frontiers | Frontiers in Neurorobotics

TYPE Original Research
PUBLISHED 14 October 2022
DOI 10.3389/fnbot.2022.932671

OPEN ACCESS

EDITED BY
Alexander N. Gorban,
University of Leicester,
United Kingdom

REVIEWED BY
Maryam Parsa,
George Mason University,
United States
Avinash Kumar Singh,
University of Technology
Sydney, Australia

*CORRESPONDENCE
Edward Staley
corban.rivera@jhuapl.edu

RECEIVED 30 April 2022
ACCEPTED 12 September 2022
PUBLISHED 14 October 2022

CITATION
Rivera C, Staley E and Llorens A (2022) Learning multi-agent cooperation. *Front. Neurorobot.* 16:932671. doi: 10.3389/fnbot.2022.932671

Learning multi-agent cooperation

Corban Rivera[1], Edward Staley[1]* and Ashley Llorens[2]

[1]Johns Hopkins Applied Physics Lab, Intelligent Systems Center, Laurel, MD, United States, [2]Microsoft Research, Microsoft, Redmond, WA, United States

Advances in reinforcement learning (RL) have resulted in recent breakthroughs in the application of artificial intelligence (AI) across many different domains. An emerging landscape of development environments is making powerful RL techniques more accessible for a growing community of researchers. However, most existing frameworks do not directly address the problem of learning in complex operating environments, such as dense urban settings or defense-related scenarios, that incorporate distributed, heterogeneous teams of agents. To help enable AI research for this important class of applications, we introduce the AI Arena: a scalable framework with flexible abstractions for associating agents with policies and policies with learning algorithms. Our results highlight the strengths of our approach, illustrate the importance of curriculum design, and measure the impact of multi-agent learning paradigms on the emergence of cooperation.

KEYWORDS

multi-agent, policy learning, reinforcement learning, artificial intelligence, learned cooperation

1. Introduction

Reinforcement learning (RL) offers a powerful approach to generating complex behaviors for intelligent systems that could not be explicitly derived or programmed. In the RL setting, the problem of learning an effective control policy is posed as a sequential decision-making problem for an agent interacting with a learning environment (Sutton and Barto, 2018). Given that learning the environment dynamics is an essential aspect of the RL problem, the ultimate effectiveness of a learned policy is dependent on the extent to which the learning environment reflects the essential aspects of the intended operating environment for the target system. Hence, many RL breakthroughs to date have focused on gaming and other applications with structured and predictable environments (Silver et al., 2016; Brown and Sandholm, 2019; Vinyals et al., 2019).

Translating progress in RL to increasingly complex applications of artificial intelligence (AI) will require the design of representative learning environments with corresponding complexity. Ensuring that future progress is reproducible and accessible for a broad community of researchers will require tools and frameworks that enable RL solutions to gracefully scale to address the problem of learning effectively in these increasingly complex settings. In general, RL frameworks must balance multiple tradeoffs, including ease of prototyping vs. training at scale, high-level abstractions vs. fine-grained control, and richness of features vs. ease of use.

2. Related work

Advances in the field of reinforcement learning has resulted in astonishing progress in the areas of robotic control (Lillicrap et al., 2015; Levine et al., 2016), and the ability to master challenging games (Mnih et al., 2015; Silver et al., 2016). To facilitate advancement in the field, numerous reinforcement learning frameworks have been developed to address scalable training (Caspi et al., 2017; Guadarrama et al., 2018; Schaarschmidt et al., 2018), reproducibility (Loon et al., 2019), robotics interoperability (Fan et al., 2018), lifelong learning (Fendley et al., 2022), ease of prototyping (Abel, 2019; Stooke and Abbeel, 2019; D'Eramo et al., 2020). Despite these advancements, these works are designed for the single-agent setting.

