Emergent Tool Use From Multi-Agent Autocurricula

Bowen Baker* OpenAI bowen@openai.com
Ingmar Kanitscheider* OpenAI ingmar@openai.com
Todor Markov* OpenAI todor@openai.com
Yi Wu* OpenAI jxwuyi@openai.com
Glenn Powell* OpenAI glenn@openai.com
Bob McGrew* OpenAI bmcgrew@openai.com
Igor Mordatch*† Google Brain imordatch@google.com

Abstract

Through multi-agent competition, the simple objective of *hide-and-seek*, and standard reinforcement learning algorithms at scale, we find that agents create a self-supervised autocurriculum inducing multiple distinct rounds of emergent strategy, many of which require sophisticated tool use and coordination. We find clear evidence of six emergent phases in agent strategy in our environment, each of which creates a new pressure for the opposing team to adapt; for instance, agents learn to build multi-object shelters using moveable boxes which in turn leads to agents discovering that they can overcome obstacles using ramps. We further provide evidence that multi-agent competition may scale better with increasing environment complexity and leads to behavior that centers around far more human-relevant skills than other self-supervised reinforcement learning methods such as intrinsic motivation. Finally, we propose transfer and fine-tuning as a way to quantitatively evaluate targeted capabilities, and we compare hide-and-seek agents to both intrinsic motivation and random initialization baselines in a suite of domain-specific intelligence tests.

1 Introduction

Creating intelligent artificial agents that can solve a wide variety of complex human-relevant tasks has been a long-standing challenge in the artificial intelligence community. Of particular relevance to humans will be agents that can sense and interact with objects in a physical world. One approach to creating these agents is to explicitly specify desired tasks and train a reinforcement learning (RL) agent to solve them. On this front, there has been much recent progress in solving physically grounded tasks, e.g. dexterous in-hand manipulation [1, 2] or locomotion of complex bodies [3, 4]. However, specifying reward functions or collecting demonstrations in order to supervise these tasks

*This was a large project and many people made significant contributions. Bowen, Bob, and Igor conceived the project and provided guidance through all stages of the work. Bowen created the initial environment, infrastructure and models, and obtained the first results of sequential skill progression. Ingmar obtained the first results of tool use, contributed to environment variants, created domain-specific statistics, and with Bowen created the final environment. Todor created the manipulation tasks in the transfer suite, helped Yi with the RND baseline, and prepared code for open-sourcing. Yi created the navigation tasks in the transfer suite, intrinsic motivation comparisons, and contributed to environment variants. Glenn contributed to designing the final environment and created final renderings and project video. Igor provided research supervision and team leadership.

†Work performed while at OpenAI

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

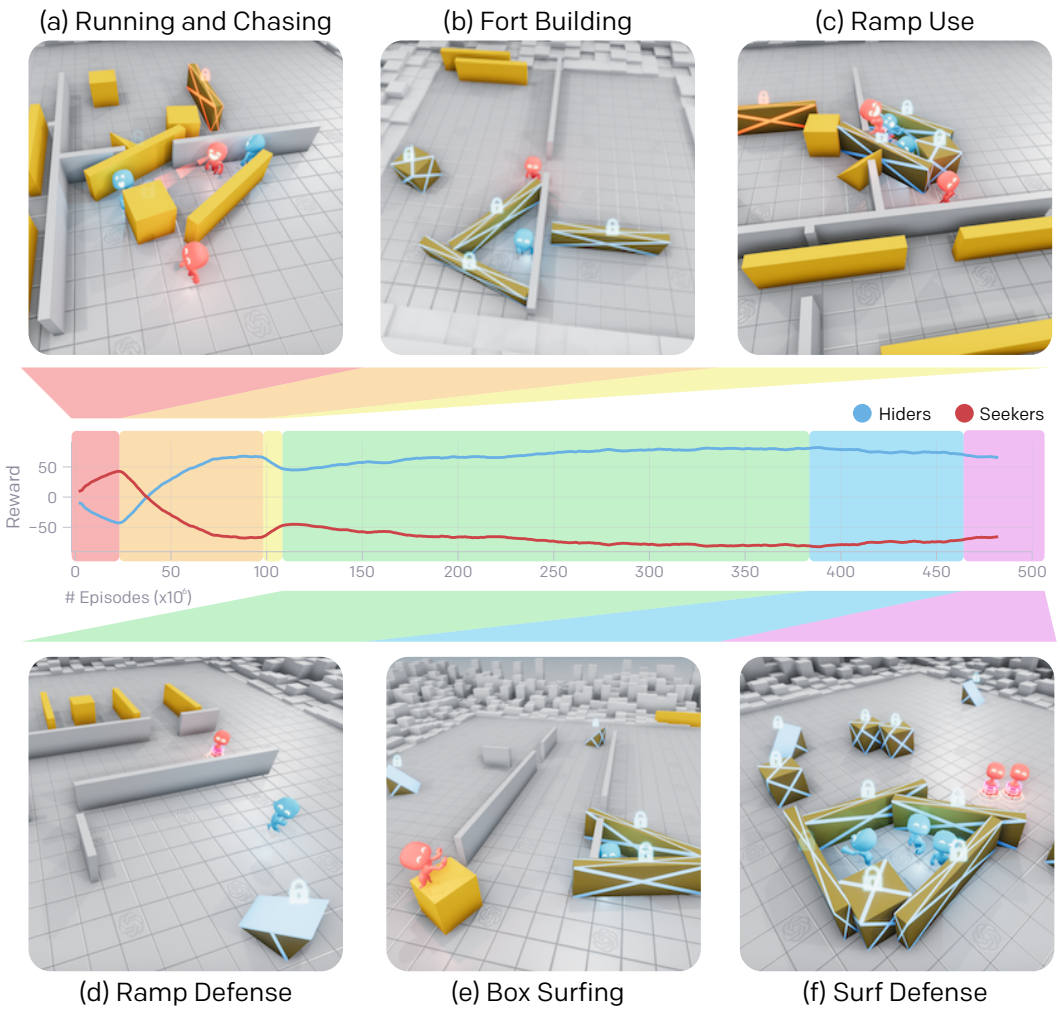

Figure 1: Emergent Skill Progression. Agents go through 6 distinct stages of emergence: (a) Seekers (red) learn to chase hiders, and hiders (blue) learn to crudely run away. (b) Hiders learn basic tool use, using boxes and sometimes existing walls to construct forts. (c) Seekers learn to use ramps to jump into the hiders' shelter. (d) Hiders quickly learn to move ramps to the edge of the play area, far from where they will build their fort, and lock them in place. (e) Seekers learn that they can jump from locked ramps to unlocked boxes and then *surf* the box to the hiders' shelter, which is possible because the environment allows agents to move together with the box regardless of whether they are on the ground or not. (f) Hiders learn to lock all the unused boxes before constructing their fort.

can be time consuming and costly. Furthermore, the learned skills in these single-agent RL settings are inherently bounded by the task description; once the agent has learned to solve the task, there is little room to improve.

Due to the high likelihood that direct supervision will not scale to unboundedly complex tasks, many have worked on unsupervised exploration and skill acquisition methods such as intrinsic motivation. However, current undirected exploration methods scale poorly with environment complexity and are drastically different from the way organisms evolve on Earth. The vast amount of complexity and diversity on Earth evolved due to co-evolution and competition between organisms, directed by natural selection [5]. When a new successful strategy or mutation emerges, it changes the implicit task distribution neighboring agents need to solve and creates a new pressure for adaptation. These evolutionary arms races create implicit *autocurricula* [6] whereby competing agents continually create new tasks for each other. There has been much success in leveraging multi-agent autocurricula to solve multi-player games, both in classic discrete games such as TD-Gammon [7] and Go [8], as

well as in continuous real-time domains such as Dota [9] and Starcraft [10]. Despite the impressive emergent complexity in these environments, the learned behavior is quite abstract and disembodied from the physical world. Our work sees itself in the tradition of previous studies that showcase emergent complexity in simple physically grounded environments [11, 12, 13, 14]; the success in these settings inspires confidence that inducing autocurricula in physically grounded and open-ended environments could eventually enable agents to acquire an unbounded number of human-relevant skills.

We introduce a new mixed competitive and cooperative physics-based environment in which agents compete in a simple game of hide-and-seek. Through only a visibility-based reward function and competition, agents learn many emergent skills and strategies including coordinated tool use. For example, hiders learn to create shelter from seekers by barricading doors or constructing multi-object forts, and as a counter strategy seekers learn to use ramps to jump into hiders' shelter. Moreover, we observe signs of dynamic and growing complexity resulting from multi-agent competition and standard reinforcement learning algorithms; we find that agents go through as many as six distinct adaptations of strategy and counter-strategy, which are depicted in Figure 1. We further present evidence that multi-agent co-adaptation may scale better with environment complexity and qualitatively centers around more human-interpretable behavior than intrinsically motivated agents.

However, as environments increase in scale and multi-agent autocurricula become more open-ended, evaluating progress by qualitative observation will become intractable. We therefore propose a suite of targeted intelligence tests to measure capabilities in our environment that we believe our agents may eventually learn, e.g. object permanence [15], navigation, and construction. We find that for a number of the tests, agents pretrained in hide-and-seek learn faster or achieve higher final performance than agents trained from scratch or pretrained with intrinsic motivation; however, we find that the performance differences are not drastic, indicating that much of the skill and feature representations learned in hide-and-seek are entangled and hard to fine-tune.

The main contributions of this work are: 1) clear evidence that multi-agent autocurricula lead to many distinct and compounding phase shifts in agent strategy, 2) evidence that when induced in a physically grounded environment, multi-agent autocurricula can lead to human-relevant skills such as tool use, 3) a proposed framework for evaluating agents in open-ended environments as well as a suite of targeted intelligence tests for our domain, and 4) open-sourced environments and code[3] for environment construction to encourage further research in physically grounded multi-agent autocurricula.

