

Attention-based Curiosity in Multi-agent Reinforcement Learning Environments

1st Márton Szemenyei

Dept. of Control Engineering and Information Technology
Budapest University of Technology and Economics
Budapest, Hungary
<https://orcid.org/0000-0003-1397-6080>

2nd Patrik Reizinger

Dept. of Control Engineering and Information Technology
Budapest University of Technology and Economics
Budapest, Hungary
<https://orcid.org/0000-0001-9861-0293>

Abstract—Several paradigms exist in Reinforcement Learning to improve the exploration capabilities of agents, among which the curiosity-driven approach is followed in this work. Extending previous work that utilizes attention to make curiosity state- and action-selective, we expand the range of experiments by introducing two multi-agent environments. The first one is for robot soccer, the second one features a driving scenario in urban settings. Moreover, as during training the different number of observations must be matched between multiple time-steps, we propose an attention-based approach, called Recurrent Temporal Attention (RTA) to do this. The corresponding implementation can be found at <https://github.com/szemenyeim/DynEnv>.

Index Terms—Reinforcement Learning, curiosity, exploration, attention, multi-agent

I. INTRODUCTION

The Reinforcement Learning (RL) paradigm, i.e. to train agents based on interactions with the environment and collect feedback (reward or punishment), has not emerged until recently, although monographs like [1] exist since the 1990s. As a consequence of the rapid advancements of the technological background essential for creating successful RL solutions (such as GPUs and deep learning frameworks), more and more complex problems can be approached using this paradigm.

Modern RL has recently turned to address problems consisting of multiple agents, whose behavior must be cooperative to achieve their goals together with a team, but competitive to beat other teams present. The most recent breakthrough in this research area was undoubtedly the introduction of the AlphaStar agent of DeepMind [2], which was able to beat top-tier human players. However, in multi-agent scenarios the main challenges of RL, like the credit assignment problem or the compromise between exploration and exploitation are more difficult to resolve. In case of credit assignment, rewards are not only assigned between time-steps but also between agents as well. By proposing a counterfactual multi-agent approach, [3] addresses this issue.

The research reported in this paper was supported by the Higher Education Excellence Program in the frame of Artificial Intelligence research area of Budapest University of Technology and Economics (BME FIKP-MI/FM), by the ÚNKP-19-2 New National Excellence Program of the Ministry for Innovation and Technology and by the EFOP 3.6.1.-16-2016-00014 program (Research, development, and educational integration of disruptive technologies in the field of e-mobility).

The second, i.e. preferring exploration over exploitation (or vice versa), is the topic of this paper. Agents incentivized to exploit are usually not able to find the best solution, as they converge too quickly to a local optimum. On the other hand, overemphasizing exploration - although generally being able to find better policies for the agent - is not beneficial either, for the agent is unlikely to follow these good policies as it expects to improve upon them. The most common approach to improve the exploration capabilities of an agent is to use model-based solutions, such as the World model architecture [4]. Furthermore, constructing additional loss terms were proven to be effective for exploration. A pioneering approach introduced a curiosity-based module, called Intrinsic Curiosity Module (ICM) [5], which is used to construct an additional - intrinsic - reward based on the capability of the neural network to predict features and actions. A large-scale evaluation study of this method can be found in [6]. Recently, a disagreement-based approach was introduced as well to express curiosity using ensemble networks [7].

This paper extends the results of Reizinger and Szemenyei [8] by comparing the performance of attention-based curiosity formulation in multi-agent scenarios. The original work introduced different usages of attention [9] in the RL setting, including actor-critic networks, the ICM architecture or its loss function. In this paper, we introduce two new, OpenAI Gym-compatible [10], multi-agent RL environments, which are utilized to measure the performance of the algorithms of [8]. The environments support multiple observation types, including partial and noisy observations. To properly handle these difficulties, a new architecture is proposed including a novel model called Recurrent Temporal Attention (RTA), which is capable of merging objects sightings over multiple time-steps. Finally, the justification of the proposals of the previous work is extended, with emphasis on the attention-aided ICM loss, termed Rational Curiosity Module (RCM).

