# NEURAL MMO: A MASSIVELY MULTIPLAYER GAME ENVIRONMENT FOR INTELLIGENT AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We present an artificial intelligence research platform inspired by the human game genre of MMORPGs (Massively Multiplayer Online Role-Playing Games, a.k.a. MMOs). We demonstrate how this platform can be used to study behavior and learning in large populations of neural agents. Unlike currently popular game environments, our platform supports persistent environments, with variable number of agents, and open-ended task descriptions. The emergence of complex life on Earth is often attributed to the arms race that ensued from a huge number of organisms all competing for finite resources. Our platform aims to simulate this setting in microcosm: we conduct a series of experiments to test how large-scale multiagent competition can incentivize the development of skillful behavior. We find that population size magnifies the complexity of the behaviors that emerge and results in agents that out-compete agents trained in smaller populations.

## 1 INTRODUCTION

Life on Earth can be viewed as a massive multiagent competition. The cheetah evolves an aerodynamic profile in order to catch the gazelle, the gazelle develops springy legs to run even faster: species have evolved ever new capabilities in order to outcompete their adversaries.

The success of biological evolution has inspired many attempts to emulate it in silico, ranging from genetic algorithms that bear only loose resemblance to natural processes, to full-blown simulations of "artificial life". A recurring question has been: at what level of abstraction should we simulate the competitive game of life?

In recent years, the field of deep reinforcement learning (RL) has embraced a related approach: train algorithms by having them compete in simulated games (Silver et al., 2016; susan zhang, 2018; Jaderberg et al., 2018). Such games are immediately interpretable and provide easy metrics derived from the game's "score" and win conditions. However, popular game benchmarks are currently still limited: they typically define a narrow, episodic task, with a small fixed number of players. In contrast, life on Earth involves a persistent environment, an unbounded number of players, and a seeming "open-endedness," where ever new and more complex species emerge over time, with no end in sight (Stanley et al., 2017).

Our aim is to develop a simulation platform (see Figure 1) that captures important properties of life on Earth, while also borrowing from the interpretability and abstractions of human-designed games. To this end, we turn to the game genre of Massively Multiplayer Online Role-Playing Games (MMORPGs, or MMOs for short). These games involve a large, variable number of players competing to survive and prosper in persistent and far-flung environments. Our platform simulates a "Neural MMO" – an MMO in which each agent is a neural net that learns to survive using RL.

We demonstrate the capabilities of this platform through a series of experiments that investigate emergent complexity as a function of the number of agents and species that compete in the simulation. We find that large populations act as competitive pressure that encourages exploration of the environment and the development of skillful behavior. In addition, we find that when agents are organized into species (share policy parameters), each species naturally diverges from the others to occupy its own behavioral niche. Upon publication, we will opensource the platform in full.

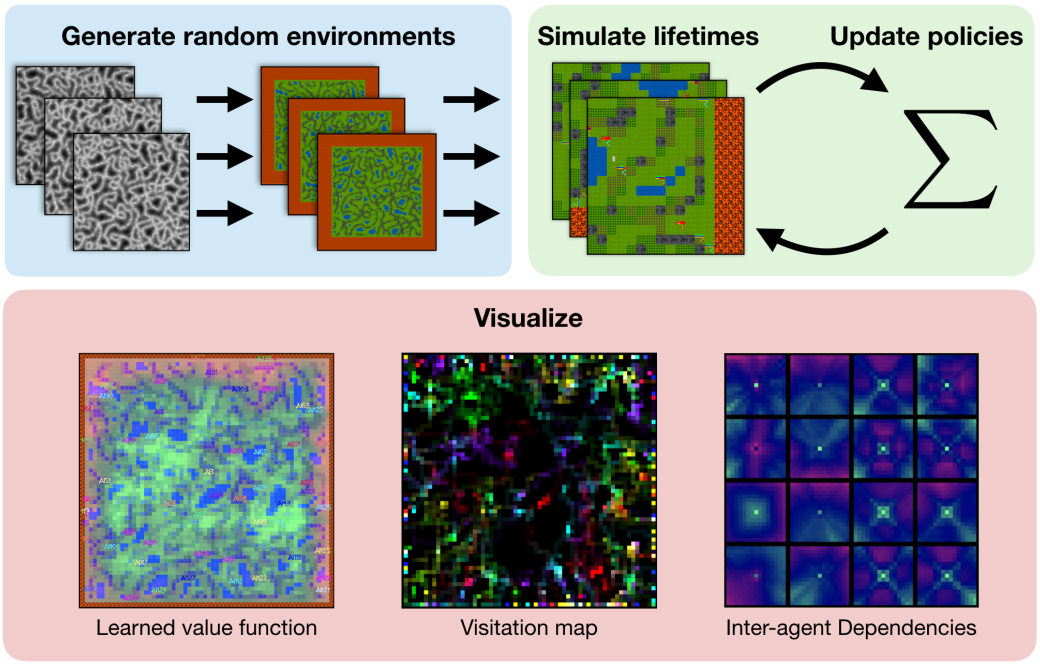


Figure 1: Neural MMO pipeline. We alternate between collecting experience across 100 procedurally generated worlds and updating parameters. Test time visualization provides insight into learned policies through value estimates, exploration patterns, and agent-agent dependence.