2 Related Work

There is a long history of using self-play in multi-agent settings. Early work explored self-play using genetic algorithms [16, 17, 18, 19]. [11] and [20] studied the emergent complexity in morphology and behavior of creatures that coevolved in a simulated 3D world. Open-ended evolution was further explored in the environments Polyworld [21] and Geb [22], where agents compete and mate in a 2D world, and in Tierra [23] and Avida [24], where computer programs compete for computational resources. More recent work attempted to formulate necessary preconditions for open-ended evolution [25, 26]. Co-adaptation between agents and environments can also give rise to emergent complexity [27, 28, 29]. In the context of multi-agent RL, [7], [30], [9], [13] and [10] used self-play with deep RL techniques to achieve super-human performance in Backgammon, Go, Dota, Capture-the-Flag and Starcraft, respectively. [12] trained agents in a simulated 3D physics environment to compete in various games such as sumo wrestling and soccer goal shooting. In [14], agents learn to manipulate a soccer ball in a 3D soccer environment and discover emergent behaviors such as ball passing and interception. In addition, communication has also been shown to emerge from multi-agent RL [31, 32, 33, 34].

Intrinsic motivation methods have been widely studied in the literature [35, 36]. One example is count-based exploration, where agents are incentivized to reach infrequently visited states by maintaining state visitation counts [37, 38, 39] or density estimators [40, 41]. Another paradigm are transition-based methods, in which agents are rewarded for high prediction error in a learned forward or inverse dynamics model [42, 43, 44, 45, 46, 47, 48, 49]. [50] consider multi-agent scenarios and adopt causal influence as a motivation for coordination. In our work, we utilize intrinsic motivation methods as an alternative exploration baseline to multi-agent autocurricula. Similar comparisons have also been made in [49] and [51].

[3] Code can be found at https://github.com/openai/multi-agent-emergence-environments

Tool use is a hallmark of human and animal intelligence [52, 53]; however, learning tool use in RL settings can be a hard exploration problem when rewards are unaligned. For example, in [54] a real-world robot uses familiar and novel tools to move objects using model-based RL and imitation learning. In [55], an agent solves construction tasks in a 2-D environment using both model-based and model-free methods. [56] uses a combination of human-designed priors and model-based policy optimization to solve a collection of physics-based puzzles requiring tool use. However, in each of these works, agents were explicitly incentivized to interact with and use tools, whereas in our environment agents implicitly create this incentive through multi-agent competition.

3 Hide and Seek

Agents are tasked with competing in a two-team hide-and-seek game in a physics-based environment. The *hiders* are tasked with avoiding line of sight from the *seekers*, and the seekers are tasked with keeping vision of the hiders. There are objects scattered throughout the environment that the agents can grab and also lock in place. There are also randomly generated immovable rooms and walls that the agents must learn to navigate. Before the game of hide-and-seek begins, the hiders are given a *preparation phase* where the seekers are immobilized, giving the hiders a chance to run away or change their environment.

There are no explicit incentives for agents to interact with objects in the environment; the only supervision given is through the hide-and-seek objective. Agents are given a *team based* reward; hiders are given a reward of 1 if all hiders are hidden and -1 if any hider is seen by a seeker. Seekers are given the opposite reward, -1 if all hiders are hidden and +1 otherwise. To confine agent behavior to a reasonable space, agents are penalized if they go too far outside the play area. An episode lasts 240 timesteps, the first 40% of which are the preparation phase where all agents are given zero reward.

We simulate the environment in the MUJOCO physics engine [57]. The world is populated with 1 to 3 hiders, 1 to 3 seekers, 3 to 9 movable boxes of which at least 3 are elongated, 2 movable ramps, and randomly generated static walls and rooms. We also experiment with a simpler, less randomized environment described in Appendix A.3. Agents observe the position, velocity, and size (in the case of the randomly shaped boxes) of objects and other agents. If entities are not in line-of-sight of the agent or not in a 135 degree cone in front of the agent, then they are masked out in the policy. Agents also have 30 range sensors arrayed evenly around them, similar to a lidar. In addition, each agent observes its own team and other agents' teams as well as how much time is left in the preparation phase.

Agents are simulated as spherical objects and have 3 action types that can be chosen simultaneously at each time step. They may *move* by setting a discretized force along their x and y axis and torque around their z-axis. They have a single binary action to *grab* objects, which binds the agent to the closest object while the action is enabled. Agents may also *lock* objects in place with a single binary action. Objects may be unlocked only by agents on the team of the agent who originally locked the object. Agents may only grab or lock objects that are in front of them and within a small radius.

4 Policy Optimization

Agents are trained using self-play and Proximal Policy Optimization (PPO) [58] in *rapid* [9], a large-scale distributed RL framework. All agents share the same policy parameters (agents observe whether they are rewarded as hiders or seekers), but act and observe independently. Policy and value function networks have separate parameters. At optimization time, the value function has access to the full unmasked environment state, similar to [59, 33, 60]. More details of policy architecture and optimization are described in Appendix B.

5 Auto-Curricula and Emergent Behavior

As agents train against each other in hide-and-seek, as many as six distinct strategies emerge, each of which creates a previously non-existing pressure for agents to progress to the next stage. Note that there are no direct incentives for agents to interact with objects or to explore, but rather the emergent strategies are solely a result of the autocurriculum induced by multi-agent competition. Figure 1

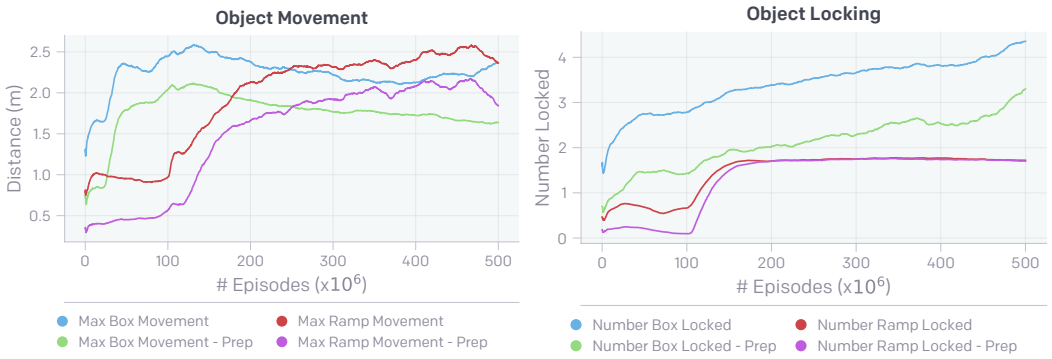

Figure 2: Environment specific statistics track skill discovery in hide-and-seek. Maximum movement of objects (left) and number of objects locked (right) at the end of preparation phase ("Prep") and at the end of episode. Newly learned skills induce rapid shifts in behavior.

shows the progression of emergent strategies agents learn in our environment (see Appendix A.1 for trajectory traces of each strategy[4]).

Initially, hiders and seekers learn to crudely run away and chase. After approximately 25 million episodes of hide-and-seek, hider learn to construct secure shelters by moving many boxes together or against walls and locking them in place. After another 75 million episodes, the seekers also learn rudimentary tool use; they learn to move and use ramps to jump over obstacles, allowing them to enter the hiders' shelter. 10 million episodes later, the hiders learn to defend against this strategy; the hiders learn to bring the ramps to the edge of the play area and lock them in place, seemingly removing the only tool the seekers have at their disposal.

After 380 million total episodes of training, the seekers learn to bring a box to the edge of the play area where the hiders have locked the ramps. The seekers then jump on top of the box and *surf* it to the hiders' shelter; this is possible because the environment allows agents to move together with the box regardless of whether they are on the ground or not. In response, the hiders learn to lock all of the boxes in place before building their shelter.

Many stages of emergent strategy can be mapped to behavioral shifts in the way agents interact with the tools in their environment, similar to [61, 62]. We therefore track basic statistics about the agents' interaction with objects during training, shown in Figure 2. For instance, as the hiders learn to build forts, they move and lock boxes much more during the preparation phase. Similarly, as the seekers learn to move and use ramps, the ramp movement in the main phase of the game increases, and as they learn to "box surf" there is a slight increase in the box movement during the main phase of the game. Finally, as the hiders learn to defend against this strategy by locking all boxes in place, the number of locked boxes in the preparation phase increases. These behavioral shifts are consistent across separate training runs with different initial seeds, as shown in Appendix A.2.

We found that scale plays a critical role in enabling progression through the emergent autocurricula in hide-and-seek. The default model, which uses a batch size of 64,000 and 1.6 million parameters, requires 132.3 million episodes (31.7 billion frames) over 34 hours of training to reach stage 4 of the skill progression, i.e. ramp defense. In Figure 3 we show the effect of varying the batch size in our agents ability to reach stage 4. We find that larger batch sizes lead to much quicker training time by virtue of reducing the number of required optimization steps, while only marginally affecting sample efficiency down to a batch size of 32,000; however, we found that experiments with batch sizes of 16,000 and 8,000 never converged.

In addition, we found emergent tool use when adding additional objects and objectives to our hide-and-seek environment as well as with several alternate game variants, providing further evidence that multi-agent interaction is a promising path towards self-supervised skill acquisition. (see Appendix A.6)

6 Evaluation

[4]Videos can be found at https://openai.com/blog/emergent-tool-use

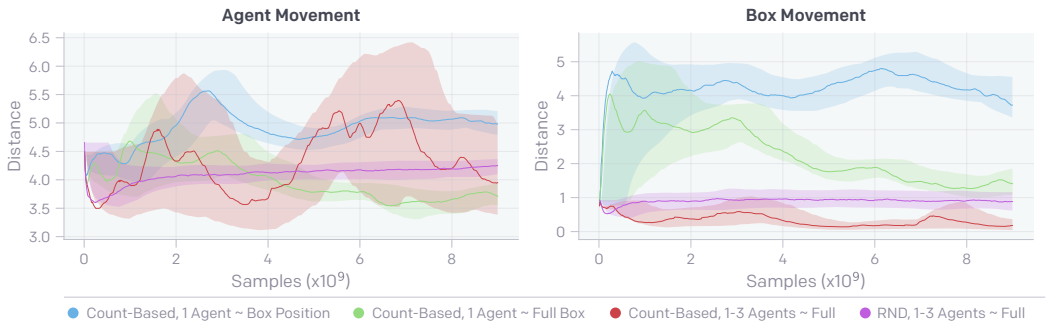

Figure 4: Behavioral statistics (box movement at end of episode and maximum agent movement) of count-based exploration variants and random network distillation (RND) across 3 seeds. State representations for count-based exploration: Single agent, 2-D box location (blue), Single agent, box location, rotation and velocity (green) and 1-3 agents, full observation space (red). Also shown is RND for 1-3 agents with full observation space (purple).

