Unaliasing States in Homogeneous Environments

Philip Raeisghasem

Abstract—A significant part of reinforcement learning is ensuring adequate exploration. An agent cannot learn how good a new strategy is if the strategy is never tried. For multi-agent tasks, the space of possible configurations grows exponentially with the number of agents. This larger space is much harder to explore. In this paper, we apply the recently proposed Random Network Distillation exploration method to a multi-agent setting. We compare this approach to other exploration methods in the multi-agent setting. We do this both by explicitly quantifying amount of exploration and by analyzing downstream performance.

Index Terms—reinforcement learning, multi-agent, exploration, random network distillation.

1 Introduction

Reinforcement learning is a paradigm of machine learning alongside supervised and unsupervised learning. It is most applicable to problems that can be framed as an agent acting within an environment to achieve some goal. Like supervised learning, reinforcement learning happens through a feedback mechanism during an initial training phase. Unlike in supervised learning, this feedback is not in the form of 'labeled' data. There are no simple 'correct' answers to directly compare the model's performance against. Instead, the feedback loop in reinforcement learning is between the agent and the environment by way of a reward function. Broadly speaking, instead of learning whether its actions are 'right' or 'wrong', the agent learns how good its actions are.

This difference in feedback mechanism is important for allowing reinforcement learning models to learn from self-generated experience. Thanks to the advent of deep neural networks, the space of problems solvable by artificial intelligence and machine learning has exploded in size. Many of these problems, however, require much more data to solve. In supervised learning, vast datasets that require significant preprocessing are the norm, and a human is needed to label every single training example. In contrast, a human often need only specify the reward function for a given environment in reinforcement learning.

While for many problems it can be challenging to design a reward function that will lead to desired behavior, a large class of problems are amenable to very simple and sparse reward functions. If the problem is a two-player game, for example, the agent could only be rewarded at the very end, depending on whether it won the game. Using these sparse reward functions comes at the cost of the agent receiving less feedback. That is, there is a tradeoff between difficulty and human effort in embedding prior knowledge. This tradeoff is not unique to reinforcement learning, but it has given rise to unique methods in reinforcement learning for addressing the issue.

In particular, using a sparse reward function exacerbates the existing problem of exploration. While learning, an agent must always be deciding how confident it is in its current strategy. If it is very confident in the information it has received so far, it will choose to act in ways that have given it large rewards in the past. The agent is said to be

• *Philip Raeisghasem is with Louisiana Tech University, Department of Cyberspace Engineering.*
E-mail: par019@latech.edu

exploiting its current knowledge. If the agent is not very confident, it may decide to try something new in order to gather information. The agent is then said to be exploring the environment. A sparse reward function lessens the effectiveness of exploration by limiting the number of times an agent is given any feedback after exploratory actions.

The difficulty of exploration depends on more than just the sparsity of the reward function, of course. Other important factors are the size and complexity of the environment as well as the number of agents acting concurrently. This last factor, the number of agents, is especially impactful. An environment's reward is a function of the states occupied and the actions taken at any given time step. If there are $|S|$ states and $|A|$ actions available in an environment with n agents, then the input space of the reward function is of size $|S|^n|A|^n$. For problems with large state or action spaces, exploration in a multi-agent setting quickly becomes a daunting task.

In this paper, we investigate the exploration of a particularly challenging class of environments for which normal methods are inadequate. We develop new methods to help make these environments tractable, and we compare their effectiveness against an existing technique.

Namely, we compare our methods to exploration by random network distillation (RND) as formulated in [1]. This method is one way of encouraging an agent to seek out states that are "surprising". Roughly speaking, an intrinsic reward bonus is given to the agent for visiting states that it has visited few times before. This intrinsic reward, when combined with the (potentially sparse) extrinsic environment reward, has been shown to result in a more thorough exploration of the environment by a single agent.

We analyze the simulated data generated by multiple agents concurrently exploring with both methods. The environment we test with is a simple 2D environment. The metrics we gather include a direct measure of exploration as well as the total reward received over time. We also collect these metrics for other exploration methods for the sake of comparison.