## 2 Background and Related Work

**Multiagent Reinforcement Learning** Multiagent reinforcement learning has received increased attention in recent years (Lowe et al., 2017; Mordatch & Abbeel, 2017; Bansal et al., 2017; Lanctot et al., 2017; Yang et al., 2018a; Zheng et al., 2017; susan zhang, 2018; Jaderberg et al., 2018). Unlike single agent environments often well modeled by classic Markov Decision Processes (MDPs), multiagent interaction often introduces nonstationarity in the transition dynamics of the environment due to the continual learning and co-adaptation of other agents.

While previous work has attempted to analyze emergent complexity in interactions of groups of 2-10 agents (Bansal et al., 2017; susan zhang, 2018; Jaderberg et al., 2018), we focus on large populations of agents in a complex world. Zheng et al. (2017) also analyze behavior of many agents, but the task setting is considerably simpler, and agents are directly rewarded for foraging or fighting. In contrast, our work focuses on complexity in a diverse environment where agents are only rewarded for survival; doing so requires maintaining their health through food and water while navigating partially obstructed terrain and with various forms of combat. Yang et al. (2018b) also simulate populations scaling to millions of learning agents. However, they study only predator-prey population dynamics in this setting, finding similarities to predator-prey dynamics in nature.

**Artificial Life** "Artificial life" projects aim to simulate the complexity of biological life (Langton, 1997), often framed as a multiagent competition, and have demonstrated that this setting can lead to the development of skilled behaviors (Yaeger, 1994) and capable morphologies (Sims, 1994). The setting we consider, in which an agent coadapts alongside others is sometimes called coevolution (Ficici & Pollack, 1998).

Our platform can simulate a kind of artificial life, but at a relatively high level of abstraction, importing design elements from games. Unlike most past work in this area, our platform is built to around deep reinforcement learning, where each agent is a neural net trained with distributed policy gradients, and includes tools for visualizing properties specific to this setting.

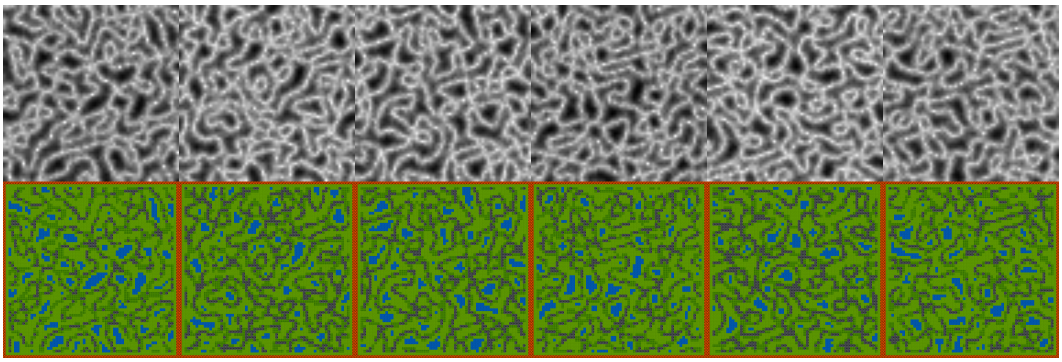

Figure 2: We procedurally generate maps by thresholding an 8 octave Perlin (Perlin, 1985) ridge fractal. We map tile type to a fixed range of values

**Game Platforms for Intelligent Agents** The Arcade Learning Environment (ALE) (Bellemare et al., 2013) and Gym Retro (Nichol et al., 2018) provide 1000+ limited scope arcade games most often used to test individual research ideas or generality across many games. Strictly better performance at a large random subset of games is a reasonable metric of quality, and strikingly divergent performance can motivate further research into a particular mode of learning. For example, Montezumas Revenge in ALE is difficult even with reasonable exploration because of the associative memory required to collect and use keys. However, recent results have brought into question the overall complexity each individual environment (Cuccu et al., 2018) and strong performance in such tasks (at least in those not requiring precise reflexes) is not particularly difficult for humans.

More recent work has demonstrated success on multiplayer games including Go (Silver et al., 2016), the Multiplayer Online Battle Arena (MOBA) DOTA2 (susan zhang, 2018), and Quake 3 Capture the Flag (Jaderberg et al., 2018). Each of these projects has advanced our understanding of a class of algorithms. However, these games were limited to 2-12 players, are round based on the order of an hour, lack persistence, and lack the game mechanics supporting large persistent populations – there is still a large gap in environment complexity compared to the real world.