Evaluation is difficult in multi-agent environments: Tracking reward is an insufficient evaluation metric, as it can be ambiguous in indicating whether agents are improving evenly or have stagnated. Metrics like ELO [63] or Trueskill [64] can more reliably measure whether performance is improving relative to previous policy versions or other policies in a population; however, these metrics still do not give insight into whether improved performance stems from new adaptations or improving previously learned skills. Finally, using environment specific statistics such as object movement (see Figure 2) can also be ambiguous, e.g. the choice to track absolute movement does not illuminate which direction agents moved, and designing sufficient metrics will become difficult and costly as environments scale.

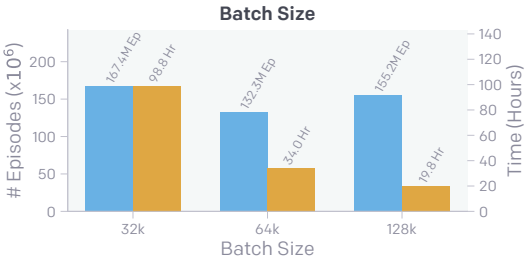

Figure 3: Effect of Scale on Emergent Autocurricula. Number of episodes (blue) and wall clock time (orange) required to achieve stage 4 (ramp defense) of the emergent skill progression presented in Figure 1. Batch size denotes number of chunks, each of which consists of 10 contiguous transitions (the truncation length for backpropagation through time).

Here we compare emergent behaviors to those learned from intrinsic motivation, a technique for unsupervised exploration and skill discovery, and propose a suite of domain-specific intelligence tests to quantitatively measure agent capabilities.

6.1 Comparison to Intrinsic Motivation

We first compare behaviors learned in hide-and-seek to a count-based exploration baseline [37] with an object invariant state representation. Count-based objectives explicitly keep track of state visitation counts and reward agents for reaching infrequently visited states (cf Appendix D). We find that intrinsic motivation methods lead to behavior that is less meaningful to humans than multi-agent competition, particularly as the dimensionality of state representation increases.

As a proxy for meaningful behavior, we measure the behavioral statistics of agent and box movement for different dimensionality of the corresponding state representation (Figure 4). In contrast to the original hide-and-seek environment, where the initial locations of agents and objects are randomized, we restrict the initial locations to a quarter of the game area to ensure that agents receive additional rewards for exploring. We find that count-based exploration leads to the largest agent and box movement if the state representation only contains the 2-D location of boxes: the agent consistently interacts with objects and learns to navigate. Yet, when using progressively higher-dimensional state representations, such as box location, rotation and velocity or 1-3 agents with full observation space, agent movement and, in particular, box movement decrease substantially. This is a severe limitation because it indicates that, when faced with highly complex environments, count-based exploration techniques require identifying by hand the "interesting" directions in state space that are relevant for the behaviors one would like the agents to discover. Conversely, multi-agent self-play does not need

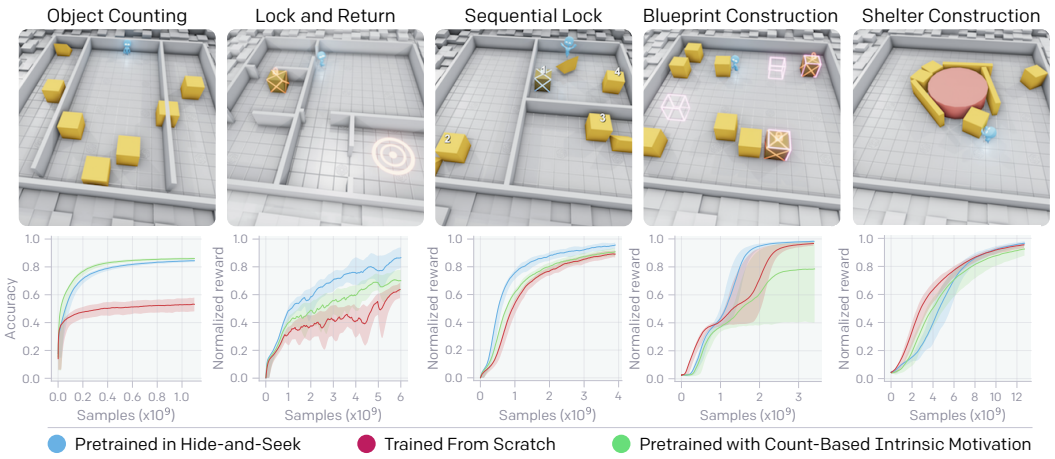

Figure 5: Fine-tuning Results. We plot the mean normalized performance and 90% confidence interval across 3 seeds smoothed with an exponential moving average, except for Blueprint Construction where we plot over 6 seeds due to higher training variance.

this degree of supervision. We also train agents with random network distillation (RND) [41], an intrinsic motivation method designed for high dimensional observation spaces, and find it to perform only slightly better than count-based exploration in the full state setting.

6.2 Transfer and Fine-tuning as evaluation

We propose to use transfer to a suite of domain-specific tasks in order to asses agent capabilities. To this end, we have created 5 benchmark intelligence tests that include both supervised and reinforcement learning tasks. The tests use the same action space, observation space, and types of objects as in the hide-and-seek environment. We examine whether pretraining agents in our multi-agent environment and then fine-tuning them on the evaluation suite leads to faster convergence or improved overall performance compared to training from scratch or pretraining with count-based intrinsic motivation. We find that on 3 out of 5 tasks, agents pretrained in the hide-and-seek environment learn faster and achieve a higher final reward than both baselines.

We categorize the 5 intelligence tests into 2 domains: cognition and memory tasks, and manipulation tasks. We briefly describe the tasks here; for the full task descriptions, see Appendix C. For all tasks, we reinitialize the parameters of the final dense layer and layernorm for both the policy and value networks.

Cognition and memory tasks:

In the *Object Counting* supervised task, we aim to measure whether the agents have a sense of object permanence; the agent is pinned to a location and watches as 6 boxes each randomly move to the right or left where they eventually become obscured by a wall. It is then asked to predict how many boxes have gone to each side for many timesteps after all boxes have disappeared. The agent's policy parameters are frozen and we initialize a classification head off of the LSTM hidden state. In the baseline, the policy network has frozen random parameters and only the classification head off of the LSTM hidden state is trained.

In *Lock and Return* we aim to measure whether the agent can remember its original position while performing a new task. The agent must navigate an environment with 6 random rooms and 1 box, lock the box, and return to its starting position.

In *Sequential Lock* there are 4 boxes randomly placed in 3 random rooms without doors but with a ramp in each room. The agent needs to lock all the boxes in a particular order — a box is only lockable when it is locked in the correct order — which is unobserved by the agent. The agent must discover the order, remember the position and status of visited boxes, and use ramps to navigate between rooms in order to finish the task efficiently.

Manipulation tasks: With these tasks we aim to measure whether the agents have any latent skill or representation useful for manipulating objects.

In the *Construction From Blueprint* task, there are 8 cubic boxes in an open room and between 1 and 4 target sites. The agent is tasked with placing a box on each target site.

In the *Shelter Construction* task there are 3 elongated boxes, 5 cubic boxes, and one static cylinder. The agent is tasked with building a shelter around the cylinder.

Results: In Figure 5 we show the performance on the suite of tasks for the hide-and-seek, count-based, and trained from scratch policies across 3 seeds. The hide-and-seek pretrained policy performs slightly better than both the count-based and the randomly initialized baselines in *Lock and Return*, *Sequential Lock* and *Construction from Blueprint*; however, it performs slightly worse than the count-based baseline on *Object Counting*, and it achieves the same final reward but learns slightly slower than the randomly initialized baseline on *Shelter Construction*.

We believe the cause for the mixed transfer results is rooted in agents learning skill representations that are entangled and difficult to fine-tune. We conjecture that tasks where hide-and-seek pretraining outperforms the baseline are due to reuse of learned feature representations, whereas better-than-baseline transfer on the remaining tasks would require reuse of learned skills, which is much more difficult. This evaluation metric highlights the need for developing techniques to reuse skills effectively from a policy trained in one environment to another. In addition, as future environments become more diverse and agents must use skills in more contexts, we may see more generalizable skill representations and more significant signal in this evaluation approach.

In Appendix A.5 we further evaluate policies sampled during each phase of emergent strategy on the suite of targeted intelligence tasks, by which we can gain intuition as to whether the capabilities we measure improve with training, are transient and accentuated during specific phases, or generally uncorrelated to progressing through the autocurriculum. Noteably, we find the agent's memory improves through training as indicated by performance in the navigation tasks; however, performance in the manipulation tasks is uncorrelated, and performance in object counting changes seems transient with respect to source hide-and-seek performance.

7 Discussion and Future Work

We have demonstrated that simple game rules, multi-agent competition, and standard reinforcement learning algorithms at scale can induce agents to learn complex strategies and skills. We observed emergence of as many as six distinct rounds of strategy and counter-strategy, suggesting that multi-agent self-play with simple game rules in sufficiently complex environments could lead to open-ended growth in complexity. We then proposed to use transfer as a method to evaluate learning progress in open-ended environments and introduced a suite of targeted intelligence tests with which to compare agents in our domain.

Our results with hide-and-seek should be viewed as a proof of concept showing that multi-agent autocurricula can lead to physically grounded and human-relevant behavior. We acknowledge that the strategy space in this environment is inherently bounded and likely will not surpass the six modes presented as is; however, because it is built in a high-fidelity physics simulator it is physically grounded and very extensible. In order to support further research in multi-agent autocurricula, we are open-sourcing our environment code[5].