II. ENVIRONMENTS

In this section, the OpenAI Gym-compatible environments (Figure 1, Figure 2) used for evaluating the proposed methods are briefly presented. Both environments use 2D physics

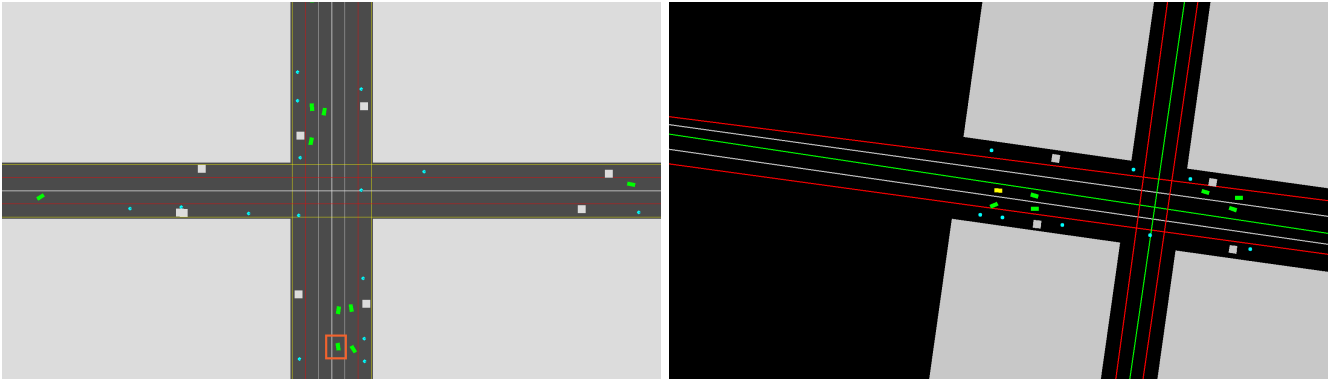


Fig. 1: The driving environment. Cars are green, pedestrians cyan and obstacles light gray. On the right, the noisy partial observation of the marked, yellow vehicle (indicated with an orange bounding box on the left image) is shown from bird’s eye view.

simulation provided the PyMunk engine¹: one environment simulates driving in urban settings, while the other is for robot soccer. The commonality of these two seemingly dissimilar tasks is that both require the algorithm to learn cooperation with team members in a competitive setting. To encourage cooperation and ease the credit assignment problem, rewards are provided both for individual agents and teams.

A. Mechanisms

The driving environment (Figure 1) implements a single urban intersection, where multiple agents (with a color of green) have to reach their own, randomly-assigned destination as quickly as possible without crashing or hitting pedestrians (marked as cyan). The environment also includes buildings and roadside obstacles (marked with light gray color on the road), which may block the view of other cars or pedestrians. Pedestrians mostly stay on the sidewalks, however, they may randomly decide to cross the road at any point.

Cars are assigned individual rewards for reaching their destination quickly and punishments for crashing. During the collision of two cars, the car that was in the opposite lane or moving towards the other car is punished more than a car that was in its own lane or moving away from the other. The negative reward for crashing is proportional to the car’s speed to encourage breaking. At the end of each episode all agents get a shared reward that is inversely proportional to the time it took for all cars to reach their destination.

In the robot soccer environment (Figure 2) the agent controls a single bipedal robot (yellow or magenta, respectively), with the goal of scoring in the opposing team’s goal (goalposts are blue in the figure). This environment uses 2D physics to simulate collisions and friction, but it abstracts over the complex physics of bipedal robots. Robots may step in four directions, turn, kick with either leg or turn their head. The first three of these actions might cause the robot to fall over in which case it gets deactivated while it stands up. Breaking the

rules (pushing, leaving the field, etc.) results in large negative rewards.