**RPGs** MMORPGs are the multiplayer analogs to the narrower game genre of RPGs (Role-Playing Games). The latter includes game franchises such as Pokemon, Final Fantasy, The Witcher, and Fallout. They involve reasoning on the order of several to several hundred hours–longer time horizons than MOBAs–but they lack the complexity induced by multiagent interaction and competition–this is our motivation for creating an MMO rather than an RPG. Like the real world, RPGs confront the player with problems that have many valid solutions, and choices that have long term consequences. For example, whereas the arcade game Montezumas Revenge has a single core mechanic (find keys, unlock doors), the game Prey allows players to proceed by remembering the code to an unguarded side door, acquiring weapons to fight through the main entrance, or bypassing the encounter altogether by searching out a hidden side passage. This makes well-designed RPGs compelling candidates for a variety of research ideas in isolation, but they are also well suited to simultaneously testing multiple research ideas at scale. Given multiple paths to proceed, each requiring different learned skills (e.g., exploration, memory, knowledge of game mechanics), the ability to reason over them all in conjunction allows players to choose the quickest, safest, or least costly path. As well designed MMOs with single player are effectively RPGs, this means that Neural MMOs have the potential to provide the benefits of a rich single agent environment as well.

## 3 NEURAL MMO

We present a persistent and massively multiagent environment that adopts many core MMO properties. Our platform aims to mirror the MMO lifecycle: developers typically produce a small base game, with persistent play and consistent content updates slowly shaping this restricted core into a sprawling macrocosm. Base mechanics are built upon over a rich set of challenges that produce increasingly engaging play later in the game. Skills needed for early content are easy for new players

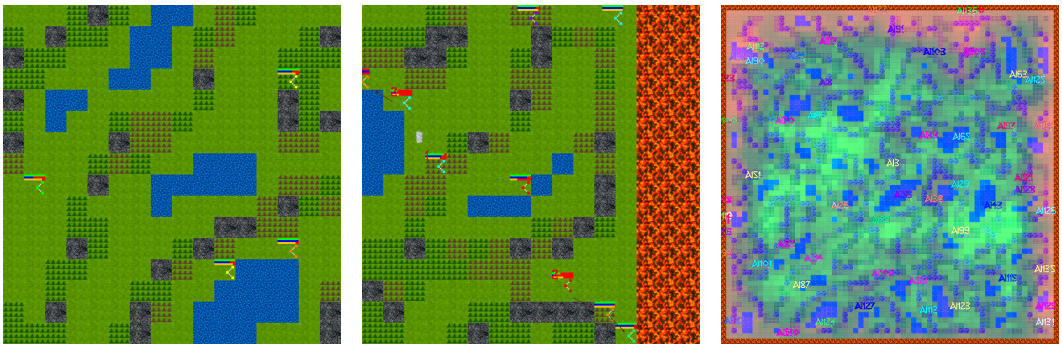

Figure 3: Left: Foraging agents learn to efficiently balance their food and water levels while competing with other agents for resources. Center: Combat agents learn the latter while also balancing melee, range, and mage attack styles to engage with and outmaneuver other agents. Right: Learned value function overlay produced via "ghosting"–faking an agent at each position and computing forward pass through the value network

to acquire, but late game content is usually unapproachable (and often incomprehensible) to those who have not progressed significantly through the game. This property acts as a natural curriculum from toy to complex that makes MMO environments amenable to scale up without saturating the task or sacrificing efficiency at small scale. This makes Neural MMOs a suitable platform for studying multiagent interaction at small scale, but as content develops, we believe the platform will grow to suit an increasing set of key problems in reinforcement learning.

This initial release includes procedural map generation as show in in Figure 2, the base environment, and two content modules: foraging and basic combat. The code base already contains additional module code for trade, gathering, crafting, item systems, equipment, communication, and trade to name a few. We are actively balancing and integrating these into the neural API.

### 3.1 Base environment

Many MMOs use an internal tile grid representation and smooth animations during rendering. We adopt this model, with agents trained on local game state. Agents observe a square crop of tiles as well as some visible properties of all agents in the corresponding tiles. Terrain tiles may be passable or impassible. We also include lava tiles, which kill entities upon contact. Agents may move one square in any of the four cardinal directions each environment timestep or "tick". We add one new agent into the game on each tick within a region of designated spawn tiles, with an optional population cap. This is only the most bare bones set of game mechanics required to run anything, somewhat equal to a tech demo of performance and functionality, but without playable content.