2 Background and Review of Literature

2.1 Single-Agent Tasks

A reinforcement learning problem is formalized as a Markov Decision Process (MDP). Much of the notation below is taken from [2]. An MDP $\mathcal{M}$ is defined as a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$. Here, $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, and $\mathcal{R}$ is a set of rewards. The dynamics function $\mathcal{P} : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ assigns probabilities to states and rewards given an agent's action and previous state. It is generally stochastic and written as a conditional probability distribution $p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$ for all $s, s' \in \mathcal{S}, r \in \mathcal{R}$ and $a \in \mathcal{A}$. The discount rate, $\gamma \in [0, 1]$, is used to control how much the agent values immediate reward over future reward.

When in a given state s at time t, an agent seeks to act in a way that will maximize its expected sum of long-term, discounted rewards. That is, it attempts to maximize

$$\begin{aligned} G_t &\doteq \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] \end{aligned}$$

where G_t is called the return at time t. The expectation is taken with respect to both the dynamics $\mathcal{P}$ of the MDP and the policy of the agent. An agent's policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, often written as conditional distribution $\pi(a|s)$, assigns a probability to every action $a \in \mathcal{A}$ when in state $s \in \mathcal{S}$.

A generalization of the MDP is the Partially Observed Markov Decision Process (POMDP), wherein an agent doesn't necessarily receive complete information about the state of the environment. A POMDP is formally defined as a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \Omega, \mathcal{O}, \gamma)$. Here Ω is a set of observations and $\mathcal{O} : \Omega \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a conditional distribution $\mathcal{O}(o \mid s, a)$ over observations $o \in \Omega$ for given $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The agent's policy is then $\pi(a|o)$, a distribution over actions given the current

observation. When an agent is situated in some (for example) 2D or 3D space, an observation is often its "field of view". It may not see the whole environment from a single position.

2.2 Multi-Agent Tasks

The formalism for the extension of the MDP to the multi-agent setting is called a Markov game. Markov games were heavily inspired by similar constructions in game theory called stochastic games [3]. We consider a partially observed version of the Markov game. Similar to a POMDP, a Markov game for N agents is defined by a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \boldsymbol{\mathcal{P}}, \Omega, \mathcal{O}, \gamma)$. The main difference is in the form of the environment dynamics $\boldsymbol{\mathcal{P}}$. Here we have that $\boldsymbol{\mathcal{P}} : \mathcal{S} \times \mathcal{R}^N \times \mathcal{S} \times \mathcal{A}^N \rightarrow [0, 1]$. The state $\mathcal{S}$ now encapsulates state information for the environment and all agents. Each agent i receives its own reward $r_i \in \mathcal{R}$, takes its own observation $o_i \sim \mathcal{O}_i(o|s, \boldsymbol{a})$ and has its own policy $\pi_i(a_i|o_i)$, where $\boldsymbol{a} = \{a_1, \ldots, a_N\}$ is the joint action of all agents. Each agent seeks to maximize its own expected return

$$G_i = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{i,t+1}\right].$$

There are several ways to classify different instances of Markov games. Depending on the environment dynamics, a Markov game could be either cooperative, competitive, or somewhere in between. Classic competitive settings include two-player, zero-sum games like chess and go. A modern cooperative setting might be that of a traffic junction on a road.

Agents in a Markov game can be classified according to how much information is shared between them. On one extreme is the case of independent learners [4] [5] [6], when no information is shared at all and each agent regards the others as part of the environment. Another extreme is the case where each agent knows every other agent's model/policy parameters [7]. A common paradigm is to even have them share parameters.

Agents could also be classified by the mechanism by which they share information. They could have explicit communication channels [8] [9] [10], learn cooperatively at training time [11] [12] [13], or model each other through observation [14] [15] [16].

2.3 Deep Reinforcement Learning

While classical reinforcement learning algorithms do exist that can find optimal agent policies in a simple environment, modern problems are often too complex to be solved without the representational capacity and generality of neural networks. This complexity usually comes in the form of high dimensionality of the state space and possibly the action space. Learning optimal behavior for a robot given a video stream of input, for example, is far beyond the capabilities of classical algorithms.

