The Neural-MMO Benchmark: Learning in Massively Multiagent Open Worlds

Joseph Suarez[*1] **Sharada Mohanty**[*2] **Dipam Chakraborty**[2] **Phillip Isola**[1]

[1]MIT [2]AIcrowd

jsuarez@mit.edu
mohanty@aicrowd.com

November 7, 2023

Abstract

Neural-MMO[7] is an open-source environment for agent-based intelligence research featuring large maps with large populations, long time horizons, and open-ended multi-task objectives. We propose a benchmark on this platform wherein participants train and submit agents to accomplish loosely specified goals – both as individuals and as part of a team. The submitted agents are evaluated against thousands of other such user submitted agents. Participants get started with publicly available code base for Neural-MMO, scripted and learned baseline models, and training/evaluation/visualization packages. Our objective is to foster the design and implementation of algorithms and methods for adapting modern agent-based learning methods (particularly reinforcement learning) to a more general setting not limited to few agents, narrowly defined tasks, or short time horizons. Neural-MMO provides a convenient setting for exploring these ideas without the computational inefficiency typically associated with larger environments.

Keywords

reinforcement learning, multiagent, cooperation, open-ended, MMO

Competition type

The Neural-MMO Benchmark will be a regular competition

[*]Lead organizers. Author ordering for the other authors is alphabetical.

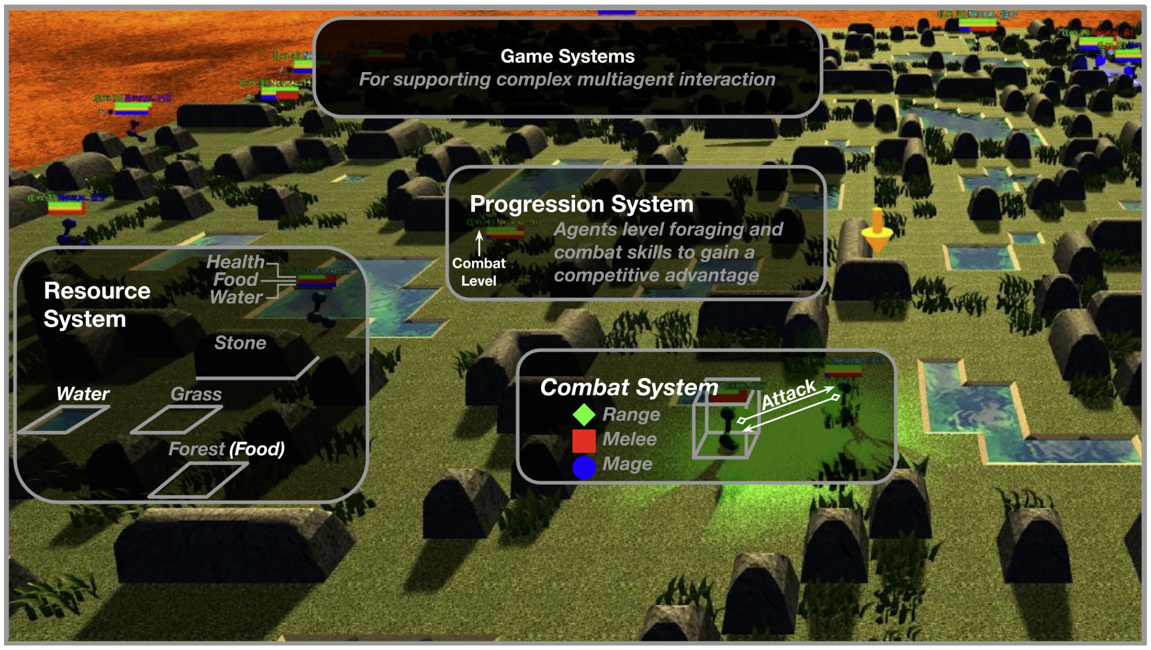


Figure 1: `The Neural-MMO Environment`

1 Competition description

1.1 Background and impact

The real world is a massively multiagent environment, and the ability to learn and reason within it is a hallmark of human intelligence. It seems inconceivable that artificial agents without this capability could operate intelligently within the real world outside of narrowly predefined tasks. Training directly within the real world has proven unwieldy because it often requires specialised hardware, is difficult to reproduce, and is generally expensive. The vast majority of tasks considered in modern reinforcement learning research are simulated environments. However, most of these are limited to a single or a few agents, short time horizons, and narrowly defined tasks.

In some sense, the lack of progress on more general environments can be attributed to a lack availability of such environments. `Neural-MMO` provides such a platform, and the purpose of the proposed benchmark is to spur research upon it. This would produce new methods for training many-agent policies, learning over long horizons, and navigating multimodal and loosely specified tasks. The ultimate goal is, of course, to integrate associated findings into real-world agents – `Neural-MMO` is simply a computationally efficient setting for developing such methods.