### 3.2 Forage

Foraging adds health, food, and water quantities to each entity. A snapshot is shown in Figure 3. Food and water decrease constantly; entities begin taking damage when they run out of either, but begin slowly healing if they are well fed and well hydrated. Water is collected at a fixed rate by standing adjacent to water tiles. Food is collected at a fixed rate by standing on forest tiles; however, these have a limited supply of food that renews slowly over time. This imposes a carrying capacity: survival is trivial with a single agent, but requires intelligent exploration in the presence of competition attempting to do the same.

### 3.3 Combat

Combat enables agents to attack each other. A snapshot is shown in Figure 3. It adds melee, ranged, and magic-based combat. Each combat style has a different maximum range of effect and damage value. This combat formulation was empirically chosen to incentivize behavior dependent on the states of other agents and to facilitate consistent and interpretable interactions.

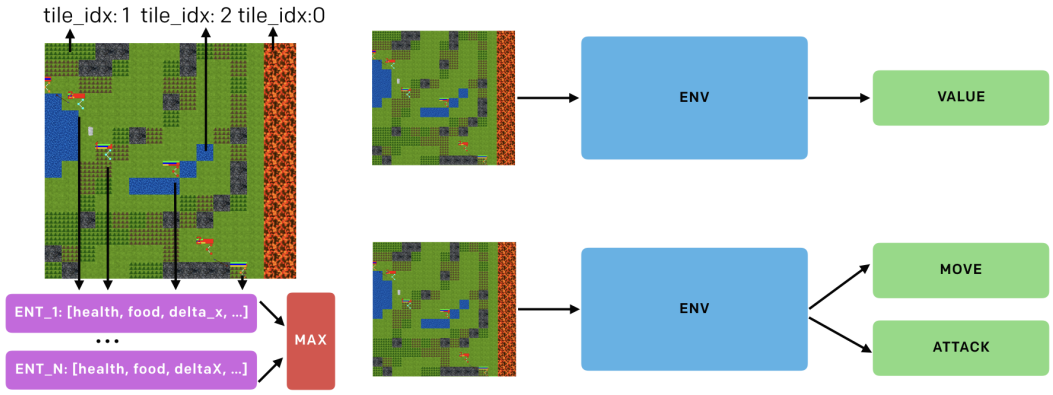


Figure 4: The model embeds local game state and computes actions via corresponding value, movement, and attack heads. These are all small fully connected networks, with input side max pooling handling variable number of entity observations. Networks have only 50-100k parameters

Agents have 10 health. Melee, ranged, and magic combat have maximum Manhattan distance of effect of 1, 2, and 3 cells respectively. They do 10, 2, and 1 damage, respectively. Magic, or "mage" combat freezes the target in place for two game ticks, preventing movement. Agents receive food/water resources equal to damage inflicted. Agents are immune during their first 15 game ticks alive. This prevents uninteresting "spawn killing" and is a common technique in human games.

## 4 Framework

The Neural MMO setting requires support for a large, variable number of agents that run concurrently. As a complex game environment, it also requires a front end visualizer in order to observe agent policies and relevant statistics.

We provide two APIs for our environment. One is a minimally modified multiagent extension of the Gym VecEnv API (Brockman et al., 2016), which distributes environment computation of observations and centralizes training and inference. While this standardization is convenient, it incurs additional communications overhead in our setting that we bypass with a fast native API. This interface is simpler to use and pins the environment and agents on it to the same CPU core. Full trajectories run locally on the same core as the environment. Interprocess communication is only required infrequently to synchronize gradients across all environments on a master core. We currently do the backwards pass on the same CPU cores for convenience, but plan on separating this into an asynchronous experience queue for GPU workers.

We also provide a renderer well suited to test time visualizations. This comes packaged with research tools for visualizing useful metrics, including value map "ghosting" as described in Figure 3, exploration maps, dependence of other agents upon other agents, and attack maps for combat (see Experiments).

## 5 Architecture and Training

To train agents, we use a policy gradient method Williams (1992) with a value function baseline. Our policy architecture consists of an environment embedding network, a movement head, and an attack head, as shown in Figure 4. The environment embedding network is a fully-connected network mapping from input stimulus to a 16 dimensional hidden vector (see below). Each head is a small fully connected network, with hidden dimensions throughout ranging from 16 to 32, for a total of 50k to 100k parameters. Environment embedding is not shared across heads. For foraging experiments, the attack network is still present for convenience, but the chosen actions are ignored. New abilities and components can be added to model by adding additional heads.

We find that, under some settings, training can be unstable. To mitigate numerical collapse, we add an entropy bonus of 0.01 during training. We have confirmed that this entropy bonus is not needed to learn exploration, and is included purely for the sake of numerical stability. We optimize with default Adam parameters. Agents receive only a stream of reward 1. We postprocess trajectories using a discount factor of 0.99.