Hide-and-seek agents require an enormous amount of experience to progress through the six stages of emergence, likely because the reward functions are not directly aligned with the resulting behavior. While we have found that standard reinforcement learning algorithms are sufficient, reducing sample complexity in these systems will be an important line of future research. Better policy learning algorithms or policy architectures are orthogonal to our work and could be used to improve sample efficiency and performance on transfer evaluation metrics.

We also found that agents were very skilled at exploiting small inaccuracies in the design of the environment, such as seekers surfing on boxes without touching the ground, hiders running away from the environment while shielding themselves with boxes, or agents exploiting inaccuracies of the physics simulations to their advantage. Investigating methods to generate environments without these unwanted behaviors is another import direction of future research [65, 66].

[5]Code can be found at https://github.com/openai/multi-agent-emergence-environments

Acknowledgments

We thank Pieter Abbeel, Rewon Child, Jeff Clune, Harri Edwards, Jessica Hamrick, Joel Liebo, John Schulman and Peter Welinder for their insightful comments on this manuscript. We also thank Alex Ray for writing parts of our open sourced code.

References

[1] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[2] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[3] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[4] Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[5] Richard Dawkins and John Richard Krebs. Arms races between and within species. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 205(1161):489–511, 1979.

[6] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.

[7] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[8] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[9] OpenAI. OpenAI Five. `https://blog.openai.com/openai-five/`, 2018.

[10] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, et al. AlphaStar: Mastering the real-time strategy game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[11] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

[12] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018.

[13] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[14] Siqi Liu, Guy Lever, Nicholas Heess, Josh Merel, Saran Tunyasuvunakool, and Thore Graepel. Emergent coordination through competition. In *International Conference on Learning Representations*, 2019.

[15] Renée Baillargeon and Susan Carey. Core cognition and beyond: The acquisition of physical and numerical knowledge. In *In S. Pauen (Ed.), Early Childhood Development and Later Outcome*, pages 33–65. University Press, 2012.

[16] Jan Paredis. Coevolutionary computation. *Artificial life*, 2(4):355–375, 1995.

[17] Jordan B Pollack, Alan D Blair, and Mark Land. Coevolution of a backgammon player. In *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 92–98. Cambridge, MA: The MIT Press, 1997.

[18] Christopher D Rosin and Richard K Belew. Methods for competitive co-evolution: Finding opponents worth beating. In *ICGA*, pages 373–381, 1995.

[19] Kenneth O Stanley and Risto Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of artificial intelligence research*, 21:63–100, 2004.

[20] Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

[21] Larry Yaeger. Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 17, pages 263–263. ADDISON-WESLEY PUBLISHING CO, 1994.

[22] AD Channon, RI Damper, et al. Perpetuating evolutionary emergence. *FROM ANIMALS TO ANIMATS 5*, pages 534–539, 1998.

[23] Thomas S Ray. Evolution, ecology and optimization of digital organisms. Technical report, Citeseer, 1992.

[24] Charles Ofria and Claus O Wilke. Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229, 2004.

[25] Tim Taylor. Requirements for open-ended evolution in natural and artificial systems. *arXiv preprint arXiv:1507.07403*, 2015.

[26] L Soros and Kenneth Stanley. Identifying necessary conditions for open-ended evolution through the artificial life world of chromaria. In *Artificial Life Conference Proceedings 14*, pages 793–800. MIT Press, 2014.

[27] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495, 2017.

[28] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

[29] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[30] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[31] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[32] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[33] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[34] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[35] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[36] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[37] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[38] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[39] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[40] Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org, 2017.

[41] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[42] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[43] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[44] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[45] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[46] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[47] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787, 2017.

[48] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

[49] Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. In *Advances in Neural Information Processing Systems*, pages 8388–8399, 2018.

[50] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049, 2019.

[51] Joel Z Leibo, Julien Perolat, Edward Hughes, Steven Wheelwright, Adam H Marblestone, Edgar Duéñez-Guzmán, Peter Sunehag, Iain Dunning, and Thore Graepel. Malthusian reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1099–1107. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[52] Gavin R Hunt. Manufacture and use of hook-tools by new caledonian crows. *Nature*, 379(6562):249, 1996.

[53] Robert W Shumaker, Kristina R Walkup, and Benjamin B Beck. *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press, 2011.

[54] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019.

[55] Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly Stachenfeld, Pushmeet Kohli, Peter Battaglia, and Jessica Hamrick. Structured agents for physical construction. In *International Conference on Machine Learning*, pages 464–474, 2019.

[56] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. The tools challenge: Rapid trial-and-error learning in physical problem solving. *arXiv preprint arXiv:1907.09620*, 2019.

[57] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[58] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[59] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[60] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[61] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[62] Julien Perolat, Joel Z Leibo, Vinicius Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*, pages 3643–3652, 2017.

[63] A.E. Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[64] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{TM}}$: a bayesian skill rating system. In *Advances in neural information processing systems*, pages 569–576, 2007.

[65] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[66] Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Beslon, David M Bryson, et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453*, 2018.

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

[68] Christopher D Rosin and Richard K Belew. New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29, 1997.

[69] Sevan G Ficici and Jordan B Pollack. A game-theoretic memory mechanism for coevolution. In *Genetic and Evolutionary Computation Conference*, pages 286–297. Springer, 2003.

[70] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

[71] Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247. ACM, 2008.

[72] Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[73] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[75] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[76] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

A Further Emergence Results

A.1 Trajectory Traces From Each Stage of Emergent Strategy

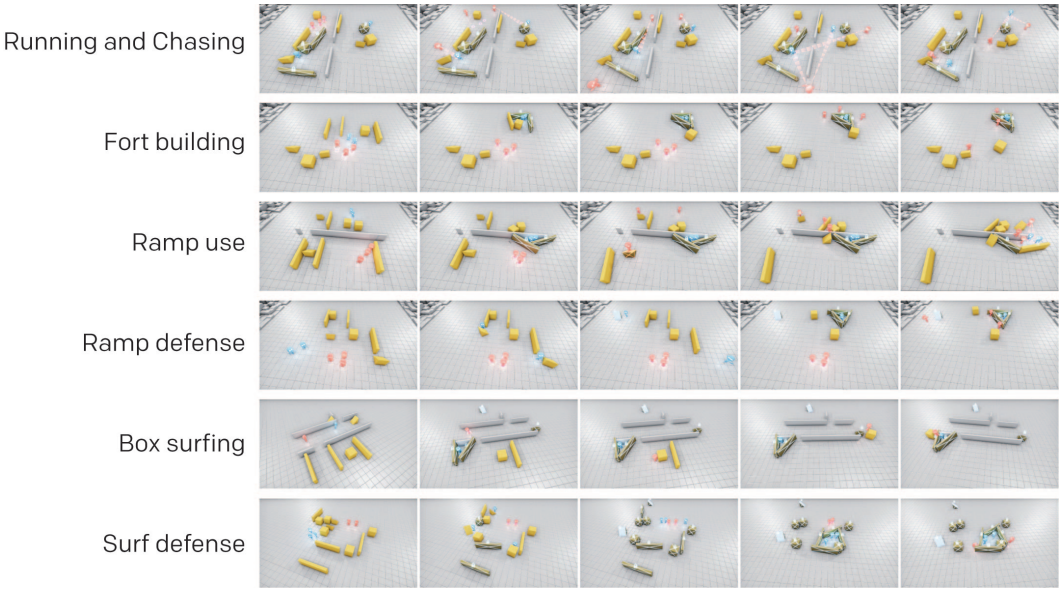

Figure A.1: Trajectory Traces From Each Stage of Emergent Strategy. Rows correspond to different strategies, columns to characteristic snapshots in chronological order within an episode that showcase the strategy. *Running and chasing*: Hiders (blue) try to hide behind walls and moveable objects, and seekers (red) chase and search for hiders. *Fort building:* During the preparation phase (first 3 snapshots), hiders build a fort out of 3 boxes and hide inside. Seekers are not able to overcome this barrier during the remainder of the trial. *Ramp use:* During the preparation phase, hiders build a fort out of 2 boxes and a wall (1st and 2nd snapshot). After the preparation phase, a seeker grabs a ramp (3rd snapshot), drags it towards the fort (4th snapshot) and uses it to enter the fort (5th snapshot). *Ramp defense:* Hiders prevent ramp use by dragging ramps away and locking them in place during the preparation phase (1st, 2nd and 3rd snapshot). They then build a fort in safe distance from the ramps (4th snapshot). After the preparation phase, seekers are not able to enter the fort (5th snapshot). *Box surfing:* A single hider locks both ramps in place and hides inside a fort built from 2 boxes and an interior wall during the preparation phase (1st and 2nd snapshot). The seeker grabs a square box (3rd snapshot) and drags it next to one of the locked ramps to jump on top (4th snapshot). It then "surfs" on the box to the edge of the fort and jumps inside (5th snapshot). *Surf defense*: During the preparation phase, hiders lock all ramps and boxes in place (1st, 2nd and 3rd snapshot) and construct a fort out of 3 boxes (4th snapshot). After the preparation phase, seekers are not able to enter the fort (5th snapshot).

A.2 Dependence of Skill Emergence on Initial Random Seed

To determine the seed dependence of skill emergence we ran two more seeds in the main environment and looked at the variability of the onset of behavioral shifts across runs (Figure A.2). We find that the behavioral shifts happen at very similar times early in training, but there is slightly more variability late in training.