Initial baselines on the `Neural-MMO` environment have revealed that simple single-agent reinforcement learning methods work surprisingly well on this new domain. This makes the competition accessible to the entire reinforcement learning community – specialization to multi-agent learning is not required. That said, the later rounds of the competition will explore team-based play within the `Neural-MMO` environment. We expect specialised cooperation-based algorithms to be useful here. However, many multiagent reinforcement learning algorithms make strong assumptions about the task structure that may not prove true in the more general `Neural-MMO` setting – it is possible that simpler methods may prevail. Understanding the scalability of various modern reinforcement learning methods to increasingly general environments is the other explicit goal of the competition.

1.2 Novelty

Projects such as OpenAI Five and AlphaStar have succeeded in training highly competent few-agent systems. Other works inspired by older ALife research have developed large-scale systems where group behavior emerges from the simple actions of each particle agent [8, 9]. `Neural-MMO` has been designed to support both objectives simultaneously: large populations and complex per-agent decision making.

Previous competitions in agent-based intelligence include ProcGen [1], MineRL [3, 4], and PommerMan [5]. The former two considered only single-agent learning, whereas PommerMan support four independent agents or two teams of two agents each. In contrast, `Neural-MMO` supports hundreds to 1024 concurrent agents divisible into any number of teams. This creates an opportunity to study many-agent and many-team interactions not present in smaller environments.

For completeness, the NeurlPS 2020 Flatland challenge [2] also supports large numbers of agents in a reinforcement learning setting. However, the task considered is a more narrowly defined vehicle scheduling problem applicable to operations research. In contrast, `Neural-MMO` is more concerned with large-scale emergent behaviors and includes mixed competitive and cooperative incentives.

1.3 Environment

`Neural-MMO` is a game environment capable of hosting large number of agents in a interactive map. The maps are procedurally generated, so every map is different and allow for different themes and configurations. For example spawn location, resource density and item locations can be configured. The map is surrounded by a lake of lava, which is lethal to all agents. Every map features terrain features like grass, forest, water, stone etc. Agent may only pass through grass and forests but get blocked by water and stone.

The environment state is represented by a grid of tiles. Each tile has a particular assigned material with various properties, but it also maintains a set of references to all occupying entities. When agents observe their local environment, they are handed a crop of all visible game tiles, including all visible properties of the tile material and all visible

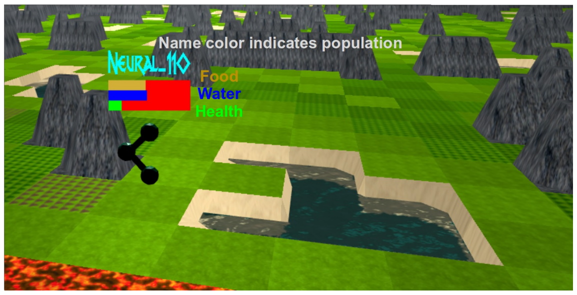


Figure 2: Example Agent

properties of occupying agents. All parameters in the following subsystems are configurable; we provide only sane defaults obtained via multiple iterations of balancing.

1.3.1 Tiles

We adopt a tile based game state, which is common among MMOs. This design choice is computationally efficient for neural agents and can be made natural for human players via animation smoothing. When there is no need to render the game client, as in during training or test time statistical tests, the environment can be run with no limit on server tick rate. Game tiles are as follows:

- **Grass:** Passable tile with no special properties
- **Forest:** Passable tile containing food. Upon moving into a food tile, the agent gains 5 food and the tile decays into a scrub.
- **Scrub:** Passable tile that has a 2.5 percent probability to regenerate into a forest tile on each subsequent tick
- **Stone:** Impassible tile with no special properties
- **Water:** Passable tile containing water. Upon moving adjacent to a water tile, the agent gains 5 water.
- **Lava:** Passable tile that kills the agent upon contact

1.3.2 Agents

1.3.2.1 Input: On each game tick, agents (Figure 2) observe a 15x15 square crop of surrounding game tiles and all occupying agents. We extract the following observable properties:

Per-tile properties:

- **Material**: an index corresponding to the tile type

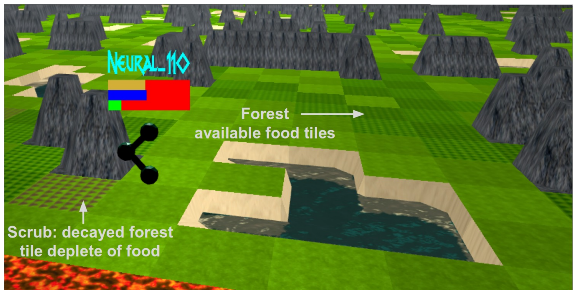


Figure 3: Example foraging behavior

- **nEnts**: The number of occupying entities. This is technically learnable from the list of agents, but this may not be true for all architectures. We include it for convenience here, but may deprecate it in the future.