The robots are rewarded for scoring or moving the ball (with red color) towards the opponent’s goal. While the entire team receives rewards for these, the robots involved receive additional rewards. To achieve this, the environment keeps track of the last few agents touching the ball. However, this ignores agents that had the opportunity to prevent the goal but failed to do so. To account for this, agents of the losing team close to the ball also receive individual rewards. To encourage initial learning, the robots closest to the ball from each team receive rewards for moving toward the ball.

B. Observations

The physics of the environments are constructed to mimic real-world mechanisms as closely as possible. This means, that performing certain actions takes up considerable time, especially in the robot soccer environment. On the other hand, vision systems run considerably faster, meaning that for a single action, the environment responds with multiple consecutive observations. In both settings, the full state consists of the agent’s own status (including position, orientation, etc.) and the positions of all relevant non-stationary objects (including obstacles in the driving environment, since these are randomly placed).

In addition to this, the environments support *partial* observations, where all objects are transformed in the agent’s coordinate system, and objects falling outside the observable area are masked out. The observable area can be limited by view angle and a maximum range, while occlusion between certain object categories is also supported.

Random noise can be added to the observations as well, including additive noise on the object features (such as position, orientation, radius, etc.) as well as random false negative or false positive sightings. Optionally, the noise can be given more realistic characteristics. In this second case, the magnitude of noise and the probability of a false negative sighting is dependent on several factors, such as distance from the observer, partial sightings or proximity to other objects.

¹<https://www.pymunk.org>

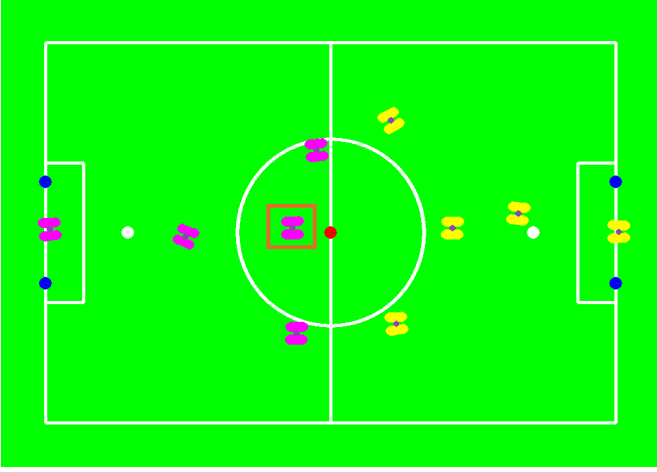


Fig. 2: The robot soccer environment with blue goalposts and a red ball and the corresponding noisy partial observation (right) of the marked agent (indicated with an orange bounding box). Multiple red circles in the observation image indicate false positives (highlighted with an orange bounding box).

In the realistic setting, misclassifications may also occur, especially between *similar* categories, such as penalty crosses and balls, or cars and obstacles. Moreover, false-positive sightings are more likely to occur in certain locations, such as false-positive balls are more likely to be seen at the feet of nearby robots. These additional heuristics are designed to model frequent failures of vision pipelines [11]. Examples of noisy observations can be seen in Figure 1 and Figure 2.

III. MODEL

In this section, the model used for the environments detailed above is presented. In our earlier work [8], the RCM architecture performed best, therefore it is used in this paper. It is important to note, however, that curiosity is an extension to existing RL architectures, it cannot be used just on its own: Most frequently, a variant of actor-critic networks is used to learn the policy. In this case, Advantage Actor-Critic (A2C) [12] was chosen as it uses a baseline for the prediction of the Q-function, thus it is able to better differentiate between actions, even in cases when only positive-valued rewards are present (which the standard actor-critic is not capable of).