Our aim is to help enable RL research for the class of applications that involve multiple teams of agents where each team may have unique learning strategies and where agents within a given team may have localized views of the environment. Distributed multi-agent applications may be thought of as analogous to "system of systems" applications from a systems engineering perspective where collections (teams) of goal-oriented systems (agents) collaborate to achieve shared objectives. These attributes may arise, for example, in smart city applications where automated traffic control systems interact with fleets of automated vehicles (Shalev-Shwartz et al., 2016) or in defense applications (Cai et al., 2013) where heterogeneous autonomous systems interact across time and space to achieve high-level mission objectives. Applications such as these often include cooperation or competition (Busoniu et al., 2008) among heterogeneous teams of agents as defining features.

Progress in the area of multi-agent reinforcement learning has been made through the development of novel algorithms (Lowe et al., 2017; Rashid et al., 2018; Son et al., 2019) and frameworks to support distributed training of multi-agent policies (Zheng et al., 2017; Juliani et al., 2018; Liang et al., 2018). While these multi-agent focused frameworks make significant contributions to the field, these frameworks are designed to train with a single learning algorithm. Reinforcement learning algorithms have unique strengths and properties that make them ideal for different scenarios. Some of these properties include sample efficiency (Mnih et al., 2013; Haarnoja et al., 2018), shared value functions (Lowe et al., 2017), intrinsic curiosity (Haarnoja et al., 2018). A single learning algorithm may not be ideal for training the policies of all agents in a complex environment. For example, in environments where agents operate at different timescales, the agent interacting with the environment at the slowest timescale will collect the least experience. In these cases, sample efficient learning algorithms may be needed (Mnih et al., 2013; Haarnoja et al., 2018). For agents that rapidly interact with the environment, on-policy algorithms may achieve a desired level of average reward more quickly (Schulman et al., 2017). In the following section, we describe how our contributions address this important limitation.

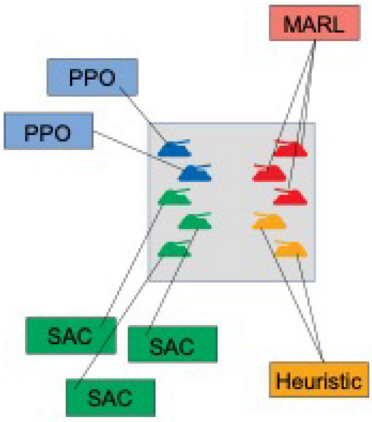

FIGURE 1
Simultaneous multi-agent multi-strategy policy learning. The figure illustrates an example multi-agent setting where the policies of individual agents are learned simultaneously with different learning strategies.

3. Introduction to the AI arena

In this work, we introduce the AI Arena: a scalable framework with flexible abstractions for distributed multi-agent reinforcement learning. A key contribution of the AI Arena framework is the introduction of abstractions to flexibly associate agents with policies and policies with learning algorithms or heuristics. Figure 1 illustrates an example environment with flexible associations between agents, policies, and learning algorithms. The framework naturally distributes experience gathering over multiple nodes and routes those experiences to the associated learning algorithms to update policies.

3.1. Multiagent environments

One primary goal of the AI Arena interface is to encourage environments in which a variety of agents may coexist and learn together. This should encompass everything from collaboration among identical entities to competition among several dissimilar groups. To that end, other properties of the environment are also converted to lists, such as action spaces or observation spaces. This allows for a variety of agent types to coexist in a single environment. For example, one learning entity may be making discrete decisions about image state data, while another entity in