Per-agent properties:

- **Lifetime**: Number of game ticks alive thus far
- **Health**: Agents die at 0 health (hp)
- **Food**: Agents begin taking damage at 0 food or water
- **Water**: Agents begin taking damage at 0 food or water
- **Position**: Row and column of the agent
- **Position Deltas**: Offsets from the agent to the observer
- **Damage**: Most recent amount of damage taken
- **Same Color**: Whether the agent is the same color (and thereby is in the same population) as the observer
- **Freeze**: Whether the agent is frozen in place as a result of having been hit by a mage attack

1.3.2.2 Output:

Agents submit one movement and one attack action request per server tick. The server ignores any actions that are not possible or permissible to fulfil, such as attacking an agent that is already dead or attempting to move into stone. *Pass* corresponds to no movement.

Movement:	**North**	**South**	**East**	**West**	**Pass**
Attack:	**Melee**	**Range**	**Mage**		

1.3.3 Foraging

Foraging (Figure 3) implements gathering based survival:

- **Food:** Agents begin with 32 food, decremented by 1 per tick. Agents may regain food by occupying forest tiles or by making use of the combat system.

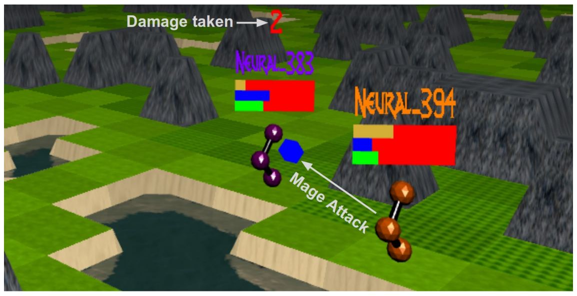


Figure 4: Example combat behavior

- **Water:** Agents begin with 32 water, decremented by 1 per tick. Agents may regain water by occupying tiles adjacent to water or making use of the combat system.
- **Health:** Agents begin with 10 health. If the agent hits 0 food, they lose 1 health per tick. If the agent hits 0 water, they lose 1 health per tick. These effects stack.

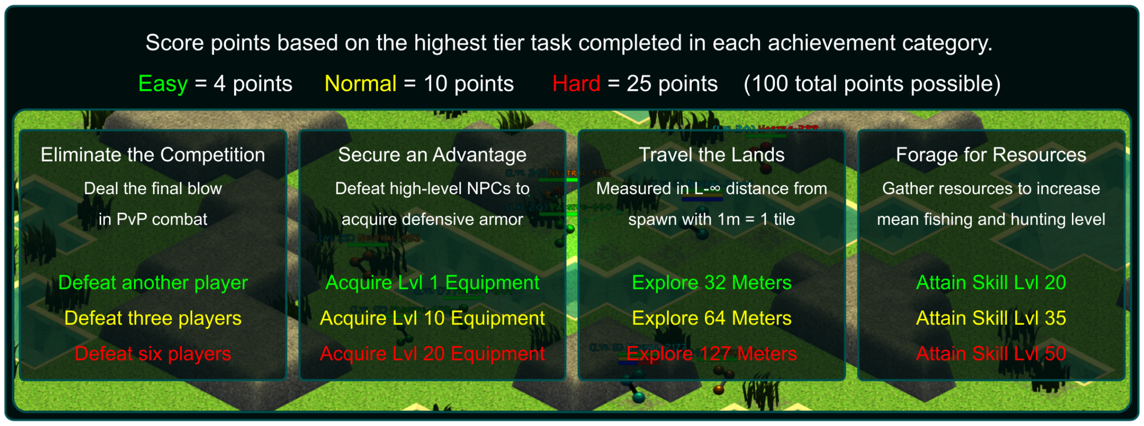
The limited availability of forest (food) tiles produces a carrying capacity. This incurs an arms race of exploration strategies: survival is trivial with a single agent, but it requires intelligent exploration in the presence of competing agents attempting to do the same.