The environment network creates a stimulus vector by embedding the local region of the world visible to the agent. It embeds each tiles material index and concatenates with the number of occupying entities and flattens. It then concatenates the resultant vector with the (unlearned) features of the current entity. These include visible properties such as health, food, water, and position. In order to handle the variable number of visible entities, a linear layer is applied to each followed by 1D max pooling. Attention Bahdanau et al. (2014) may appear the more conventional approach, but recently susan zhang (2018) demonstrated that simpler and more efficient max pooling suffices. The pooled entity vector is then concatenated with the tile and self embeddings to produce a single vector and followed by a final linear layer as shown in Figure 4.

The value network uses the learned environment embedding to predict the discounted expected lifetime of the agent, equal to summing over a discounted trajectory of 1's. We found it possible to obtain good performance without discounting, but training was less stable.

The movement head and attack head select a movement direction (North, South, East, West, Pass) and attack style (Melee, Range, Mage), respectively. Our final set of experiments prescribes targeting to the agent with lowest health. This does not preclude agents from learning intelligent strafing and attack patterns. We have experimented with learned entity targeting using attention over candidate entity embeddings and cell targeting based off a flat list of relative cell locations. We find that these are increasingly difficult to learn due to required sample complexity: while we were able to learn semi-reasonable policies with learned entity targeting, we were not able to reproduce these results with cell targeting. We believe these to be feasible tasks at relatively low compute scale, but perhaps not in our setting with simplest possible architecture and training configuration.

## 6 EXPERIMENTS

We present an initial series of experiments using our platform to explore large population multiagent interaction. We find that agent competence scales with population size. In particular, increasing the maximum number of concurrent players (#ent) magnifies exploration and increasing the maximum number of populations with unshared weights (#pop) magnifies niche formation.

In all experiments, we fix an upper cap on maximum number of concurrent agents per experiment and per simulation of the environment. One agent is spawned per game tick, provided that doing so would exceed the spawn cap. Agents are sampled uniformly from a number of "populations". Ideally, we would fix #ent = #pop, as is the case in standard MMOs (humans are independent networks with unshared weights). However, this incurs sample complexity proportional to number of populations. We therefore share parameters across groups of up to 16 agents for efficiency.

### 6.1 SERVER MERGE: ELO FACILITATES GAME MECHANICS AND INTERPRETABILITY

We evaluate the effects on foraging performance of training with larger and more populations. At train time, we vary the spawn cap and number of populations in (16, 32, 64, 128) and (1, 2, 4, 8), respectively. We train on 100 prebuilt but randomly generated maps for a fixed number of trajectories per population 2. Each world is randomly assigned a maximum number of entities between 16 and the spawn cap. Mean lifetime is not comparable among experiments, as much of the task difficulty is induced by proximity to environment carrying capacity. We propose tournament style evaluation in order to directly compare policies learned in different experiment settings. Tournaments are formed by simply concatenating the player bases of each experiment–this is identical to an MMO server merge. Results are shown in Figure 5: we vary the maximum number of agents at test time and find that agents trained in larger settings consistently outperform agents trained in smaller settings.

Multiagent competition is a curriculum *magnifier*, not a curriculum. We do not expect that, given any rudimentary environment, multiagent competition will produce complex and interesting behavior–the task must contain something to magnify. We have demonstrated that, given a reasonable foraging