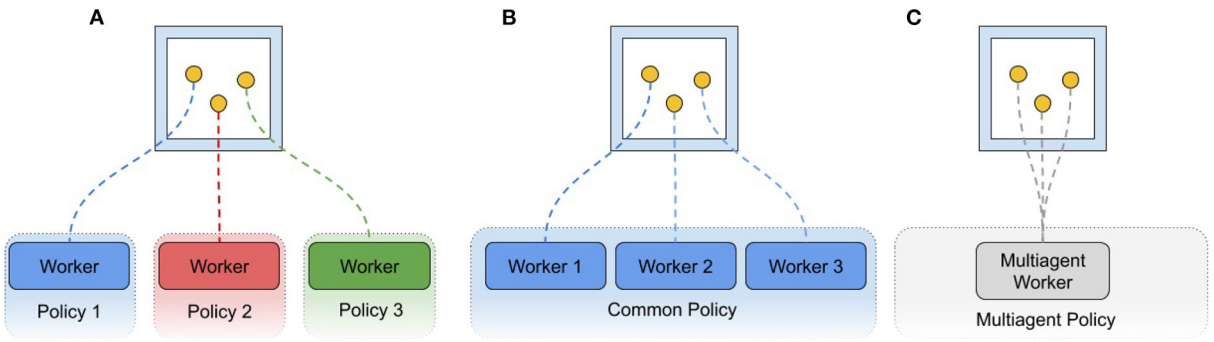

FIGURE 2
Possible worker configurations. Policy workers may be attached in environment entities in any desired combination. **(A)** Three entities are assigned to three independent policies that are learning separately and may have workers attached to other environments. **(B)** Entities are each attached to workers of the same policy, such that some of the agents contributing to the policy are coexisting in the same environment. **(C)** Entities are all attached to the same policy worker, which takes all of their data into account in a multiagent manner, possibly one of many workers for a distributed multiagent algorithm.

FIGURE 3
TanksWorld Multi-Agent Environment for AI Safety Research. These images illustrate different views of the Tanksworld environment **(Left)** is a birds-eye rendering of the environment, **(Center)** is an agent's-view rendering, and **(Right)** is the state representation actually provided as observations to the RL algorithm.

the same environment may expect continuous actions based on a vectorized state space.

An implication of this multi-entity setup is that all entities, as well as their actions, observations, and rewards, are occurring in lock-step at the same rate. Each step, the environment expects decisions corresponding to all entities, and will return information to all of them about the consequences of those decisions. While this may seem limiting at first, it is better to think of this as supporting the most extreme case of multi-agent interaction: all entities can be involved in a single frame of the environment. It is fairly straightforward to embed special cases within this framework: an entity which has exited an episode early can send and receive null values, or an entity with a lower interaction frequency can easily on every Nth observation and repeat actions until that observation occurs. The global "done" signal is especially useful for simulations and games in which there is a common or mutually exclusive objective, as is often the case.

3.2. Multiple learning policies

A further goal of the AI Arena interface is to enable complex distributed training architectures in which many policies may be training simultaneously in shared environments. The policies may be several instances of the same algorithm or be entirely separate approaches to learning. The inclusion of many entities in a single environment breaks from a typical training paradigm of one policy-worker thread corresponding to one agent in one environment. Rather, it is up to the user of this interface to distribute the many entities in an environment (or across many environments in the distributed case) to as many agents as desired. For example, an environment with N agents may function as N workers to a single distributed algorithm, or on the other extreme, single workers to N distinct policies. They may also be grouped such that M agents are in fact controlled by a single instance of a multi-agent policy (Figure 2). In other words, the agency of a given entity is at the discretion of the user.

```
# define policies. 2-5 reference frozen PPO
# policies that have been trained previously
policies = {1:"ppo", 2:"static", 3:"random",
    4:"passive", 5:"aggressive"}

# define environments and policy assignments
match_list = [  [1,1,1,1,1, 2,2,2,2,2],
                [1,1,1,1,1, 3,3,3,3,3],
                [1,1,1,1,1, 4,4,4,4,4],
                [1,1,1,1,1, 5,5,5,5,5],  ]

# train with this configuration
arena.kickoff(match_list, policies, 100000,
    render=False, scale=True)
```