1.3.4 Combat

Combat (Figure 4) enables direct agent-agent confrontation by implementing three different attack "styles":

- **Melee:** Inflicts 10 damage at 1 range
- **Ranged:** Inflicts 2 damage at 1-2 range
- **Mage:** Inflicts 1 damage at 1-3 range and freezes the target in place, preventing movement for two ticks

Each point of damage inflicted steals one point of food and water from the target and returns it to the attacker. This serves as an incentive to engage in combat. It is still fully possible for agents to develop primarily foraging based strategies, but they must at least be able to defend themselves. The combat styles defined impose clear but difficult to optimize trade offs. Melee combat fells the target in one attack, but only if they are able to make their attack before the opponent retaliates in kind. Ranged combat produces less risky but more prolonged conflicts. Mage combat does little damage but immobilizes the target, which allows the attacker to retreat in favor of a foraging based strategy. More aggressive agents can use mage combat to immobilize their target before closing in for the kill. In all cases, the best strategy is not obvious, again imposing an arms race.


Score points based on the highest tier task completed in each achievement category.

Easy = 4 points Normal = 10 points Hard = 25 points (100 total points possible)

Eliminate the Competition	Secure an Advantage	Travel the Lands	Forage for Resources
Deal the final blow in PvP combat	Defeat high-level NPCs to acquire defensive armor	Measured in L-∞ distance from spawn with 1m = 1 tile	Gather resources to increase mean fishing and hunting level
Defeat another player	Acquire Lvl 1 Equipment	Explore 32 Meters	Attain Skill Lvl 20
Defeat three players	Acquire Lvl 10 Equipment	Explore 64 Meters	Attain Skill Lvl 35
Defeat six players	Acquire Lvl 20 Equipment	Explore 127 Meters	Attain Skill Lvl 50

Figure 5: Achievement system with point thresholds for the first round.

1.3.5 Achievement System

`Neural-MMO` features an achievement system inspired by modern games. Agents may complete tasks of different difficulty (easy, normal, and hard) in a number of different categories, each awarding a different number of points. Figure 5 illustrates the achievement system with point thresholds for the first round of the proposed competition.

By design, it is possible to complete easy tasks without much deviation in overall strategy. Medium tasks require significant planning and time investment. Hard tasks require commitment to a premeditated strategy.

Relative difficulty across tasks was calibrated using the scripted and pre-trained baseline models. We evaluated these baselines across several trials and adjusted task thresholds accordingly. The baselines solve easy tasks roughly 50% of the time, normal tasks 10% of the time, and hard tasks less than 1% of the time. We believe this to be a reasonable initial calibration, and we will refine these thresholds throughout the competition based on participant feedback.

Points are awarded per-player in round 1 and per-team in rounds 2 and 3. Concretely, all rounds have a maximum score of 100 points, but in rounds 2 and 3, a task is considered completed if it is achieved by *any* agent on the team. This incentivizes specialization and division of labor – another important property of real-world intelligence that has been relatively unexplored in reinforcement learning.

1.4 Competition Design

1.4.1 Rounds

The proposed benchmark is divided into three Rounds defined by progressively increasing configurations of the map size, population size, and team size:

Round	Map Size	Population Size	Team Size
#1	128	128	1
#2	128	128	8
#3	1024	1024	32

Table 1: Configurations across Rounds

1.4.1.1 Round 1

Round 1 is designed as an accessible entry point into the competition. Round 1 features Game Servers of **128x128 maps** with a **population size of 128 agents** and a **team size of 1**. Hence, participants submit a single policy to compete against 127 other agents on a 128x128 map. Simple methods are expected to work reasonably well and do not require extensive prior knowledge of multi-agent algorithms or even reinforcement learning in general. Our baselines were trained using a single GPU in less than a day, and reasonable performance is attainable within a few hours of training: participants with access to a single personal GPU or even Google Colab will be able to compete. At the same time, the task ceiling is sufficiently high to distinguish between submitted policies: there are few, if any, environments that require robustness to so many unseen opponents. This introduces an opportunity for significant advances in robust learning algorithms including population-based training, historical self-play, and exploiter networks.

1.4.1.2 Round 2

Round 2 features Game Servers of **128x128 maps** with a **population size of 128 agents** and a **team size of 8**, introducing team-based play. Participants may use their Round 1 submission as a starting point but will need to introduce cooperation into their approach. Credit assignment becomes rapidly (perhaps exponentially) more difficult as the number of cooperative agents increases. It is unclear whether the reward sharing schemes used in 2-5 agent cooperative tasks will scale to teams of 32. This task incentives participants to explore the issue in order to take advantage of greater coordination.