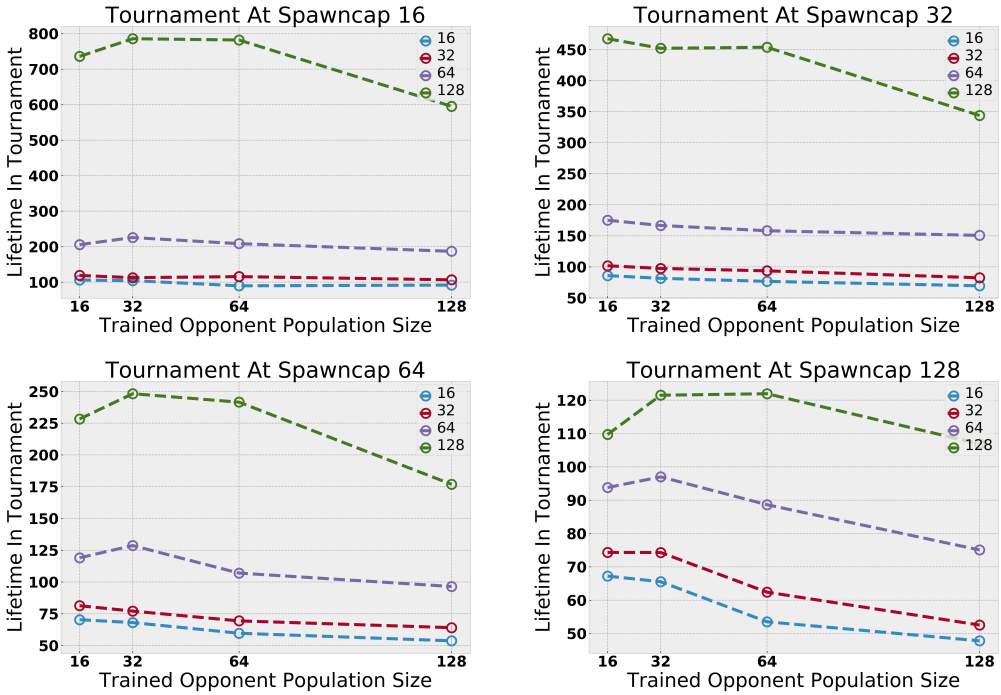


Figure 5: Agents are trained with maximum spawn caps of 16, 32, 64, 128. At test time, we merge the populations learned in pairs of experiments and evaluate lifetimes at a fixed spawn cap. We find that agents trained in larger populations outperform agents trained in small populations – including at low spawn cap.

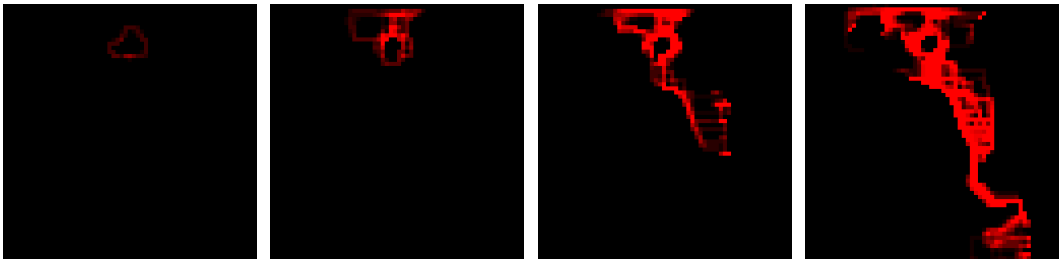

Figure 6: #Entities magnifies exploration. From left to right: we train with population size 1, 8, 32, 128. Agents spread out to avoid competition.

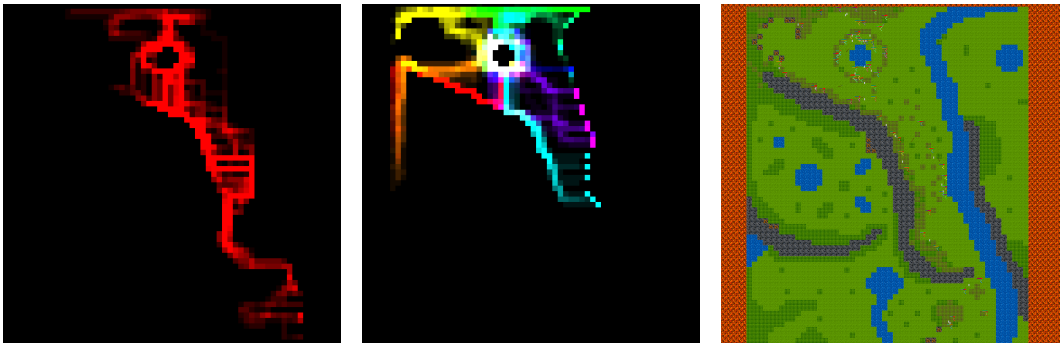

Figure 7: #Populations magnifies niche formation. From left to right, we train with 1, 8 populations. Populations spread out to avoid additional modes of competition.

task, multiagent competition leads to foraging behavior that scales with #ent and #pop. We now introduce the combat module as an additional learnable mode of variation on top of foraging and observe more interesting policies as a result, with agent actions strongly coupled to the states of other agents. As a sanity, we also confirm that all of the populations trained with combat handily outperform all of the foraging runs with random combat policies.

While tournaments demonstrate the curriculum magnifying effect of multiagent competition, it is not clear a priori exactly where the effect comes from. To better understand this result, we decouple this variability into two modes: maximum number of concurrent players (#ent) and maximum number of populations with unshared weights (#pop). This allows us to examine the effects of each factor independently. We find that environment randomization also encourages exploration; we therefore perform these experiments primarily as qualitative evaluations on a fixed map. This disentangles correlated factors and produces more immediately obvious results; however, results persist to some capacity in the environment randomized setting. We address this briefly in the Discussion section.

### 6.2 #Entity: Multiagent Competition Magnifies Exploration

In the natural world, competition between animals can incentivize them to spread out in order to avoid conflict. From Figure 6, we observe that overall exploration scales with number of concurrent agents, with no other variable factors (the map used is shown in Figure 7). Agents learn to explore only because the presence of other agents provides a natural and beneficial curriculum for doing so.