The ICM module consists of two networks called forward and inverse dynamics, which are utilized to predict the next state and the action between states, respectively. As inputs, the forward model uses the state and action of the previous time-step, while the inverse model the states of two consecutive time-steps. In the original formulation, each of these features is concatenated and fed into the neural network, i.e.:

$$\hat{\phi}_{t+1} = \text{Forward}(\phi(s_t), a_t) \quad (1)$$

$$\hat{a}_t = \text{Inverse}(\phi(s_t), \phi(s_{t+1})), \quad (2)$$

, where $\phi(s_t)$ is the feature tensor, a_t the action at time-step t , while the subscripts *fwd* and *inv* stand for the forward and inverse models, respectively, whereas $\hat{\cdot}$ is used for predicted values.

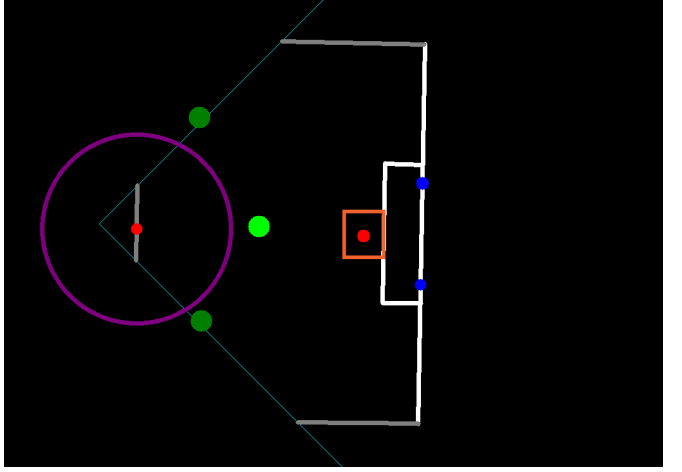


Fig. 3: Forward loss formulation of the RCM agent [8]

A. Rational curiosity

The most successful proposal of [8] is the Rational Curiosity Module (RCM), which utilizes attention on the forward loss of the ICM module. The motivation for this is that curiosity is not necessarily useful in every situation, for instance when the agent itself generates random (i.e. difficult to predict) dynamics, as it is the case in [6], where the agent had control over a noisy TV. Based on this, an adaptive approach was proposed, as shown in Figure 3, RCM weights the Mean Squared Error (MSE). In the original paper, this choice was justified by providing adaptive means to separate the state space based on the effectiveness of utilizing curiosity. In this work, a new approach is presented, namely, that of Weighted Least Squares (WLS). The reason for which is the fact that the MSE can be written as:

$$MSE = \frac{1}{N} e_{t+1}^T e_{t+1} \quad (3)$$

$$e_{t+1} = \sqrt{W} \left(\phi(s_{t+1}) - \hat{\phi}(s_{t+1}) \right), \quad (4)$$

where W denotes the weighting matrix of the *Attention* layer given $\phi(s_{t+1})$ (which is always non-negative as the SoftMax nonlinearity is used in an attention layer). Thus, RCM implements an adaptive version of the WLS method, making it possible to put weight to the forward loss only if necessary. We conjecture that this approach helps the agent to generalize better - in [8] it was shown that on all Atari games involved in the experiments this method performed significantly better

(in the statistical sense, using a 95% confidence interval with a paired Bayesian t-test).

B. Encoder

The encoder part of our model is based on the architecture proposed by Baker, Kanitscheider, Markov, *et al.* [13], since their model also had to handle a varying number of object sightings. Still, we expand this module further by adding an Recurrent Temporal Attention (RTA) module to embed temporal features of the observations. Our model is displayed in Figure 4.

The first step of the encoder architecture is to embed different object classes independently into an intermediary feature space. This step is needed to transform different object types into the same feature space so they could be concatenated. Since a single step of the environment returns multiple observations, the first part of the encoder creates a separate array of object embeddings for every time-step. These object sequences are propagated through a self-attention layer independently to allow the network to learn object relations. Since the network should be invariant to the order of the object sightings, no positional encoding is used.