Not until 2015 did anyone successfully scale reinforcement learning up to high-dimensional observations. In their paper [17], researchers from DeepMind successfully utilized the power of deep neural networks to play Atari video games directly from pixels on the screen. The weights within the neural network comprise the parameters of the agent's policy model. The techniques they employed are still being built upon today.

2.4 Exploration

One of the most central tasks of a reinforcement learning algorithm is to explore the policy space adequately enough to discover optimal behavior. In the single-agent setting, basic exploration strategies include occasionally taking random actions [18], using optimistic prior estimates for the value of all actions [19], and using state visit counts to measure uncertainty in action value estimates [20]. In many ways, these basic ideas remain state of the art for the general case. For example, count-based methods of measuring uncertainty have recently been extended to large and continuous state spaces [21] [22].

Several modern methods of exploration achieve diversity of experience through parallel execution of many agents at once. In [23], many non-interacting agents concurrently explore their environment (through occasional random

actions) and pool their experience to jointly learn a single policy. The same idea is applied in [24], except diversity of experience is achieved by each agent receiving unique priors and observation functions.

Instead of using many parallel single-agent environments, [25] utilizes a multi-agent setting to learn optimal single-agent policies. They drive exploration through population dynamics and the pressures of competition and resource scarcity. Somewhat similarly, a pair of adversarial agents are used in [26] to discover single-agent policies by rewarding one for "out-exploring" the other.

Much recent research has been done in the realm of intrinsic motivation. Agents that are intrinsically motivated attempt to optimize their own intrinsic rewards alongside the (extrinsic) rewards provided by the environment. Having intrinsic rewards is especially useful in environments with sparse extrinsic rewards. One form of intrinsic motivation is the idea of empowerment [27], a measure of the effect an agent has on its environment. Another is the idea of curiosity, wherein an agent is incentivized to seek novel observations. It could, for example, favor states it had not predicted well [28].

2.5 Random Network Distillation

We make use of one form of intrinsic motivation in this paper: exploration by random network distillation (RND) [1]. RND is a form of curiosity that rewards agents for receiving observations that they have not received many times before. It does this through the use of two auxiliary neural networks called the "target network" and the "predictor network". The target network is a fixed, random network that takes an agent's observations as input. The predictor network, which has the same architecture as the target network, is trained over time to predict the output of the random target network. The error between the outputs of the two auxiliary networks approaches zero for observations received many times. It is this error that is used as the agent's intrinsic reward at each time step.

RND was chosen in this paper for its simplicity and its effectiveness. Simple both conceptually and practically, it was a good candidate for use in a simple one-quarter project. Additionally, at the time of its publication, it was the state of the art exploration method by a large margin, by some measures. In particular, it was the first method to successfully complete the first level of Montezuma's Revenge, a famously difficult exploration Atari game, without learning from human examples or extra state information.

3 Problem Description

As pointed out in [25], though popular, curiosity-driven agents (among many others) still fail to solve a particular class of MDP. This class is that of environments that have very sparse rewards, are partially observed, and have local optima near the agent's initial state. Informally, this is like situating an agent in a large, dark, and mostly empty room. The agent is actually discouraged from exploring by the existence of a nearby reward, one that appears to be the only reward in the whole environment.

An illustration of the exact environment used in our experiments is shown in Fig. 1. The corresponding MDP has a finite state and action space. In the figure, colored squares are states with associated rewards. The actions available to the agent are "left", "right", "up", "down", and "stop". The agent receives zero reward until it stops, at which time it receives a reward proportional to the intensity of the color of square it's on. It is then reset to its initial position. The agent always starts in the center state and has a 3x3 observation window. Each agent is given 10,000 episodes of navigating and stopping during which to explore, learn, and optimize behavior.

Having most states map to the same featureless observation and having sparse rewards proves to be a difficult combination within an MDP. States that look the same due to partial observability are called "aliased" states. Most agents, including RND agents, quickly associate this common observation with zero reward and learn to stop immediately in the center to achieve the local optimum. The only way an agent can be successful is to somehow learn to differentiate between aliased states.