A.3 Quadrant Environment

As well as the default *random rooms* environment described above, we also train and evaluate agents in the much simpler *quadrant* environment (Figure A.3) for illustration purposes. This environment consists of a room in the lower right corner that is connected by 1-2 doors to the remainder of the environment. Unlike in the main environment, the interior walls are fixed and only the location of the doors vary from episode to episode. Hiders are spawned randomly in the environment, but seekers are only spawned outside the room. Two cubic boxes are spawned inside the room and one ramp is

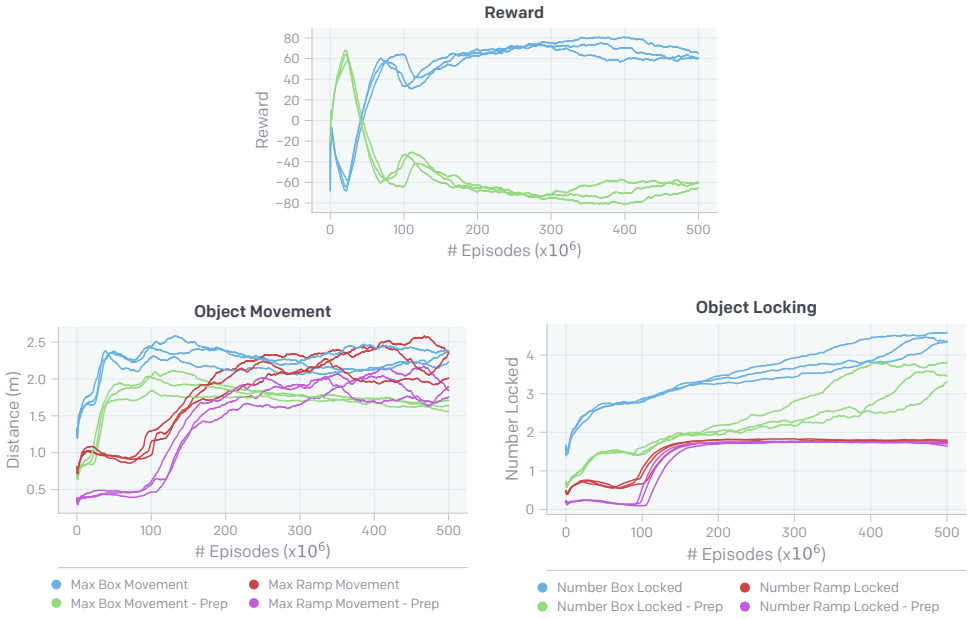

Figure A.2: Dependence of Reward Curves and Behavioral Statistics on Initial Random Seeds. Rewards and behavioral statistics are very consistent early in training, but more variable at later times.

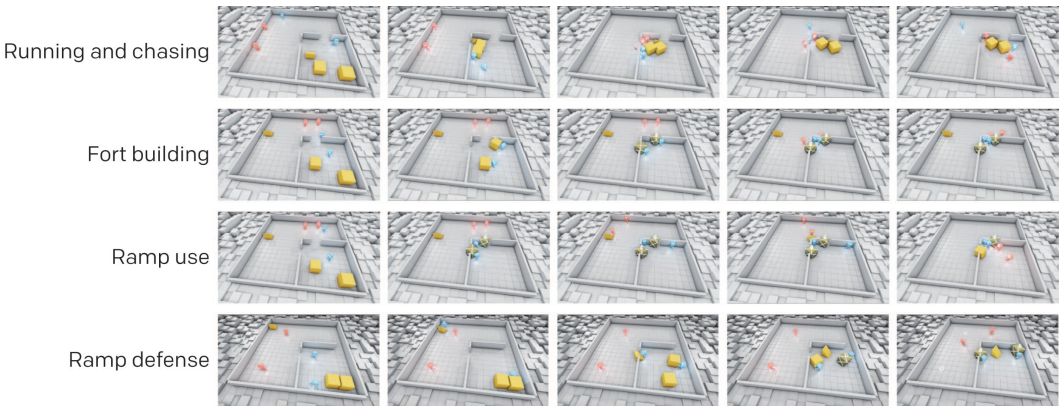

Figure A.3: Sample Trajectory Traces From Each Stage of Emergent Strategy in the Quadrant Environment. Rows correspond to different strategies, columns to characteristic snapshots in chronological order within an episode that showcase the strategy. *Running and chasing*: Hiders (blue) try to hide behind walls and moveable objects, seekers (red) chase and search for hiders. *Fort building:* During the preparation phase (first 3 snapshots), hiders push the two boxes into the two doors of the lower right room and lock them in place. Seekers are not able to overcome this barrier during the remainder of the trial. *Ramp use:* During the preparation phase, hiders secure the room using the two boxes (1st and 2nd snapshot). After the preparation phase, a seeker grabs the ramp (3rd snapshot), drags it towards the walls of the room (4th snapshot) and both seekers use the ramp to enter the room (5th snapshot). *Ramp defense:* Hiders prevent ramp use by grabbing the ramp during the preparation phase (2nd snapshot), pulling it inside the room (3rd snapshot), and then secure the doors using the 2 blocks (4th snapshot). After the preparation phase, the seekers are not able to enter the room (5th snapshot). Also, note that while one hider brings the ramp into the room, the other hider drags a box out of corner and then passes it to its team mate (3rd and 4th snapshot), evidence of sophisticated coordination between hiders.

spawned inside or outside the room. Both hiders and seekers receive a negative reward if they leave the environment or if they bring an object outside the environment. Also, in contrast to the main environment, only boxes, but not ramps can be locked in place.

The quadrant environment gives rise to a similar skill progression as in the main environment (Figure A.3): basic running and chasing, then fort building by hiders, ramp use by seekers, and ramp defense by hiders. Hiders create shelter by pushing boxes into the doors of the room in the lower right corner, preventing seekers from entering. Seekers drag ramps to the exterior walls of the room and then use them to jump inside. As ramps are not lockable in this environment, hiders instead defend against ramp use by pulling the ramp inside the room before securing the wall opening with boxes.

A.4 Further Ablations

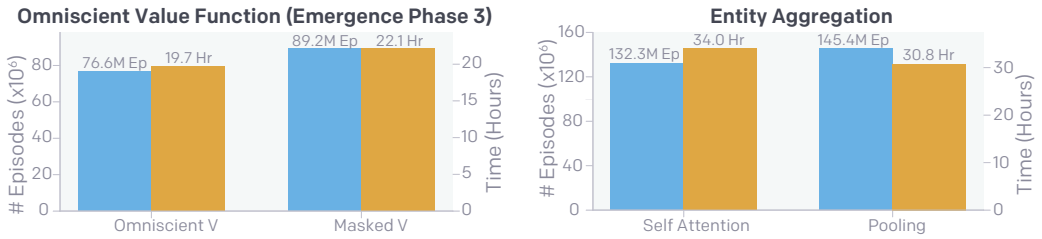

Figure A.4: Effect of Omniscient Value Function and Pooling Architectures on Emergent Autocurricula. In blue we show the number of episodes required and in orange the wall clock time required to achieve stage 4 (ramp defense) of the emergent skill progression presented in Figure 1 for varying batch sizes. Note that in our ablation comparing omniscient and masked value functions, the experiment with a masked value function never reached stage 4 in the allotted time, so here we compare timing to stage 3.

In Figure A.4 we compare the performance between a masked and omniscient value function as well a purely pooling architecture versus self-attention. We find that using an omniscient value function, meaning that the value function has access to the full state of the unobscured environment, is critical to progressing through the emergent autocurricula at the given scale. We found that with the same compute budget, training with a masked value function never progressed past stage 3 (ramp usage). We further found that our self-attention architecture increases sample efficiency as compared to an architecture that only embeds and then pools entities together with a similar number of parameters. However, because self-attention requires more compute despite having the same number of parameters, the wall-clock time to convergence is slightly slower.

A.5 Evaluating Agents at Different Phases of Emergence

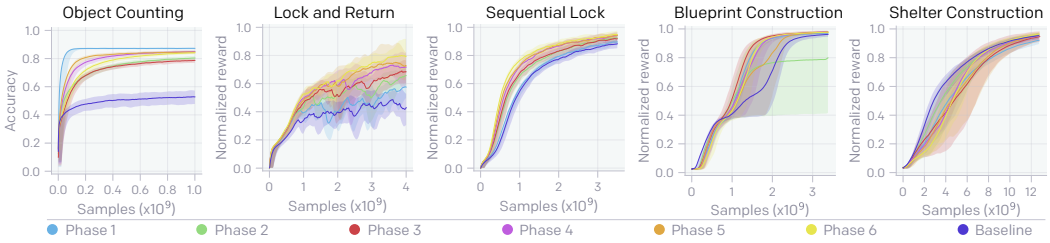

Figure A.5: Fine-tuning From Different Phases in the Emergent Autocurriculum. We plot the mean performance on the transfer suite and 90% confidence interval across 3 seeds and smooth performance and confidence intervals with an exponential moving average. We show the fine-tuning performance for a policy sampled at each of the six phases of emergent strategy (see Figure 1).

In Figure A.5 we evaluate policies sampled during each phase of emergent strategy on the suite of targeted intelligence tasks, by which we can gain intuition as to whether the capabilities we measure improve with training, are transient and accentuated during specific phases, or generally uncorrelated to progressing through the autocurriculum. We find that the hide-and-seek agent improves on the navigation and memory tasks as it progresses; notably on Lock and Return, the performance

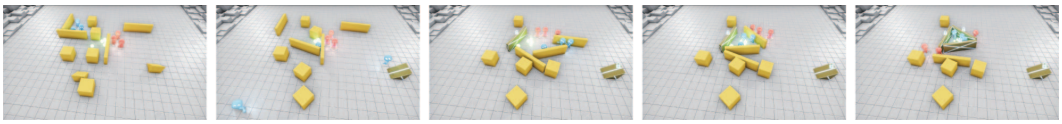

Figure A.6: Example trajectory in hide-and-seek environment with additional food reward. During the preparation phase, hiders lock the ramps at the boundary of the play area (2nd snapshot) and construct a fort around the food reward (3rd and 4th snapshot). After the preparation phase, hiders can eat food and receive the additional food reward, because they are hidden from seekers inside the fort.

monotonically increases with emergence phase, and the policy from the phase 6 performs 20% better than the policy from phase 1. However, performance on Object Counting is transient; during phase 1 the hide-and-seek agent performs extremely well, much better than all baselines and other phases but loses this ability in later stages. Finally, we find that performance on the manipulation tasks is relatively uncorrelated to the phases of emergence, and surprisingly the policy transferred from phase 1, the phase before any tool use emerges, performs comparably well to other phases.