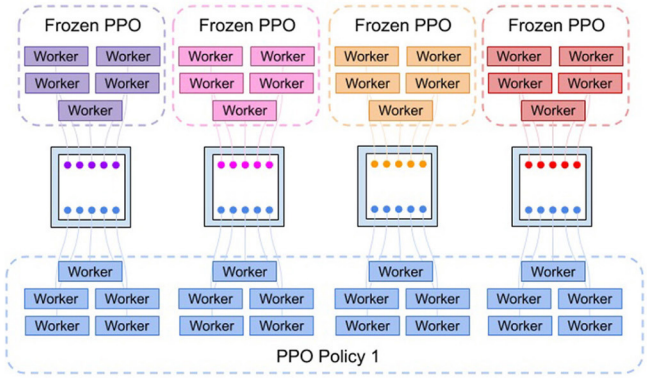

FIGURE 4
Pseudo-code and process diagram for the TanksWorld training scheme. The training scheme was used by both the static and dynamic curriculum. **(Left)** Pseudo-code of the training scheme against multiple policies. "ppo" refers to a PPO (Schulman et al., 2017) policy that is being trained, while the other policies refer to custom frozen policies. **(Right)** The resulting processes and their organization. Four environments are created, each housing 10 entities. The agents in the competition are depicted as nodes and colored based on the policies that they follow. All blue tanks are contributing to a single policy.

```
maxsteps ← 4000000 ;
maxbatches ← maxsteps /32 ;
p ← 0;
batch ← 0;
while batch ≤ max batches do
    if batch mod 20833 == 0 then
        p ← p + .05 ;
        Set penality for safety violations ← p;
    end
    Collect batch;
    Train on batch;
    batch ← batch + 1;
end
```

Algorithm 1. Dynamic training curriculum.

```
maxsteps ← 4000000 ;
maxbatches ← maxsteps /32 ;
Set the penalty for safety violations to .3 ;
batch ← 0;
while batch ≤ max batches do
    Collect batch;
    Train on batch ;
    batch ← batch + 1;
end
```

Algorithm 2. Static training curriculum.

FIGURE 5
Pseudo-code for the dynamic vs. static training curriculum. The dynamic curriculum increased penalties for safety violations from 0 to 0.3 in increments of 0.05 distributed evenly over four million steps. The static curriculum keeps the penalty of safety violations at 0.3 throughout training. The safety violation parameter sets the penalty for damage or death caused by an ally to another ally or neutral entity.

While this is a potentially powerful paradigm, it can be complex to implement. Additional details about interfaces, architecture, capabilities are described in Appendix.

4. Results and discussion

In this section, we highlight results that illustrate some of the key features of the AI Arena. Our experiments were designed to (i) test the importance of curriculum design on agent performance in the Tanksworld environment, and (ii) measure learned cooperation among several multi-agent paradigms in a cooperative navigation environment. For each experiment, we describe the environment, experiment design, and results.

4.1. The impact of curriculum design in the Tanksworld environment

A curriculum is anything that results in non-stationarity over the course of training (Bengio et al., 2009). A lot has