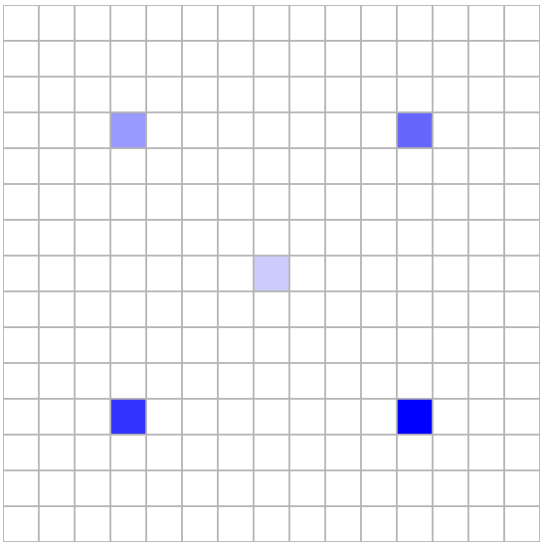

Fig. 1. A simple grid environment that discourages exploration.

4 Model Description

We introduce a possibly new method for differentiating between aliased states. As the agent moves, it changes the environment around it, leaving a trail of markers behind it to indicate where it has already been. The marker on each state decays over time.

This method is similar to using recurrent memory blocks, like an RNN or LSTM [29] block, within the agent's neural network. These blocks learn to maintain a useful state conditioned on past observations. Using this architecture, however, limits an agent to communicating only with its future self. Leaving environment trails also enables agents to communicate with each other.

As mentioned above, there are already several possible communication methods between agents within the same environment. However, traditional communication methods are all transient and do not carry temporal information without the aid of something like an LSTM block. The motivation of using environment trails is to achieve with fewer parameters the same effect of pairing communication with recurrent architectures. Having fewer parameters to learn also ideally reduces training time.

Two variants of this idea are tested. One, which we call "reward trails", directly affects the environment's reward function in states where a trail marker still exists. To promote exploration, these rewards are negative, thereby directly discouraging agents from stopping somewhere they have already been.

The other variant, coined "feature trails", does not directly affect the reward function. Different states are still un-aliased, but the agent must learn for itself through trial and error how much value to place on the newly un-aliased states.

Both variants are tried both with and without the addition of RND intrinsic motivation, making four different types of agents. Every agent type learns with the Q-learning algorithm [18] and uses a 2-layer, fully connected neural network as its value function approximator.

5 Data and Experimental Results

We ran single-agent experiments for all four agent types discussed above. For each type, we collected both raw state visit counts as well as statistics on its final total reward. When calculating total rewards for model comparisons, we did not include any self-imposed negative trail rewards. Possible environment rewards were 0, 2, 3, 4, 5, and 6, depending on where the agent stopped. The local maximum of 2 was placed in the center, starting position. Table 1 summarizes the results of our experiments.

Due to the nature of the problem, the data in Table 1 may seem contradictory. The sample variance of final returns seems to be too high for any of the differences between means to be significant. Indeed, z-tests on all pairs of means fails to find any significant $(\alpha = 0.05)$ difference between them. Specifically, the z-score used was

$$z = \frac{\bar{x} - \bar{y}}{\sqrt{S_x^2 + S_y^2}}$$

where

$$S_x = \sum_i \frac{(x_i - \bar{x})^2}{n - 1},$$

S_y is defined similarly, and $n = 40$.

	Feature Trails	Reward Trails	Feature + Curiosity	Reward + Curiosity
Sample Mean	3.03	3.49	2.51	3.75
Sample Variance	1.76	1.71	1.17	2.22
Success Rate	18/40	28/40	8/40	29/40

TABLE 1
Statistics over 40 runs of 10,000 steps each. The sample mean and variance are of the average return over the last 500 steps, and the success rate is of converging on an optimum other than the center one.

However, if we instead make our test statistic the empirical success rate of converging upon one of the optima not in the center, we do find some significant ($\alpha = 0.05$) differences. The reward trails always significantly outperform their feature trails counterparts. In these tests of differences between proportions, we have that

$$z = \frac{p_x - p_y}{SE}$$

where

$$SE = \sqrt{p(1-p)[(1/n_x) + (1/n_y)]}$$

is the standard error and

$$p = \frac{p_x n_x + p_y n_y}{n_x + n_y}$$

is the pooled sample proportion. We use the pooled sample proportion because the null hypothesis is that the two proportions are equal. Here, $n_x = n_y = 40$.