A.6 Alternative Games to Hide-and-Seek with Secondary Objectives

A.6.1 Hide-and-Seek with Food Reward

In the main hide-and-seek environment, hiders build forts at locations that would give them the best chance of staying hidden from seekers. To test whether hiders could be incentivized by a secondary objective to adapt the location of forts, we add additional food rewards and test whether hiders would bias the location of their forts towards the location of the food. In this environment, hiders can eat food and receive food rewards only under the following conditions: after the preparation phase, when all hiders are hidden from seekers and when the food is sufficiently close and visible to them. Therefore, just chasing after food would not be an effective strategy for hiders, because hiders receive neither hide-and-seek nor food rewards if one or more hiders are seen by a seeker. Instead, hiders are incentivized to build forts around the location of food and then eat the food while being unobserved by seekers (figure A.6).

Food is distributed in the form of 5 food pellets that are spawned close to each other in a rectangular area in the center of the environment whose side length is 1/4 of the room size. Each food pellet can provide a positive food reward of +1 for each time step. Food rewards are shared between hiders, irrespective of which hider eats a food item. Hiders (as well as seekers) can observe the locations of food pellets as separate entities; if food pellets are obstructed or outside the field of view they are masked out like other type of objects.

As shown in figure A.7, this environment gives rise to four levels of skill progression, similar to the one of the main environment: basic running and chasing, then fort building by hiders, ramp use by seekers, and ramp defense by hiders. Moreover, hiders consume food, and food consumption is highly correlated with their ability to construct stable forts; food consumption decreases during the initial phase as seekers get better at chasing and therefore prevent hiders from eating food. Food consumption then increases again as hiders learn to construct forts and shield themselves from the view of seekers, but plateaus once seekers learn to use ramps. Finally, food consumption rises again as hiders get better at defending against ramp use.

A.6.2 Hide-and-Seek with Dynamic Food

In the food variant introduce in Sec. A.6.1, the location of food pellets is fixed throughout the episode. Here we consider a dynamic food variant such that a food pellet will be eaten up, i.e., disappear, when a hider is close to it, and then a new food pellet will show up in a different location but still within the center region. More precisely, the game area is simply an empty room without outside walls containing 2 seekers, 3 hiders, 1 dynamic food and 8 elongated boxes. The food will be always located within a square in the center of the environment with side length 1/5 of the game area size. We inherit the same reward and policy structures from the previous game.

In this game, merely building a center fort is not sufficient for obtaining the highest reward since the food might disappear and respawn outside the fort. The agent must ensure that the fort is large enough

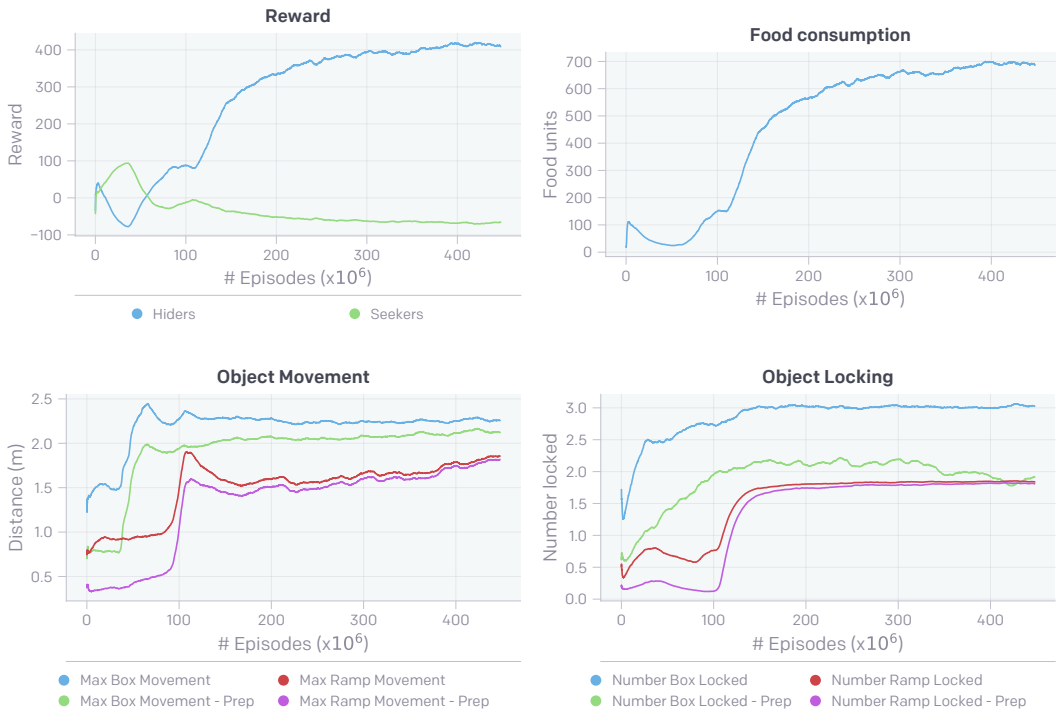

Figure A.7: Reward and environment specific statistics during emergence in hide-and-seek with secondary food rewards for hiders.

such that all the possible spawning positions of the food will be inside the fort. Such a behavior does emerge after training for around 4.5×10^{10} samples.

We also experimented on variants where the food spawning region has different side length. When the side length is reduced to 1/6 of the game area, the same behavior emerges faster taking 1.5×10^{10} samples. However, when the side length of the dynamic food region is increased to 1/4 of the game area, hiders converge to a policy that ignores the food and only builds a small fort to protect themselves.

A.6.3 Food Protection Game

In the previous two variants, we introduce extra food reward to the hider in addition to the original hide-and-seek reward for promoting more goal-oriented behavior. Now we consider a different game rule such that the competition between hiders and seekers only depends on the food collecting reward and show that this rule can also lead to tool use and complex behavior.

We consider an empty game area surrounded by walls that contains 50 food pellets randomly distributed in a center square of size 2/3 of the game environment. There are 3 hiders, 2 seekers and 7 elongated boxes in the game. The only goal for seekers is to collect food. Once a food pellet is collected by any of the seeker, a +3 reward will be given to all the seekers and then the food will disappear permanently from the game. The goal for the hider is to protect the food from seekers and their reward is simply the negative value of seekers. Each episode consists of 200 time steps. The preparation period, in which only hiders can move, extends to the first 100 time steps. The last 60 steps of the game corresponds to a food collecting period, during which the hiders cannot move. Additionally, after the preparation phase we also add a -1 "boundary penalty" for hiders when they are too close to the wall to ensure they stay within the food region.

Our initial motivation for this task was to promote hiders to learn to construct complex and large fort structures with a more direct competition pressure. For example, we expected that the agents would learn to build a large fort that would cover as much food as possible. Unfortunately, such behavior did not emerge. However, after around 7×10^10 samples of training, the hiders surprisingly discover an extremely efficient and effective strategy: they learn to physically 'push' the immobile seekers to the corner of the environment during the preparation phase and then collaboratively use

Figure A.8: Example trajectory in the food protection game. During the preparation phase, the hiders first explore the room to find seekers, then physically push the seekers to the corner and eventually collaboratively grab elongated boxes to build a small fort surrounding the seekers to trap them there. To secure the trap, they even learn to build the fort using *two* elongated boxes to prevent seekers from escaping.

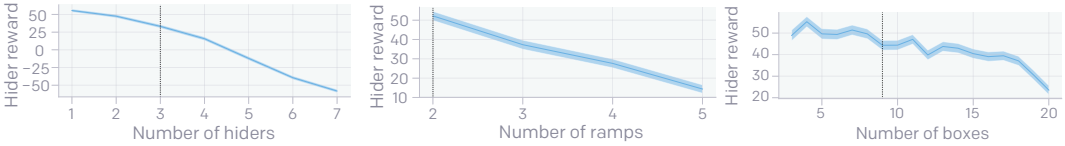

Figure A.9: Zero-shot generalization to a larger number of hiders (left), ramps (center) and boxes (right). The dotted line denotes the boundary of the training distribution (1-3 hiders, 2 ramps, 1-9 boxes). Error bars denote standard error of the mean.

elongated boxes to trap the seekers in the corner. Figure A.8 illustrates a trajectory corresponding to this strategy. Interestingly, the hiders demonstrate strong collaborations. Since there are 2 seekers and 3 hiders, when the 2 hiders are pushing the seekers to the corner, the 3rd hider will follow the 'pushing' hiders distantly, and once the pushing hiders finish the job, this 3rd hider will quickly grab a box to build the trap without blocking the escaping route of those 2 pushing hiders. We emphasize that this behavior was completely surprising to us and further indicates that complex behavior can emerge from multi-agent self-play with a wide range of competition rules.

Lastly, we also experimented with different versions of the close-to-wall "boundary penalties". In all cases, we observe similar fort-building behaviors, namely blocking the seekers to the corner, with tiny variations. When we remove the penalty, the behavior emerges twice faster (3.5×10^{10} samples) and 1 hider will stay in the fort with the seekers instead of getting outside. When we add the penalty throughout the episode, the hiders even learned to stay in the center region while grabbing an elongated box as a tool to push the seekers towards the corner. However, the behavior emerges twice slower in this setting, i.e., 1.4×10^{11} samples required.

A.7 Zero-shot generalization

We find that the trained policies can zero-shot generalize to environments with slightly larger number of hiders, ramps or boxes in Figure A.9. The hider reward provides a reasonably good measure for generalization performance, since the modifications in question make constructing safe shelter more difficult. First, we looked at increasing the number of hiders from 1-3 hiders, as in the training distribution, to 4-6 hiders; increasing the number of hiders decreases the hider reward because all hiders need to stay hidden from seekers. However, the decline in hider reward is very gradual, indicating that the policy generalizes well to more than 4 hiders. A similar effect occurs when increasing the number of ramps because hiders need to secure more ramps from seekers. If we increase the number of ramps from 2 to 3 or 4 the hider reward drops only gradually. Finally, we find hider performance is remarkably stable, though still slowly declines, when increasing the number of boxes.