It is desirable, however, to allow the network to consider relations between the sightings at different time-steps as well, since this might arguably lead to a more robust embedding. A straightforward way to do this would be to use a recurrent layer (such as Long Short-Term Memory (LSTM) [14]) to merge observations at different time-steps. However, recent studies have shown [9] that using attention is superior, especially with longer sequences. Moreover, attention is invariant to ordering, making it impossible for an attention layer to overfit to accidental features in the ordering of objects.

For these reasons, we use a module called RTA (shown in Figure 5), which is a multi-head attention layer [9] where the output sequence is fed back as input keys and values, and the sequence at the next time-step is used as query. The input keys and values are initialized with the sequence at time-step zero. To avoid loss of objects due to potential false negative sightings, the query sequence is padded to the same length as the keys. This ensures that the final output of the module is as long as the longest input sequence. The RTA module also outputs a confidence score for all objects in the final sequence to help mask out false positive sightings. Finally, the object sequence is averaged along the objects dimension using the confidence values as weights, resulting in a single embedding vector.

C. Temporal Entropy Regularization

Regularization is an essential part of training deep neural networks for which numerous methods exist. In the field of deep reinforcement learning, a common regularization method is rewarding the entropy of action distributions [12]. The reasoning behind this is that an agent with a low-entropy action distribution tends to strongly prefer a certain action over others, which is a sign of overfitting. Even more importantly,

randomly-initialized agents often implement degenerate policies, where the same action is chosen regardless of the input. Such bad initialization prevents the algorithm from exploring, and thus learning will certainly fail. Rewarding high-entropy action distributions, especially early in the training process alleviates this problem.

It is easy to see, however, that in both discussed environments, the optimal policy also has relatively low entropy. This is because in order to achieve good solutions in these scenarios, a single action has to be chosen over relatively long time periods, otherwise, the agents will 'jerk' in one place. In other words, the regularization method that prevents the agents from following degenerate policies also discourages them from following the optimal one.

Nonetheless, a relatively simple solution can be offered to this conundrum: our goal is to discourage agents from performing the same action regardless of the state without discouraging them to perform the same action in successive time-steps. To do that, the action probabilities are averaged along the batch dimension (the batch contains $n_{agent} \times n_{env}$ action distributions, where n_{agent} is the number of agents per environment, and n_{env} is the number of parallel environments), and the entropy is computed from that. This regularization term allows individual agents to have relatively low-entropy action distributions, as long as the other agents compensate for it, thus still preventing degenerate policies.

IV. RESULTS

The methods were evaluated on both environments using 8 parallel environments and a rollout size of 10. The robot soccer environment was trained for 30,000 updates (1,250 episodes), while the driving environment was trained for 60,000 updates (1,000 episodes). In both environments three agent types were evaluated: The standard ICM agent is used as a baseline, compared against ICM with Temporal Entropy Regularization (ICM-TER) and the RCM agent, also using TER.

We use an average reward metric to compare the agents' performance: first, the episode rewards are averaged for all parallel agents and environments (R_{avg}). Then, a 10-episode rolling average (R_{avg}^{10}) is computed. To put these reward values in context, two other baselines are presented: The first is a rough human baseline derived from rewards achieved by the authors, while the second is the reward achieved during random play.

The driving environment was run with two parallel agents, each assigned random initial and goal locations. In this case, the partial observation mode was used with small, realistic noise. The number and the location of pedestrians and obstacles were also randomly chosen for each episode.

The results (Figure 6 and Table I for this environment show that both our agents are clearly superior to the baseline, with the RCM agent also considerably outperforming the ICM-TER variant. Notably, while the 10-episode average reward of RCM is well below the human baseline, its best episode reward (which is still the average computed over 8 parallel environments with two agents each) managed to surpass it by