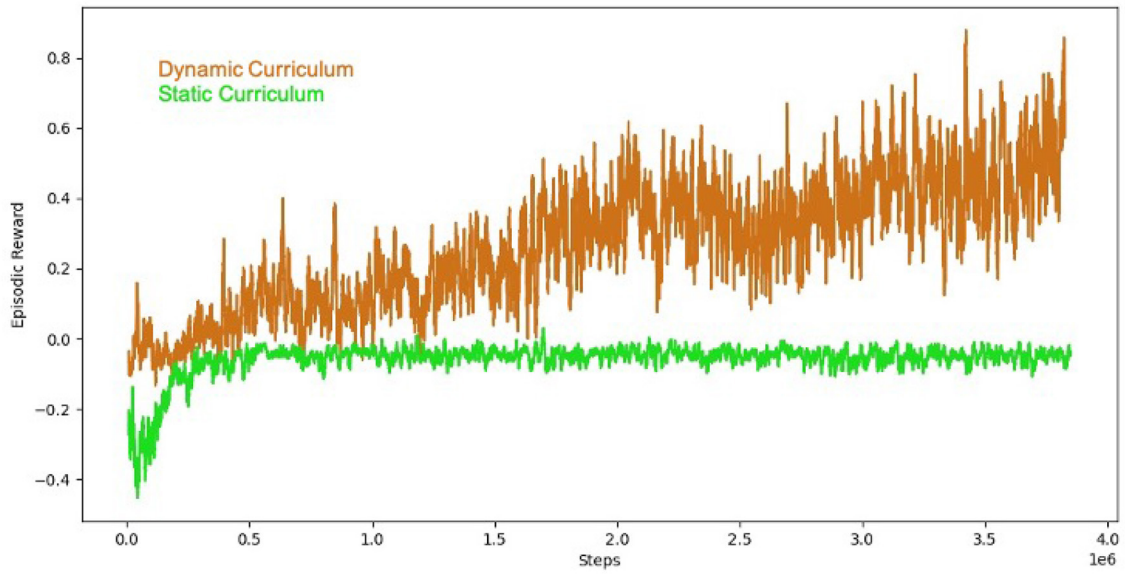

FIGURE 6
Round-based curriculum training organized with the AI Arena (brown). Successive rounds of training increased the difficulty by slowing introducing safety penalties over three rounds of training with penalty weights (0, 0.05, 0.1, 0.15, 0.20, 0.25, 0.3). The baseline (green) starts with the penalty threshold of 0.3. The result illustrates the value of successive rounds of curriculum training for teams of tanks in the AI Safety Challenge domain.

been written about the challenges of non-stationarity in multi-agent environments (Gronauer and Diepold, 2022). In this experiment, we explored the potential benefits.

4.1.1. Environment

Illustrated in Figure 3, TanksWorld (Rivera et al., 2020) is a competitive 5 vs. 5 environment that challenges teams of agents to simultaneously win against the opposing team, cooperate with diverse teammates, and cope with uncertainty in the environment. The reward structure is a linear combination of rewards from enemy kills and damage and penalties for allied and collateral kills and damage. Additional details on the Tanksworld environment and reward structure can be found in the manuscript (Rivera et al., 2020).

4.1.2. Experiment design

The experiment compares reinforcement learning training against multiple opponents simultaneously with and without curriculum training. As shown in Figure 4, we train a policy with PPO (Schulman et al., 2017) against four different opponent policies (i.e., static, random, aggressive, and passive policies). The training scheme also illustrates the expressiveness of the abstractions in the AI Arena for multi-agent training. The policy weights for the aggressive and passive policies were pretrained with to saturation *via* PPO and frozen. Curriculum training was used to slowly introducing penalties for safety violations. The curriculum was composed of increasing penalties for safety violations from 0 to .3 in increments of 0.05 distributed evenly over four million steps. We compared the curriculum training approach to a baseline approach without a curriculum that sets the penalty for safety violations at 0.3. The difference between the static and dynamic curriculum training is illustrated by the algorithms in Figure 5. This means that the final reward configuration for both the static and dynamic curriculum cases were the same. We recorded average episodic reward over the number of steps in the environment.

4.1.3. Results

The results of the comparison are shown in Figure 6. The non-curriculum baseline reaches at plateau at just below 0, while the curriculum-based approach achieves a higher overall combined episodic reward which is a combined metric including both safety and performance. The early penalties for safety violations in the static curriculum inhibited exploration leading to a sub-optimal policy.