B Optimization Details

B.1 Notation

We consider the standard multi-agent reinforcement learning formalism of N agents interacting with each other in an environment. This interaction is defined by a set of states $\mathcal{S}$ describing the state of the world and configurations of all agents, a set of observations $\mathcal{O}^1, \dots \mathcal{O}^N$ of all agents, a set of actions $\mathcal{A}^1, \dots, \mathcal{A}^N$ of all agents, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \dots \mathcal{A}^N \rightarrow \mathcal{S}$ determining the distribution over next states, and a reward for each agent i which is a function of the state and

the agent's action $r^i : \mathcal{S} \times \mathcal{A}^1 \dots \mathcal{A}^N \to \mathbb{R}$. Agents choose their actions according to a stochastic policy $\pi_{\theta_i} : \mathcal{O}^i \times \mathcal{A}^i \to [0, 1]$, where θ_i are the parameters of the policy. In our formulation, policies are shared between agents, $\pi_{\theta_i} = \pi_\theta$ and the set of observations $\mathcal{O}$ contains information for which role (e.g. hider or seeker) the agent will be rewarded. Each agent i aims to maximize its total expected discounted return $R^i = \sum_{t=0}^{H} \gamma^t r_t^i$, where H is the horizon length and γ is a time discounting factor that biases agents towards preferring short term rewards to long term rewards. The *action-value function* is defined as $Q^{\pi_i}(s_t, a_t^i) = \mathbb{E}[R_t^i | s_t, a_t^i]$, while the *state-value function* is defined as $V^{\pi_i}(s_t) = \mathbb{E}[R_t | s_t]$. The *advantage function* $A^{\pi_i}(s_t, a_t^i) := Q^{\pi_i}(s_t, a_t^i) - V^{\pi_i}(s_t)$ describes whether taking action a_t^i is better or worse for agent i when in state s_t than the average action of policy π_i.

B.2 Proximal Policy Optimization (PPO)

Policy gradient methods aim to estimate the gradient of the policy parameters with respect to the discounted sum of rewards, which is often non-differentiable. A typical estimator of the policy gradient is $g := \mathbb{E}[\hat{A}_t \nabla_\theta \log \pi_\theta]$, where $\hat{A}$ is an estimate of the advantage function. PPO [58], a policy gradient variant, penalizes large changes to the policy to prevent training instabilities. PPO optimizes the objective $L = \mathbb{E}\left[\min(l_t(\theta)\hat{A}_t, \text{clip}(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right]$, where $l_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ denotes the likelihood ratio between new and old policies and $\text{clip}(l_t(\theta), 1-\epsilon, 1+\epsilon)$ clips $l_t(\theta)$ in the interval $[1-\epsilon, 1+\epsilon]$.

B.3 Generalized advantage estimation

We use Generalized Advantage Estimation [3] with horizon length H to estimate the advantage function. This estimator is given by:

$$\hat{A}_t^H = \sum_{l=0}^{H} (\gamma\lambda)^l \delta_{t+l}, \qquad \delta_{t+l} := r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l})$$

where δ_{t+l} is the TD residual, $\gamma, \lambda \in [0, 1]$ are discount factors that control the bias-variance tradeoff of the estimator, $V(s_t)$ is the value function predicted by the value function network and we set $V(s_t) = 0$ if s_t is the last step of an episode. This estimator obeys the reverse recurrence relation $\hat{A}_t^H = \delta_t + \gamma\lambda \hat{A}_{t+1}^{H-1}$.

We calculate advantage targets by concatenating episodes from policy rollouts and truncating them to chunks of $T = 160$ time steps (episodes contain 240 time steps). If a chunk $(s_0, \dots, s_{T-1})$ was generated in a single episode we use the advantage targets $(\hat{A}_0^{H=T}, \hat{A}_1^{H=T-1}, \dots, \hat{A}_{T-1}^{H=1})$. If a new episode starts at time step j we use the advantage targets $(\hat{A}_0^{H=j}, \hat{A}_1^{H=j-1}, \dots, \hat{A}_{j-1}^{H=1}, \hat{A}_j^{H=T-j}, \dots, \hat{A}_{T-1}^{H=1})$.

Similarly we use as targets for the value function $(\hat{G}_0^{H=T}, \hat{G}_1^{H=T-1}, \dots, \hat{G}_{T-1}^{H=1})$ for a chunk generated by a single episode and $(\hat{G}_0^{H=j}, \hat{G}_1^{H=j-1}, \dots, \hat{G}_{j-1}^{H=1}, \hat{G}_j^{H=T-j}, \dots, \hat{G}_{T-1}^{H=1})$ if a new episode starts at time step j, where the return estimator is given by $\hat{G}_t^H := \hat{A}_t^H + V(s_t)$. This value function estimator corresponds to the TD(λ) estimator [67].

B.4 Normalization of observations, advantage targets and value function targets

We normalize observations, advantage targets and value function targets. Advantage targets are z-scored over each buffer before each optimization step. Observations and value function targets are z-scored using a mean and variance estimator that is obtained from a running estimator with decay parameter $1 - 10^{-5}$ per optimization substep.

B.5 Optimization setup

Agents are trained using self-play, which acts as a natural curriculum as agents always play opponents of an appropriate level. Policies are composed of two separate networks with different parameters – a policy network which produces an action distribution and a value function network which predicts the

discounted future returns. During each episode, players have a 5% chance of using a policy uniformly sampled from past versions, which is commonly used to improve policy robustness [68, 69, 12].

Training is performed using the distributed *rapid* framework [9]. Using current policy and value function parameters, CPU machines roll out the policy in the environment, collect rewards, and compute advantage and value function targets. Rollouts are cut into chunks of 160 timesteps and reformatted into 16 blocks of 10 timesteps (the BPTT truncation length). The rollouts are then collected in a training buffer of 320,000 blocks. Each optimization step consists of 60 SGD substeps using Adam with mini-batch size 64,000. One rollout block is used for at most 4 optimization steps. This ensures that the training buffer stays sufficiently on-policy.

B.6 Optimization hyperparameters

Optimization hyperparameter settings are shown in Table 1.

Buffer size	320,000
Mini-batch size	64,000 blocks of 10 timesteps
Learning rate	$3 \cdot 10^{-4}$
PPO clipping parameter ϵ	0.2
Gradient clipping	5
Entropy coefficient	0.01
γ	0.998
λ	0.95
Max GAE horizon length T	160
BPTT truncation length	10

Table 1: Optimization hyperparameters

B.7 Policy architecture

We utilize entity-centric observations [70, 71] and use attention mechanisms to capture object-level information [72, 73]. As shown in Figure B.1 we use a self-attention [74] based policy architecture over entities, which is permutation invariant and generalizes to varying number of entities.

Lidar observations are first passed through a circular 1D-convolution and concatenated onto the agents representation of self, x_{self}. Each object is concatenated with x_{self} and then embedded with a dense layer where parameters are shared between objects of the same type, e.g. all boxes share the same embedding weights. All the embedded entities are then passed through a residual self-attention block, similar to [74] but without position embeddings, in the form of $y = \text{dense}(\text{self\_attention}(x)) + x$. We then average-pool entity embeddings and concatenate this pooled representation to x_{self}. Note that in the policy network the entities not observed by each agent are masked away through self-attention and pooling. Finally, this pooled representation is passed through another dense layer and an LSTM [75] before pulling off separate action heads for each of the 3 action types described in Section 3. We also add layer normalization [76] to every hidden layer of the policy network except the 1D-convolution layer. We empirically observe that layer normalization leads to faster training and better transfer performance.

Further policy architecture hyperparameters are specified in Table 2.

Size of embedding layer	128
Size of MLP layer	256
Size of LSTM layer	256
Residual attention layer	4 attention heads of size 32
Weight decay coefficient	10^{-6}

Table 2: Policy architecture hyperparameters

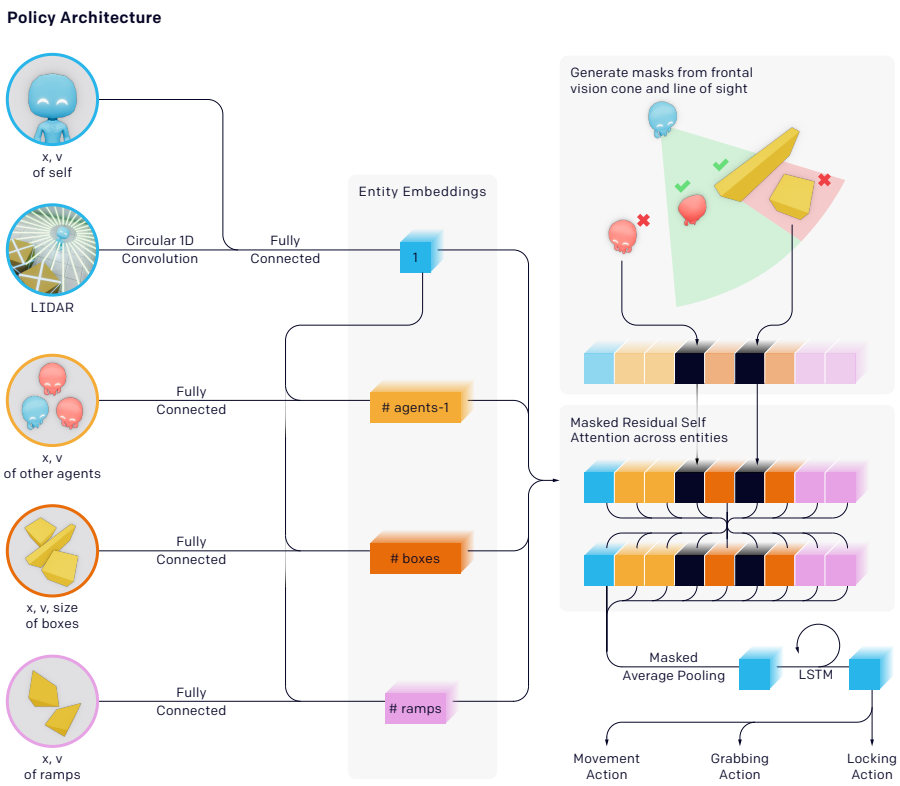

Figure B.1: Agent Policy Architecture.