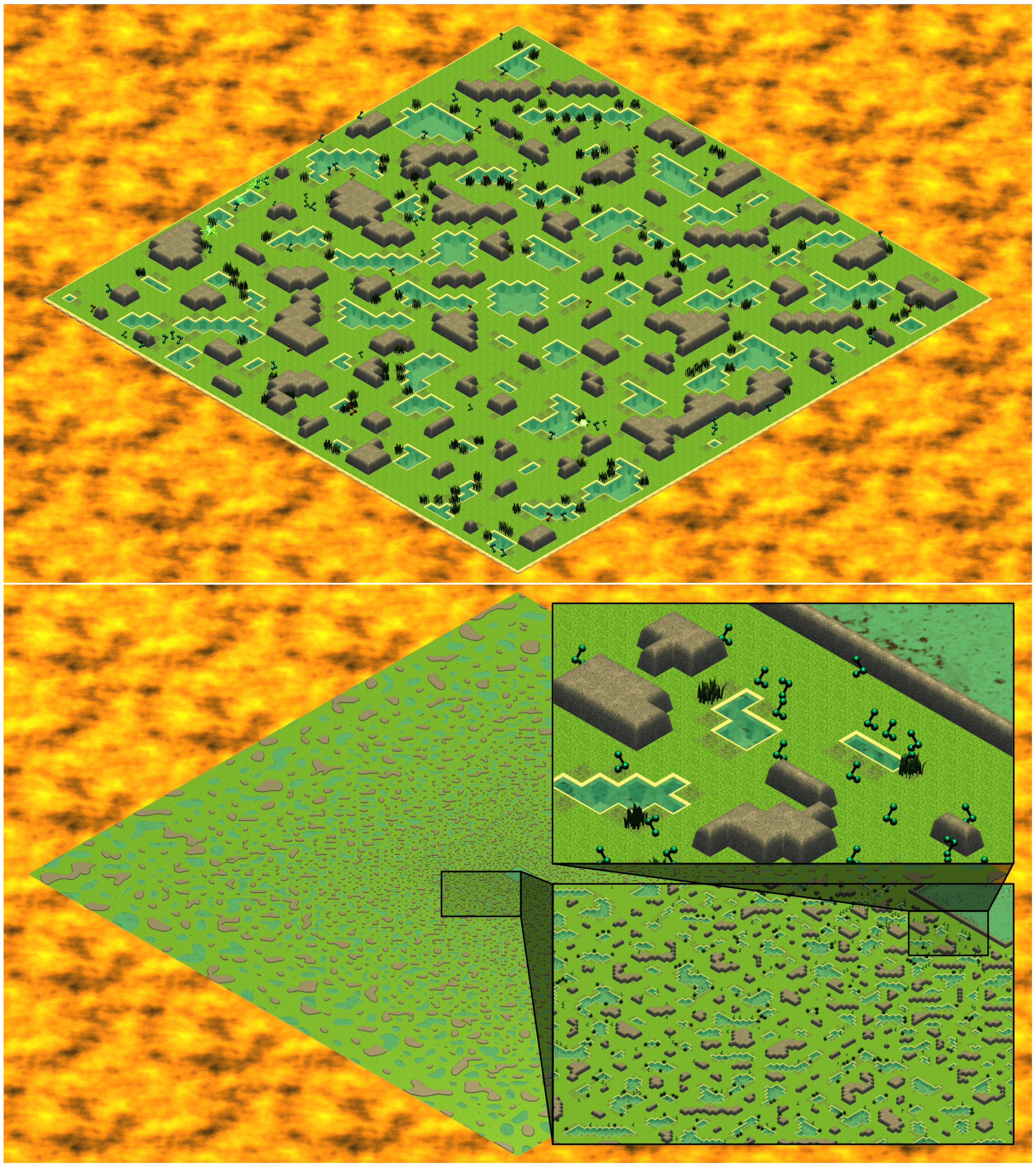


Figure 6: Sample map from Rounds 1 and 2 (top) and Round 3 (bottom). Participants are given access to the map generation code and may sample any number of maps or alter the training distribution as they choose.

1.4.1.3 Round 3

Round 3 is a large-scale version of the previous two tasks: participants train policies which control teams of 32 agents within 1024x1024 maps with a total population of 1024 agents. Larger maps give agents more room to make use of the progression system in `Neural-MMO` to train skills and specializations over multiple hours of gameplay. Doing so will require participants to develop robust and cooperative policies capable of extended reasoning. These properties of human intelligence are currently inaccessible to artificial agents. While even our largest environment is quite simple compared to the real world, the level of abstraction of `Neural-MMO` allows us to explore the same qualitative properties in a computationally efficient setting.

1.5 Metrics

Across all the rounds, participants will be able to submit their trained policies to our automated evaluation servers. On receipt of the submitted policies, our evaluators will evaluate the submitted policies immediately on several held-out maps against a fixed pool of scripted and/or pretrained opponents. Some of these scripted and pretrained oppponents will be made available to the participants in order to facilitate local validation for the participants. Points will be awarded based on the achievement system described in Section 1.3.5, and all scores will be recorded on the public leaderboard of the round. This evaluation provides immediate feedback but will not be used to determine competition winners.