4.2. Learned cooperation with multi-agent soft actor critic

In this experiment, we aimed to better understand the effect of different multi-agent training paradigms using soft actor critic

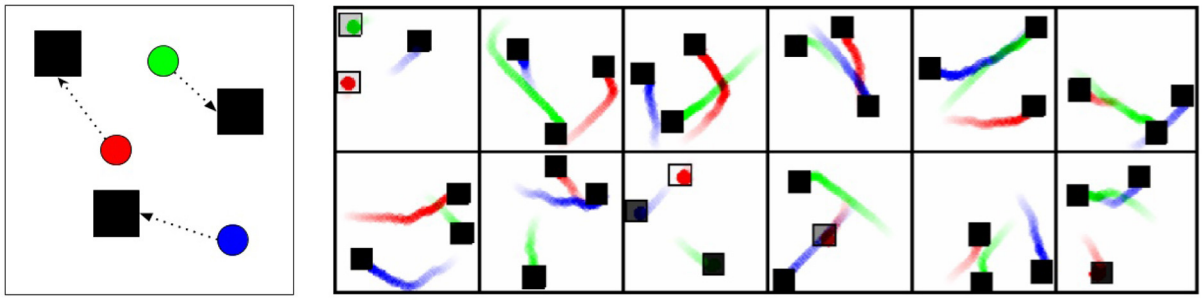

FIGURE 7
Cooperative navigation environment and behavior. The Cooperative Navigation environment has three targets (black squares) and three agents (circles). The agents must coordinate to cover all targets. **(Left)** Illustration of environment and potential solution. **(Right)** Traces of the testing behavior from the learned MASAC policies. The actions were reduced in magnitude to create slow paths to the targets. Some targets appear as outlines, showing that the agent happened to start on or near that target. The traces are interesting in that they show clear coordination but occasionally sub-optimal pairings of entities and targets. If the actions were not reduced, such that the entities reached the targets in only a handful of steps, these sub-optimalities would have little consequence on score.

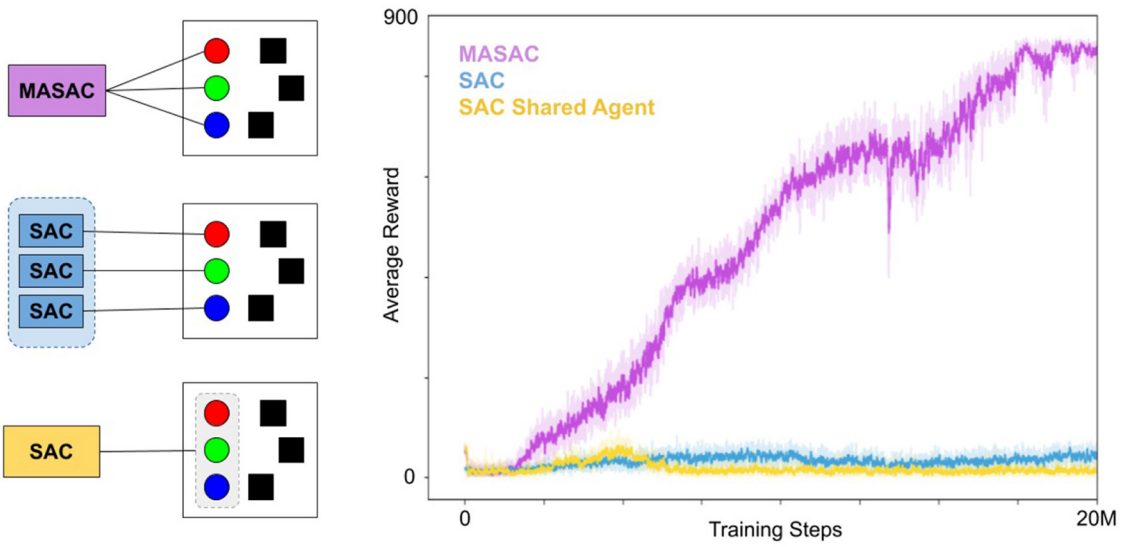

FIGURE 8
Training curves for MASAC and comparisons. **(Left)** Entity assignments for the three approaches: Truly multiagent policy (MASAC), treating each entity as an SAC worker, and grouping all entities into a single agent. In all cases, the assignment was duplicated over several environments for distributed training. **(Right)** The corresponding training curves for each approach. MASAC was the only successful algorithm, making slow and deliberate progress for roughly 18 million steps before leveling off.