Surprisingly, raw state visitation counts didn't seem to vary that much between types of agents. This suggests that the difference in their performance is at least partly due to their relative ability to effectively retain and exploit information they have gained through exploration.

Not surprising is the relative success of reward trails over feature trails. The agent does not need to interpret the meaning of its own trails when they are directly encoded as rewards.

Curiosity in the form of an RND intrinsic reward actually proved to be a major hindrance when using feature trails. We hypothesize that this is because curiosity drives agents to the edges of the environment, which are inherently more surprising states than those in the center. States along edges are also easier to find than those that contained rewards. Our hypothesis is further supported by the fact that a purely curiosity-driven agent nearly always found the global optimum in a related problem where the rewards were in the corners.

6 Conclusion and Future Work

We have presented what we think is a novel method for successfully exploring POMDPs with a large number of aliased states and a local optimum that discourages exploration. All variants of our method were able to successfully escape the local optimum sometimes, whereas the baseline memory-less RND and Q-learning agents fail to do so at all.

The next step in assessing this exploration method in the single-agent case is to compare its performance and memory footprint with those of traditional recurrent neural network architectures. Also, testing the environment trail as a means of inter-agent communication requires running multi-agent experiments.

Beyond searching for the best architecture/method, a search also needs to be done for the best hyperparameters. In this paper, no hyperparameters were tuned once reasonable performance was achieved. Also, learning would almost definitely be accelerated and improved with a multi-step reinforcement learning algorithm. Q-learning is a simple but poor choice for environments with sparse reward functions.

Code for this project can be found at `github.com/philiptkd/stat_exploration`.

REFERENCES

[1] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2nd ed., 2018.

[3] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[4] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *CoRR*, vol. abs/1710.03748, 2017.

[5] J. Z. Leibo, V. F. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," *CoRR*, vol. abs/1702.03037, 2017.

[6] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *CoRR*, vol. abs/1511.08779, 2015.

[7] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," *CoRR*, vol. abs/1709.04326, 2017.

[8] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," *CoRR*, vol. abs/1605.07736, 2016.

[9] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *CoRR*, vol. abs/1605.06676, 2016.

[10] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," *CoRR*, vol. abs/1703.10069, 2017.

[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *CoRR*, vol. abs/1706.02275, 2017.

[12] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *CoRR*, vol. abs/1705.08926, 2017.

[13] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," *CoRR*, vol. abs/1803.11485, 2018.

[14] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," *CoRR*, vol. abs/1802.09640, 2018.

[15] Z. Hong, S. Su, T. Shann, Y. Chang, and C. Lee, "A deep policy inference q-network for multi-agent systems," *CoRR*, vol. abs/1712.07893, 2017.

[16] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine theory of mind," *CoRR*, vol. abs/1802.07740, 2018.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 EP –, Feb 2015.

[18] C. Watkins, "Learning from delayed rewards," 01 1989.

[19] R. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," 08 1996.

[20] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4 – 22, 1985.

[21] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, "#exploration: A study of count-based exploration for deep reinforcement learning," *CoRR*, vol. abs/1611.04717, 2016.

[22] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *CoRR*, vol. abs/1606.01868, 2016.

[23] 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," *CoRR*, vol. abs/1602.01783, 2016.

[24] M. Dimakopoulou, I. Osband, and B. V. Roy, "Scalable coordinated exploration in concurrent reinforcement learning," *CoRR*, vol. abs/1805.08948, 2018.

[25] J. Z. Leibo, J. Pérolat, E. Hughes, S. Wheelwright, A. H. Marblestone, E. A. Duéñez-Guzmán, P. Sunehag, I. Dunning, and T. Graepel, "Malthusian reinforcement learning," *CoRR*, vol. abs/1812.07019, 2018.

[26] S. Sukhbaatar, I. Kostrikov, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," *CoRR*, vol. abs/1703.05407, 2017.

[27] K. Gregor, D. J. Rezende, and D. Wierstra, "Variational intrinsic control," *CoRR*, vol. abs/1611.07507, 2016.

[28] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *CoRR*, vol. abs/1705.05363, 2017.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.