Workshop Exp. Econ. Mach. Learn. (EEML)*. Moscow, Russia: National Research University Higher School of Economics. Cham, Switzerland: CEUR Workshop Proceedings, Jul. 2016, pp. 24–33.

[20] I. Makarov, M. Tokmakov, P. Polyakov, P. Zyuzin, M. Martynov, O. Konoplya, G. Kuznetsov, I. Guschenko-Cheverda, M. Uriev, I. Mokeev, O. Gerasimova, L. Tokmakova, and A. Kosmachev, "First-person shooter game for virtual reality headset with advanced multi-agent intelligent system," in *Proc. 24th ACM Int. Conf. Multimedia (MM)*. New York, NY, USA: Univ. of Amsterdam, Oct. 2016, pp. 735–736.

[21] J. Lee and M. Mitici, "Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics," *Rel. Eng. Syst. Saf.*, vol. 230, Feb. 2023, Art. no. 108908.

[22] F. Rasheed, K. A. Yau, R. M. Noor, C. Wu, and Y.-C. Low, "Deep reinforcement learning for traffic signal control: A review," *IEEE Access*, vol. 8, pp. 208016–208044, 2020.

[23] Z. Li, C. Xu, and G. Zhang, "A deep reinforcement learning approach for traffic signal control optimization," 2021, *arXiv:2107.06115*.

[24] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," 2017, *arXiv:1705.05363*.

[25] B. Li, T. Lu, J. Li, N. Lu, Y. Cai, and S. Wang, "Curiosity-driven exploration for off-policy reinforcement learning methods," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2019, pp. 1109–1114.

[26] L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," in *Advances in Neural Information Processing Systems*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, 2021, pp. 3757–3769.

[27] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," 2020, *arXiv:2003.04960*.

[28] D. Zhang, W. Bao, W. Liang, G. Wu, and J. Cao, "A curriculum learning based multi-agent reinforcement learning method for realtime strategy game," in *Proc. 8th Int. Conf. Big Data Inf. Anal. (BigDIA)*, Aug. 2022, pp. 447–452.

[29] Y. Zhang, P. Abbeel, and L. Pinto, "Automatic curriculum learning through value disagreement," 2020, *arXiv:2006.09641*.

[30] J. Holas and I. Farkaš, "Adaptive skill acquisition in hierarchical reinforcement learning," in *Artificial Neural Networks and Machine Learning—ICANN 2020*, I. Farkaš, P. Masulli, and S. Wermter, Eds. Cham, Switzerland: Springer, 2020, pp. 383–394.

[31] J. Holas and I. Farkaš, "Advances in adaptive skill acquisition," in *Artificial Neural Networks and Machine Learning—ICANN 2021*, I. Farkaš, P. Masulli, S. Otte, and S. Wermter, Eds. Cham, Switzerland: Springer, 2021, pp. 650–661.

[32] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

[33] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Van de Wiele, V. Mnih, N. Heess, and J. Tobias Springenberg, "Learning by playing—Solving sparse reward tasks from scratch," 2018, *arXiv:1802.10567*.

[34] X. Lin, H. Baweja, G. Kantor, and D. Held, "Adaptive auxiliary task weighting for reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019.

[35] M. Klissarov and D. Precup, "Reward propagation using graph convolutional networks," 2020, *arXiv:2010.02474*.

[36] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li, "CoLight," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1913–1922.

[37] D. Gammelli, K. Yang, J. Harrison, F. Rodrigues, F. C. Pereira, and M. Pavone, "Graph neural network reinforcement learning for autonomous mobility-on-demand systems," 2021, *arXiv:2104.11434*.

[38] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[39] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016, *arXiv:1602.01783*.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[41] M. Grzes and D. Kudenko, "Online learning of shaping rewards in reinforcement learning," *Neural Netw., Off. J. Int. Neural Netw. Soc.*, vol. 23, pp. 541–550, May 2010.

[42] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. 16th Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, 1999, pp. 278–287.
[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, *arXiv:1609.02907*.
[44] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018, *arXiv:1710.10903*.
[45] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017, *arXiv:1706.02216*.
[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[47] A. Namdari, M. A. Samani, and T. S. Durrani, "Lithium-ion battery prognostics through reinforcement learning based on entropy measures," *Algorithms*, vol. 15, no. 11, p. 393, Oct. 2022.
[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[49] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*.
[50] G. Zheng, Y. Xiong, X. Zang, J. Feng, H. Wei, H. Zhang, Y. Li, K. Xu, and Z. Li, "Learning phase competition for traffic signal control," 2019, *arXiv:1905.04722*.
[51] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018, *arXiv:1803.00933*.
[52] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," 2022, *arXiv:2003.00982*.
[53] J. You, R. Ying, and J. Leskovec, "Design space for graph neural networks," 2021, *arXiv:2011.08843*.
[54] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.
[55] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 2006, pp. 631–636.
[56] D. Blakely, J. Lanchantin, and Y. Qi, "Time and space complexity of graph convolutional networks," Tech. Rep. Accessed: Dec. 31, 2021.

MATVEY GERASYOV received the bachelor's degree in computer science from the Moscow State Technical University of Civil Aviation, Moscow, Russia, and the master's degree in data science from the National Research University Higher School of Economics, Moscow, in 2022.

He is currently with the School of Data Analysis and Artificial Intelligence, National Research University Higher School of Economics, where he is also continuing his research. Author contribution: model code and experiments, and paper writing.

ILYA MAKAROV received the Specialist degree in mathematics from the Lomonosov Moscow State University, Moscow, Russia, and the Ph.D. degree in computer science from the University of Ljubljana, Ljubljana, Slovenia.

Since 2011, he has been a Lecturer with the School of Data Analysis and Artificial Intelligence, HSE University, where he was the School Deputy Head, from 2012 to 2016, and is currently an Associate Professor and a Senior Research Fellow. He was the Program Director of the BigData Academy MADE from VK, and a Researcher with the Samsung-PDMI Joint AI Center, St. Petersburg Department, V.A. Steklov Mathematical Institute, Russian Academy of Sciences, Saint Petersburg, Russia. He is also a Senior Research Fellow with the Artificial Intelligence Research Institute (AIRI), Moscow, where he leads the research in industrial AI. He became the Head of the AI Research Center and the Data Science Tech Master Program in NLP, National University of Science and Technology MISIS. Author contribution: paper revision, help with experiment and model design, and research supervision.