(SAC) (Haarnoja et al., 2018) on the emergence of cooperation. We evaluated cooperation using the cooperative navigation environment from MADDPG (Lowe et al., 2017). In the next section, we describe the environment in more detail.

4.2.1. Environment

The cooperative navigation environment is illustrated in Figure 7. In the environment there are 3 agents that can move in 2D space and must navigate to cover three targets. Targets provide a reward of +1 if they are occupied, so the optimal behavior is to have each entity travel to a unique target, such that all targets are occupied. There is no penalty for colliding with other agents. The environment runs for 300 steps, so the maximum theoretical score is 900 (all entities starting directly on a target and staying there for the duration, for $300 \times 3 = 900$ points).

4.2.2. Experiment design

We trained and compared three paradigms for multi-agent policy learning with SAC (Haarnoja et al., 2018) including (i)

agents are controlled by individual SAC policies that share experience, (ii) a single SAC policy that controls all three agents, and a multi-agent variant of SAC (MASAC). These paradigms are illustrated in Figure 8 (left). Our implementation of Multi-agent Soft Actor Critic (MASAC) is a direct extension of soft actor critic (Haarnoja et al., 2018) to the multi-agent domain using the common critic framework initially described by MADDPG (Lowe et al., 2017). MASAC ran on eight environments (eight workers, each controlling three entities), SAC ran on six environments (18 workers, three per environment) and the combined entity SAC ran on eight environments (eight workers, each controlling three entities as one agent). All three approaches collected and trained using 20 M steps of experience.

4.2.3. Results

As seen in Figure 8, our agents converge to a cooperative set of behaviors that clear 800 points on average, which is nearly optimal. The agents move quite quickly in this environment, but we have slowed them down in testing to create visualizations of their movements in Figure 7. While they do not always attempt to reach the nearest target, they have coordinated in such a way that all the targets become occupied. Crucially, they do not communicate during testing, so it is only through training with the common critic that they have learned complementary policies that can deploy independently while still working together.

The policy for individual control of converged at a low average reward as seen in Figure 8. Treating the agents as separate workers for SAC does not properly assign rewards to the agents, since all three agents are collectively rewarded based on target occupancy, and therefore the distributed SAC approach is not able to solve the credit-assignment problem among multiple workers. Similarly, the single SAC agent controlling all three entities converged at a low level of average reward. Treating all three entities as a single agent may suffer from a similar problem in that any rewards that are experienced do not reflect credit for the action taken but rather a subset of the action taken.

5. Conclusions

In this work, we introduced the AI Arena: a scalable framework with flexible abstractions for distributed multi-agent reinforcement learning. Our aim is to help enable RL research for the class of applications that involve multiple teams of agents where each team may have unique learning strategies and where agents within a given team may have localized views of the environment. Our experiment in the Tanksworld environment illustrated the importance of curriculum design, and our experiment with cooperative navigation highlighted the importance of multi-agent algorithms for the emergence of cooperative behavior.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

CR and ES jointly wrote the manuscript, produced the figures, ran the experiments, and developed the code. AL provided crucial guidance. All authors reviewed the manuscript. All authors contributed to the article and approved the submitted version.

Acknowledgments

The authors would like to thank I-Jeng Wang and Christopher Ratto for technical and manuscript reviews.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2022.932671/full#supplementary-material

References

Abel, D. (2019). "Simple_rl: reproducible reinforcement learning in python," in *RML@ ICLR* (New Orleans, LA).