1.5.1 Tournaments

Apart from the public leaderboard of the round, we will also maintain a Tournament Leaderboard for each of the rounds. Every week we will run a Tournament using the best submission from the top-N teams on the public leaderboard. The Tournament will be composed of multiple ”Games” using `Neural-MMO` where user submitted policies will be the ”players”. The Game specific rankings will be determined by the in-game achievement points (as described in Section 1.3.5), which will be used to update the TrueSkill [6] ratings for the individual players (user submitted policies). The updated TrueSkill ratings are then used to match groups of ”players” optimally for the subsequent games. These games are iteratively conducted until the TrueSkill Ratings converge. The final TrueSkill ratings will be then used to update the weekly Tournament Leaderboard. The winners of the competition will be determined by the final standings on this Tournament Leaderboard.

Honorary mentions may be awarded for approaches that stand out in other ways, such as using a particularly small model, a simple training scheme, or lack of reward shaping.

1.6 Baselines, code, and material provided

`Neural-MMO` includes two scripted baselines and a pretrained recurrent network with associated training code. Evaluation results for all three models are available on the project

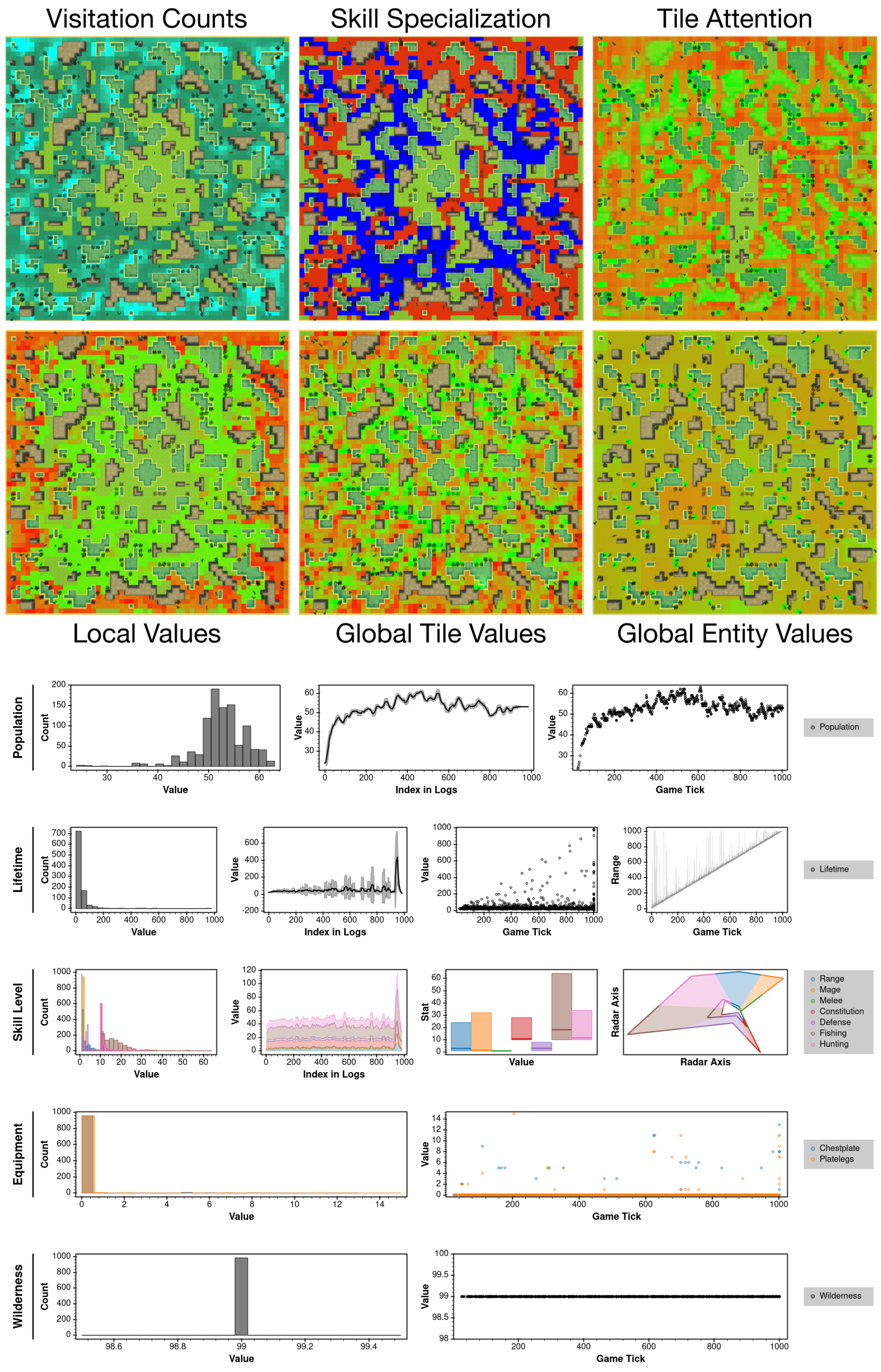


Figure 7: Overlays for visualizing learned agent behaviors (top) and interactive dashboard for comparing policies across achievement point distribution and various correlated metrics of potential use for model development (bottom).

homepage above. The round 1 task is very similar but not identical to the setting for which these baselines were designed/trained. We have already begun performing the necessary adaptations.

- **Baselines:**
  https://jsuarez5341.github.io/neural-mmo/build/html/rst/baselines.html
- **Source Code:**
  https://github.com/jsuarez5341/neural-mmo
- **API:**
  https://jsuarez5341.github.io/neural-mmo/build/html/rst/forge.trinity.env.html

Participants will have access to the full source code of `Neural-MMO`: environment, map generator, evaluations, renderer, and visualizations. The baselines above include two strong scripted models and one pretrained recurrent model with all associated source and training code. We have also provided API documentation and guides for the various components of the environment.

We will not be providing hardware or cloud credits for training, but we expect participants with access to a single GPU to be able to take part comfortably. Note that even our baseline was trained using a single commodity GPU for less than a day. We believe this efficiency is currently unique to the `Neural-MMO` platform: there are no other comparably complex many-agent environment, and many of the best few-agent environments require industry-scale hardware (e.g. DoTA, Starcraft, Go, DeepMind Capture the Flag, OpenAI Hide and Seek).

1.7 Tutorial and documentation