### 6.3 #Population: Multiagent Competition Magnifies Niche Formation

We find that, given a sufficiently large and resource rich environment, different populations of agents tend to separate to avoid competing with other populations. Both MMOs and the real world often reward masters of a single craft more than jacks of all trades. From Figure 7, specialization to particular regions of the map scales with number of populations. This suggests that the presence of other populations force agents to discover a single advantageous skill or trick. That is, increasing the number of populations results in diversification to separable regions of the map. As entities cannot out-compete other agents of their population with shared weights, they tend to seek areas of the map that contain enough resources to sustain their population. Regions that are difficult to get to or otherwise unoccupied are especially desirable; this is revealed by observing value maps over time.

## 7 Discussion

### 7.1 Environment Randomized Exploration

The trend of increasing exploration with increasing entity number is clear when training on a single map as seen in Figure 6, 7. It is perhaps more subtle once environment randomization is incorporated. From Figure 8, all population sizes explore adequately. We believe that this is because "exploration" as defined by map coverage is not as difficult a problem as developing robust policies. As demonstrated by the Tournament experiments, the exploration learned by smaller populations is more brittle and does not generalize to scenarios with more competitive pressure–even against a similar number of agents.

### 7.2 Agent-Agent Dependencies

We visualize agent-agent dependencies in Figure 9 using "ghosting" similar to that used to produce global value maps. We fix an agent at the center of a hypothetical map crop. For each position visible to that agent, we fake another agent and compute the value function estimate of the resultant pair. We find that agents learn policies dependent on those of other agents in both the foraging and combat environments.

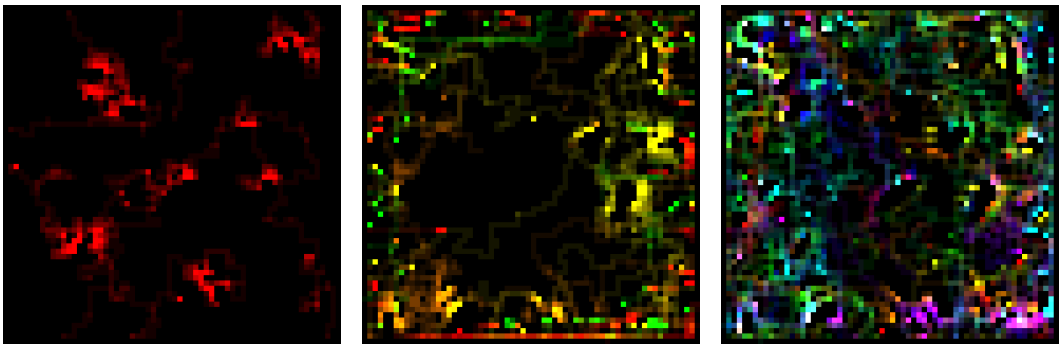

Figure 8: Exploration maps in the environment randomized settings. From left to right: population size 8, 32, 128. All populations explore well, but larger populations develop more robust policies, utilize the map more efficiently, and thereby do better in tournaments.

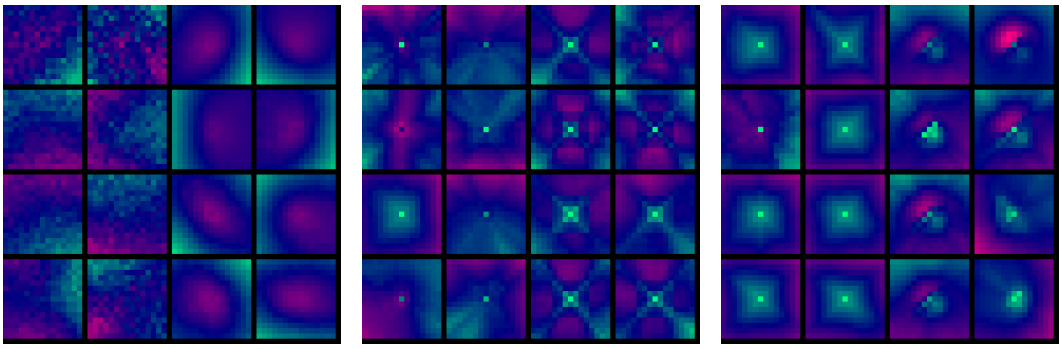

Figure 9: Agents learn to depend on other agents. Left: Columns 1, 2 denote random policies; columns 3, 4 are "bulls eye" avoidance maps learned after only a few minutes of training. Middle: Columns 1, 2 denote foraging; columns 3, 4 are learned combat policies with automatic targeting. As indicated by the vibrant square map centers, agents, agents fixate on the presence of other agents within combat range. Right: columns 1, 2 denote foraging; columns 3, 4 are combat policies with learned targeting. These begin to learn sensible dependencies, but are noisy.