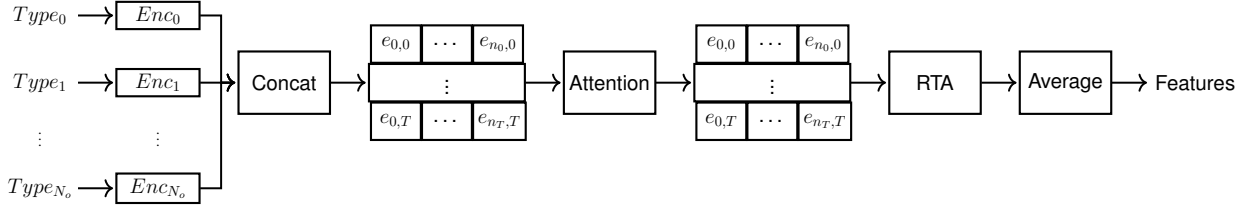


Fig. 4: The encoder architecture.

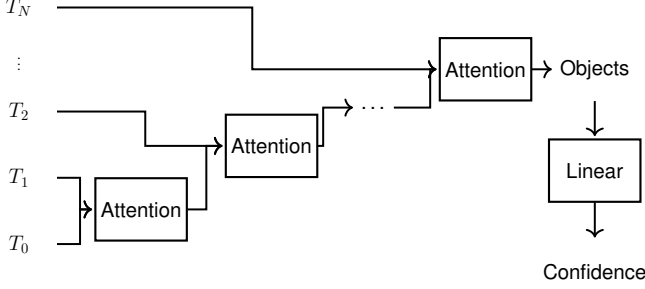


Fig. 5: The RTA module.

Model	Random	Human	ICM	ICM-TER	RCM
R_{avg}^{10}	0.7	42.3	16.1	19.5	24.6
R_{avg}	0.7	42.3	39.8	48.7	49.3

TABLE I: Best rewards achieved on the driving environment.

a significant margin. All three agents vastly outperform the random baseline.

The robot soccer environment was trained with 4 agents per environment (2 per team) with random, but legal starting positions. Here, the initial ball location was always the center point of the field. Here, we used the full observation mode, since partial sightings are far more limited in this setting, and arguably would slow down the learning process significantly.

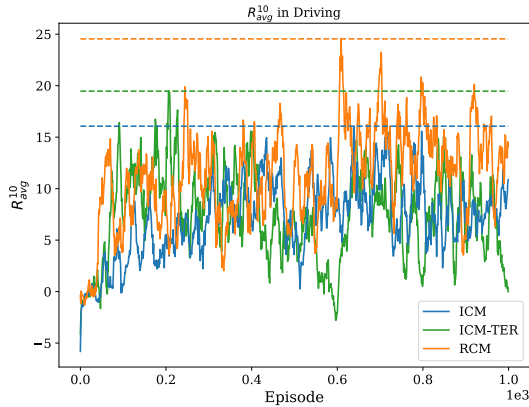


Fig. 6: Average rewards (R_{avg}^{10}) in the Driving environment. Dashed lines indicate the level of the corresponding maximum reward

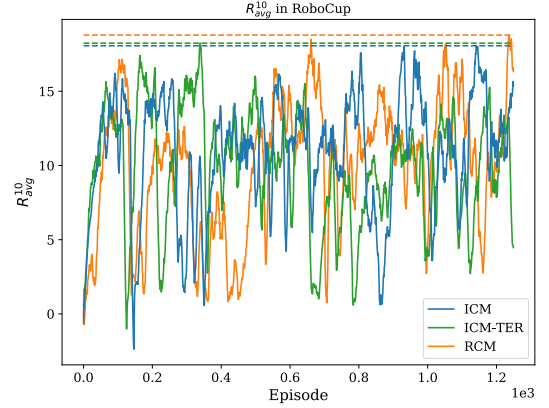


Fig. 7: Average rewards (R_{avg}^{10}) in the RoboCup environment. Dashed lines indicate the level of the corresponding maximum reward

Model	Random	Human	ICM	ICM-TER	RCM
R_{avg}^{10}	-0.8	30.5	18.1	18.2	18.8
R_{avg}	-0.8	30.5	22.4	22.2	24.3

TABLE II: Best rewards achieved on the robot soccer environment.