A list of publications and whitepapers is available at https://jsuarez5341.github.io/neural-mmo/build/html/rst/updates.html. Full tutorials and documentation are available on the same site at https://jsuarez5341.github.io/neural-mmo/build/html/rst/userguide.html. These documents currently reflect the latest official environment release. We have already started performing necessary adaptations for the competition version – the differences are minimal.

2 Organizational aspects

2.1 Protocol

2.1.1 Platform

The evaluation of the submissions will be managed by AIcrowd. The platform offers state-of-the art technology and management specifically for running competitions like the one proposed here. The technology underlying AIcrowd has been tried and tested in over 50 competitions, including many at NeurIPS (NeurIPS 2020 Procgen Challenge, NeurIPS 2020 Flatland Challenge, The Learning to Run Challenge in 2017, 2018, 2019, the AI for Prosthetics Challenge in 2018, and the Adversarial Vision Challenge in 2018, Disentanglement Challenge in 2019, MineRL Competition in 2019/2020, REAL Robots Challenge in 2019). The AIcrowd platform offers to the full range of services for a competition, from user management to discussion forums and leaderboards. The evaluation infrastructure can be flexibly adapted to any execution requirements, and allows for full traceability and reproducibility of the submissions.

2.1.2 Submission Protocol

Throughout the competition, participants can work on their code bases as private git repositories on `https://gitlab.aicrowd.com`. The requirements of the AIcrowd evaluators require participants to package their intended software runtime in their repositories, to ensure that the evaluators can automatically build relevant Docker images from their repositories, and orchestrate them as needed by the evaluators of the particular round. This approach also ensures that all the user submitted code that is successfully evaluated in context of this competition is both versioned across time, and also completely reproducible.

2.1.2.1 Software Runtime Packaging. Packaging and specification of the software runtime is among the most time consuming (and frustrating) tasks for many participants. And hence, to make this step easier, we will be supporting numerous approaches of easily packaging the software runtime by the help of aicrowd-repo2docker. A tool which lets participants specify their runtime using Anaconda environment exports, requirements.txt , or even with the fall back of a traditional Dockerfile. This significantly decreases the barrier to entry for the less technically inclined participants by transforming an irritating debug cycle to a deterministic one-liner which does all the work behind the scenes.

2.1.2.2 Submission Mechanism. Participants can collectively collaborate together on their private git repository throughout the competition, and whenever they wish to make a submission, they have to create and push a *git tag*, which triggers the evaluation pipeline: image building, orchestration of the containers with the required evaluation context. On the successful evaluation of the submission, the scores and any relevant artifacts(generated media, etc) are added automatically to the leaderboard.