C Intelligence Test Suite Details

All evaluation tasks in the intelligence test suite are single-agent tasks. In order to align the observations between the hide-and-seek environment and the evaluation tasks, we add fake hider observations, preparation phase observations and set them all to 0.

C.1 Cognition and memory task

All the variations of cognition and memory tasks have a horizon of 120 timesteps, and all boxes are cuboids.

Object counting: An agent is pinned in place and watches as 6 boxes move either to left or right at random. The boxes disappear behind walls such that agent can no longer see them, and the agent is asked to predict how many boxes went left or right far after all the boxes have disappeared. In this test we evaluate the quality of the *existing* representation by holding the agent's policy fixed and only train a new classification head from the agent's LSTM state. The classification head first passes the LSTM state through a layernorm and a single dense layer with 64 units. We then do a 7-class classification predicting whether 0 through 6 boxes have gone to the left.

Lock and return: In the *lock and return* game, the agent needs to navigate towards a hidden box, lock it and then return to its starting position.

The game area has 6 randomly generated connected rooms with static walls and 1 box. When the box is locked, the agent will be given a reward of +5. If the agent unlocks the box during the episode, a -5 penalty will be given. Additionally, if the box remains unlocked at the end of the episode, the agent will be given another -5 penalty. A success is determined when the agent returns to its starting location within 0.1 radius and with the box locked. For promoting fast task accomplishment, we give the agent a +1 reward for each timestep of success. We also introduce shaped reward with coefficient

of 0.5 for easier learning: at each time step, the shaped reward is the decrement in the distance between the agent location towards the target (either the unlocked box or the starting location).

Sequential lock: In the *sequential lock* game, there are 4 boxes and the agent needs to lock all the boxes in an unobserved order sequentially. A box can be locked only if it is locked in the right order.

The game area is randomly partition into three rooms with 2 walls. The 4 boxes are randomly placed in the game area. Each room has a ramp. The agent has to utilize the ramps to navigate between rooms. When a box is successfully locked (according to the order), a +5 bonus is given. If a box is unlocked, -5 penalty will be added. When all the boxes get locked, the agent will receives a +1 per-timestep success bonus. We also use the same shaped distance reward as the *lock and return* task here.

C.2 Manipulation task

All variations of the manipulation task have 8 boxes, but no ramps.

Construction from blueprint:

The horizon is at most 240 timesteps, but an episode can end early if the agent successfully finishes the construction. The game area is an empty room. The locations of the construction sites are sampled uniformly at random (we use rejection sampling to ensure that construction sites do not overlap).

For each construction site, agents observe its position and that of its 4 corners. Since there are no construction sites in the hide-and-seek game and the count-based baseline environments, we need to change our policy architecture to integrate the new observations. Each construction site observation is concatenated with x_{self} and then embedded through a new dense layer shared across all sites. This dense layer is randomly initialized and added to the multi-agent and count-based policies before the start of training. The embedded construction site representations are then concatenated with all other embedded object representations before the residual self-attention block, and the rest of the architecture is the same as the one used in the hide-and-seek game.

The reward at each timestep is equal to a reward scale constant times the mean of the smooth minimum of the distances between each construction site corner and every box corner. Let there be k construction sites and n boxes, and let d_{ij} be the distance between construction site corner i and box corner j, and let d_i be the smooth minimum of the distances from construction site corner i to all box corners. The reward at each timestep follows the following formula:

$$
\begin{aligned}
d_i &= \left(\sum_{j=1}^{4n} d_{ij} e^{\alpha d_{ij}} \right) / \sum_{j=1}^{4n} e^{\alpha d_{ij}} \quad \forall i = 1, 2, \ldots, 4k \\
rew &= s_d \left(\sum_{i=1}^{4k} d_i \right) / 4k
\end{aligned}
$$

Here, s_d is the reward scale parameter and α is the smoothness hyperparameter (α must be non-positive; $\alpha = 0$ gives us the mean, and $\alpha \to -\infty$ gives us the regular min function). In addition, when all construction sites have a box placed within a certain distance d_{min} of them, and all construction site corners have a box corner located within d_{min} of them, the episode ends and all agents receive reward equal to $s_c * k$, where s_c is a separate reward scale parameter. For our experiment, $n = 8$ and k is randomly sampled between 1 and 4 (inclusive) every episode. The hyperparameter values we use for the reward are the following:

$$
\begin{aligned}
\alpha &= -1.5 \\
s_d &= 0.05 \\
d_{min} &= 0.1 \\
s_c &= 3
\end{aligned}
$$

Shelter construction: The goal of the task is to build a shelter around a cylinder that is randomly placed in the playing field. The horizon is 150 timesteps, and the game area is an empty room. The location of the cylinder is uniformly sampled at random a minimum distance away from the edges of the room (this is because if the cylinder is too close to the external walls of the room, the agents are physically unable to complete the whole shelter). The diameter of the cylinder is uniformly

randomly sampled between d_{min} and d_{max}. There are 3 movable elongated boxes and 5 movable square boxes. There are 100 rays that originate from evenly spaced locations on the bounding walls of the room and target the cylinder placed within the room. The reward at each timestep is $(-n * s)$, where n is the number of raycasts that collide with the cylinder that timestep and s is the reward scale hyperparameter.

We use the following hyperparameters:

$$\begin{aligned} s &= 0.001 \\ d_{min} &= 1.5 \\ d_{max} &= 2 \end{aligned}$$

D Intrinsic Motivation Methods

We inherit the same policy architecture as well as optimization hyperparameters as used in the hide-and-seek game.

Note that only the *Sequential Lock* task in the transfer suite contains ramps, so for the other 4 tasks we remove ramps in the environment for training intrinsic motivation agents.

D.1 Counted-Based Exploration

For each real value from the continuous state of interest, we discretize it into 30 bins. Then we randomly project each of these discretized integers into a discrete embedding of dimension 16 with integer value ranging from 0 to 9. Here we use discrete embeddings for the purpose of accurate hashing. For each input entity, we concatenate all its obtained discrete embeddings as this entity's feature embedding. An max-pooling is performed over the feature embeddings of all the entities belonging to each object type (i.e., agent, lidar, box and ramp) to obtain a entity-invariant object representation. Finally, concatenating all the derived object representations results in the final state representation to count.

We run a decentralized version of the count-based exploration where each parallel rollout worker shares the same random projection for computing embeddings but maintains its own counts. Let $N(S)$ denote the counts for state S in a particular rollout worker. Then the intrinsic reward is calculated by $\frac{0.1}{\sqrt{N(S)}}$.

D.2 Random Network Distillation

Random Network Distillation (RND) [41] uses a fixed random network, i.e., a target network, to produce a random projection for each state while learns anther network, i.e., a predictor network, to fit the output from the target network on visited states. The prediction error between two networks is used as the intrinsic motivation.

For the random target network, we use the same architecture as the value network except that we remove the LSTM layer and project the final layer to a 64 dimensional vector instead of a single value. The architecture is the same for predictor network except that each hidden layer is only a quarter of the original size to increase the prediction difficulty. We use the squared difference between predictor and target network output with an coefficient of 0.1 as the intrinsic reward.

D.3 Fine-tuning From Intrinsic Motivation Variants

We compare the performances of different policies pretrained by intrinsic motivation variants on our intelligence test suites in Figure D.1. The pretraining methods of consideration include RND and 3 different count-based exploration variants with different state representations. Generally, the policies pretrained by RND performs consistently the worst among all the tasks. For policies trained by count-based variants, as discussed in the Sec. 6.1, we know that more concise state representation for counts leads to better emergent object manipulation skills, i.e., only using object position is better than using position and velocity information while using all the input as state representation performs the worst. In our transfer task suites, we observe that polices with better pretrained skills perform better in 3 of the 5 tasks except the *object counting* cognitive task and the *blueprint construction*

task. In *blueprint construction*, policies with better skills adapt better in the early phase but may have a higher chance of failure later in this challenging task. In *object counting*, the polices with better skills perform worse. Interestingly, this observation on the object-counting task is consistent with the transfer result in the main paper (Figure 5), where the policy pretrained by count-based exploration outperforms the multi-agent pretrained policy. We conjecture that the *object counting* task examines some factors of the agents that may not strongly relate to the quality of emergent skills, such as navigation and tool use.

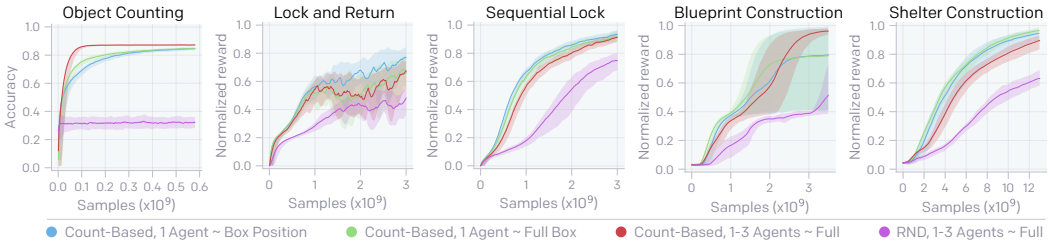

Figure D.1: Fine-tuning From Intrinsic Motivation Variants. We plot the mean performance on the suite of transfer tasks and 90% confidence interval across 3 seeds and smooth performance and confidence intervals with an exponential moving average. We vary the state representation used for collecting counts: in blue we show the performance for a single agent where the state is defined on box 2-D positions, in green we show the performance of a single agent where the state is box position but also box rotation and velocity, in red we show the performance of 1-3 agents with the full observation space given to a hide-and-seek policy, and finally in purple we show the performance also with 1-3 agents and a full observation space but with RND which should scale better than count-based exploration.