The results (Figure 7 and Table II) show that RCM once again outperforms the other two methods, albeit by a slightly smaller margin. In this case, the ICM-TER method does not yield a significant improvement over the baseline. The best rewards of RCM are well below the human baseline here, yet, all three models outperform the random baseline significantly once again.

V. CONCLUSION

In this paper, two novel Reinforcement Learning environments were presented for simultaneous multi-agent cooperation and competition. The first environment is a simulation of autonomous driving in urban environments, while the second is a humanoid robot soccer simulator. Both environments include physics and sensor simulation intended to emulate real-world challenges, such as noisy, false negative and false positive sightings. Most current RL environments [13], [15], [16] do not support noisy observations with false negative or positive sightings.

A novel model architecture capable of learning from such observations was also introduced. These models include the Recurrent Temporal Attention (RTA) module to combine ob-

servations from multiple steps, as well as Temporal Entropy Regularization (TER), which prevents degenerate policies without discouraging the agents from temporally consistent policies. The results of the training indicate that these contributions were successful, while they also corroborate the efficiency of the Rational Curiosity Model (RCM).

A significant unsolved problem is to reconstruct the entire world state from partial observations. By integrating our architecture and RCM with World models, we plan to use cooperating agents to create full reconstructions from partial observations. Moreover, we plan to extend the RCM model to predict features over longer windows, making the predictions usable for developing long-term strategies. The corresponding implementation of the environments and agents can be found at <https://github.com/szemenyeim/DynEnv>.

REFERENCES

- [1] R. S. Sutton, Ed., *Reinforcement Learning*. Springer US, 1992. DOI: 10.1007/978-1-4615-3618-5.
- [2] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, Oct. 2019. DOI: 10.1038/s41586-019-1724-z.
- [3] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” May 24, 2017. arXiv: <http://arxiv.org/abs/1705.08926v2> [cs.AI].
- [4] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 2450–2462. [Online]. Available: <http://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.pdf>.
- [5] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jul. 2017. DOI: 10.1109/cvprw.2017.70.
- [6] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, “Large-scale study of curiosity-driven learning,” Aug. 13, 2018. arXiv: <http://arxiv.org/abs/1808.04355v1> [cs.LG].
- [7] D. Pathak, D. Gandhi, and A. Gupta, “Self-supervised exploration via disagreement,” Jun. 10, 2019. arXiv: <http://arxiv.org/abs/1906.04161v1> [cs.LG].
- [8] P. Reizinger and M. Szemenyei, “Attention-based curiosity-driven exploration in deep reinforcement learning,” Oct. 23, 2019. arXiv: <http://arxiv.org/abs/1910.10840v1> [cs.LG].
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” Jun. 12, 2017. arXiv: <http://arxiv.org/abs/1706.03762v5> [cs.CL].
- [10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” Jun. 5, 2016. arXiv: <http://arxiv.org/abs/1606.01540v1> [cs.LG].
- [11] D. L. Buckman, Z. Tian, and D. D. Lee, “The upennalizers robocup standard platform league team description paper 2016,” 2016.
- [12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML’16, New York, NY, USA: JMLR.org, 2016, pp. 1928–1937. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3045390.3045594>.
- [13] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, “Emergent tool use from multi-agent autocurricula,” Sep. 17, 2019. arXiv: <http://arxiv.org/abs/1909.07528v1> [cs.LG].
- [14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.
- [15] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull, *Duckietown environments for openai gym*, <https://github.com/duckietown/gym-duckietown>, 2018.
- [16] N. G. Lopez, Y. L. E. Nuin, E. B. Moral, L. U. S. Juan, A. S. Rueda, V. M. Vilches, and R. Kojcev, “Gym-gazebo2, a toolkit for reinforcement learning using ros 2 and gazebo,” Mar. 14, 2019. arXiv: <http://arxiv.org/abs/1903.06278v2> [cs.RO].