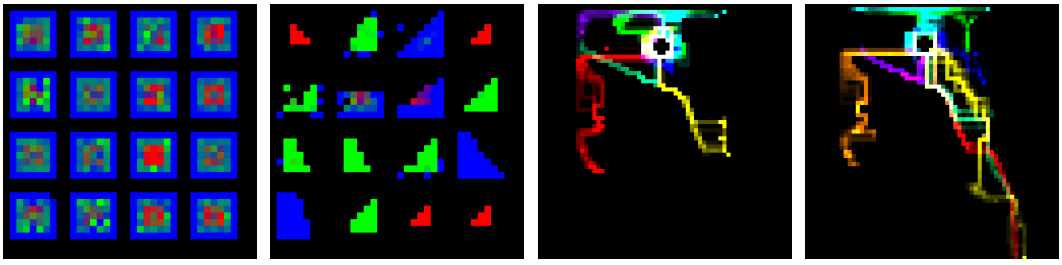

Figure 10: Attack maps and niche formation quirks. Left: combat maps from automatic and learned targeting. The left two columns in each figure are random, for comparison. Agents with automatic targeting learn to make effective use of melee combat (denoted by higher red density). Right: noisy niche formation maps learned in different combat settings with mixed incentives to engage in combat

### 7.3 Additional Insights

We briefly detail several miscellaneous investigations and subtle points of interest in Figure 10. First, we visualize learned attack patterns of agents. Each time an agent attacks, we splat the attack type to the screen, with red, green, and blue corresponding to melee, range, and mage, respectively. There are a few valid strategies as per the environment. Melee is intentionally overpowered: learning to utilize it at close range serves as a sanity check. This also incentivizes agents to keep their distance, as the first to strike wins, and priority is determined arbitarily. From Figure 10 and observation of the policies, we find that this behavior is learned when targeting is automated.

This is more problematic when targeting is learned jointly with attack style selection. We note that this formulation requires an attentional mechanism to handle variable number of visible targets; this is not yet numerically stable. We believe fixing this is likely to resolve the targeting issues. However, policies learned using automatic targeting are still compelling: agents learn to strafe at the edge of their attack radius, attacking opportunistically.

Second, a note on tournaments. We equate number of trajectories trained upon as a fairest possible metric of training progress. However, one possible source of bias is number of gradient steps (more steps on smaller batch size vs fewer steps on larger batch size), which is not currently comparable due to implementation specifics. We are actively correcting this. Note that we cannot train for equal flop count, as sample complexity is proportional to number of populations, which would result in a biased comparison. However, even in this scenario, we find the trend persists when evaluating at maximum spawn cap 128. As training progresses, the trend pushes to lower spawn caps as well, but with more mixed results. Finally, a quick note on niche formation. Obtaining clean visuals is dependent on having an environment where interaction with other agents is unfavorable. When this is not the case, niche formation may still occur in another space (e.g., diverse attack policies). This is the importance of niche formation–we expect it to stack well with population-based training (Jaderberg et al., 2017) and other such methods that require sample diversity.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

Giuseppe Cuccu, Julian Togelius, and Philippe Cudre-Mauroux. Playing atari with six neurons. *arXiv preprint arXiv:1806.01363*, 2018.

Sevan G Ficici and Jordan B Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the sixth international conference on Artificial life*, pp. 238–247. MIT Press, 1998.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Julien Perolat, David Silver, Thore Graepel, et al. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4190–4203, 2017.

Christopher G Langton. *Artificial life: An overview*. Mit Press, 1997.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985. ISSN 0097-8930. doi: 10.1145/325165.325247. URL http://doi.acm.org/10.1145/325165.325247.

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.

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

Kenneth O Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge youve never heard of. https://www.oreilly.com/ideas/open-endedness-the-last-grand-challenge-youve-never-heard-of, 2017. Accessed: 2017-09-26.

jakub pachocki henrique pond brooke chan filip wolski szymon sidor rafa jzefowicz przemysaw dbiak david farhi greg brockman jonathan raiman jie tang christy dennison paul christiano shariq hashme larissa schiavo ilya sutskever eric sigler jonas schneider john schulman christopher hesse jack clark quirin fischer diane yoon christopher berner scott gray alec radford david luan susan zhang, michael petrov. Openai five, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

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, pp. 263–263. ADDISON-WESLEY PUBLISHING CO, 1994.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*, 2018a.

Yaodong Yang, Lantao Yu, Yiwei Bai, Ying Wen, Weinan Zhang, and Jun Wang. A study of ai population dynamics with million-agent reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2133–2135. International Foundation for Autonomous Agents and Multiagent Systems, 2018b.

Lianmin Zheng, Jiacheng Yang, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. *arXiv preprint arXiv:1712.00600*, 2017.