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY), 41–48. doi: 10.1145/1553374.1553380

Brown, N., and Sandholm, T. (2019). Superhuman ai for multiplayer poker. *Science* 365, 885–890. doi: 10.1126/science.aay2400

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part C* 38, 156–172. doi: 10.1109/TSMCC.2007.913919

Cai, Y., Yang, S. X., and Xu, X. (2013). "A combined hierarchical reinforcement learning based approach for multi-robot cooperative target searching in complex unknown environments," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)* (Singapore), 52–59. doi: 10.1109/ADPRL.2013.6614989

Caspi, I., Leibovich, G., Novik, G., and Endrawis, S. (2017). *Reinforcement Learning Coach*. doi: 10.5281/zenodo.1134899

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. (2020). Mushroomrl: simplifying reinforcement learning research. *arXiv preprint arXiv:2001.01102*. doi: 10.48550/arXiv.2001.01102

Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., et al. (2018). "Surreal: open-source reinforcement learning framework and robot manipulation benchmark," in *Conference on Robot Learning* (Zürich), 767–782.

Fendley, N., Costello, C., Nguyen, E., Perrotta, G., and Lowman, C. (2022). Continual reinforcement learning with tella. *arXiv preprint arXiv:2208.04287*. doi: 10.48550/arXiv.2208.04287

Gronauer, S., and Diepold, K. (2022). Multi-agent deep reinforcement learning: a survey. *Artif. Intell. Rev.* 55, 895–943. doi: 10.1007/s10462-021-09996-w

Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., et al. (2018). *TF-Agents: A Library for Reinforcement Learning in Tensorflow*. Available online at: https://github.com/tensorflow/agents

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*. doi: 10.48550/arXiv.1812.05905

Juliani, A., Berges, V.-P., Vckay, E., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: a general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*. doi: 10.48550/arXiv.1809.02627

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* 17, 1334–1373. doi: 10.48550/arXiv.1504.00702

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., et al. (2018). "Rllib: abstractions for distributed reinforcement learning," in *International Conference on Machine Learning* (Stockholm), 3053–3062.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Loon, K. W., Graesser, L., and Cvitkovic, M. (2019). SLM lab: a comprehensive benchmark and modular software framework for reproducible deep reinforcement learning. *arXiv preprint arXiv:1912.12482*. doi: 10.48550/arXiv.1912.12482

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O. and Mordatch, I., (2017). "Multi-agent actor-critic for mixed cooperativecompetitive environments," in *Advances in Neural Information Processing Systems* (Long Beach, CA), 30.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*. doi: 10.48550/arXiv.1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. (2018). "Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning* (Stockholm), 4295–4304.

Rivera, C. G., Lyons, O., Summitt, A., Fatima, A., Pak, J., Shao, W., et al. (2020). Tanksworld: a multi-agent environment for AI safety research. *arXiv preprint arXiv:2002.11174*. doi: 10.48550/arXiv.2002.11174

Schaarschmidt, M., Mika, S., Fricke, K., and Yoneki, E. (2018). Rlgraph: modular computation graphs for deep reinforcement learning. *arXiv preprint arXiv:1810.09028*. doi: 10.48550/arXiv.1808.07903

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. doi: 10.48550/arXiv.1707.06347

Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*. doi: 10.48550/arXiv.1610.03295

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529, 484–489. doi: 10.1038/nature16961

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. (2019). "QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International Conference on Machine Learning* (Long Beach, CA), 5887–5896.

Stooke, A., and Abbeel, P. (2019). rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*. doi: 10.48550/arXiv.1909.01500

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: A Bradford Book.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* 575, 350–354. doi: 10.1038/s41586-019-1724-z

Zheng, L., Yang, J., Cai, H., Zhang, W., Wang, J., and Yu, Y. (2017). MAGENT: a many-agent reinforcement learning platform for artificial collective intelligence. *arXiv preprint arXiv:1712.00600*. doi: 10.1609/aaai.v32i1.11371