2.1.2.3 Orchestration of the Submissions. The ability to reliably orchestrate user submissions over large periods of time will be a key determining feature of the success of the proposed competition. We will be using the evaluators of `aicrowd.com` which use custom Kubernetes clusters to orchestrate the submissions against pre-agreed resource usage constraints. The same setup has been successfully used previously in numerous other complicated machine learning competitions in the past. The evaluation setup allows for evaluations of arbitrarily long time-periods, and also can privately provide feedback about the current state of the evaluation to the respective participants.

2.2 Rules

1. Do not make any effort to intentionally overwhelm our evaluation servers.
2. You may mark up to three submissions for inclusion in the tournament at any given time.
3. You may adjust your algorithm, training scheme, and reward in response to aggregate observations of other agents, but do not hard-code exclusive targeting of a particular submission or subset of submissions (harassment) or similar nonaggression (teaming).
4. Reproducible training code is required for prize eligibility. We will contact you individually if you qualify for a prize.
5. Attempting to circumvent any of the above will result in disqualification.

We will be monitoring the submission portal for suspicious activity including duplicate or similar models or weights from different users. Contact us if you believe your submission is being individually targeted by another competitor or if you see signs of teaming. We will investigate all accusations thoroughly and disqualify any participants for whom we find conclusive evidence of cheating.

2.3 Schedule and readiness

- Apr 13th: Proposal Accepted
- May 1st: Beta release of all benchmark resources
- June 1st - August 1st: Round 1
- August 1st - September 15th: Round 2
- September 15th - October 19th: Round 3
- October 20th - October 25th : Post Competition Analysis
- October 25th : Final Results Announced
- October 16th - November 10th : Post Competition Wrap-Up

Indicate what, at the time of writing this proposal, is already ready.

As mentioned above, the competition version used in `The Neural-MMO Benchmark` is practically identical to the current v1.5 public release. The only significant changes are:

- The inclusion of an achievement-based reward function
- Agents are all spawned at the start of the simulation rather than gradually over time
- Minor tweaks to terrain generation code for better task consistency across rounds
- In Round 3, agents are spawned at the edges rather than at the center of the map
- Retraining of baselines with the above modifications
- Documentation update with the above modifications

Thus far, we have implemented the achievement-based reward system and retrained the round 1 baseline accordingly. We have also updated the terrain generation code and made appropriate tweaks to spawning code for rounds 1 and 2. We still need to finalize the round 3 spawning code, add a cooperative baseline trained with shared rewards, and update documentation accordingly. There are perhaps one or two person-days of work left for each of these tasks with the option of spending longer depending on how strong of a baseline we wish to present for round 2. We also need to write a thin wrapper around `Neural-MMO` for loading several submitted policies and evaluating TrueSkill. This will take two or three person-days to implement, leaving well over a week for integration with existing evaluation infrastructure and fine-tuning.

2.4 Competition promotion

We will advertise `The Neural-MMO Benchmark` using:

- AIcrowd's website and platform
- The contact points of several labs which have previously invited `Neural-MMO` related talks
- The MIT Embodied Intelligence slack and mailing list
- The existing `Neural-MMO` Discord server with 250 members, which will also be used for support and discussion during the competition
- The popular reinforcement learning Discord server with several hundred active members
- The organizers' professional Twitter accounts

3 Resources

3.1 Organizing team

Sharada Mohanty is the CEO and Co-founder of AIcrowd, a community of AI researchers built around a platform encouraging open and reproducible artificial intelligence research. He was the co-organizer of many large-scale machine learning competitions, such as NeurIPS 2017: Learning to Run Challenge, NeurIPS 2018: AI for Prosthetics Challenge, NeurIPS 2018: Adversarial Vision Challenge, NeurIPS 2019: MineRL Competition, NeurIPS 2019: Disentanglement Challenge, NeurIPS 2019: REAL Robots Challenge, NeurIPS 2020: Flatland Competition, NeurIPS 2020: Procgen Competition. He is extremely passionate about benchmarks and building communities. During his Ph.D. at EPFL, he worked on numerous problems at the intersection of AI and health, with a strong interest in reinforcement learning. In his previous roles, he has worked at the Theoretical Physics department at CERN on crowdsourcing compute for Monte-Carlo simulations using Pythia; he has had a brief stint at UNOSAT building GeoTag-X - a platform for crowdsourcing analysis of media coming out of disasters to assist in disaster relief efforts. In his current role, he focuses on building better engineering tools for AI researchers and making research in AI accessible to a larger community of engineers.
Joseph Suarez is a PhD student in Phillip Isola's lab at MIT, the founder of `Neural-MMO`, and the lead developer of the competition-specific version of the environment. He has presented `Neural-MMO` at the ICML 2020 Learning in Artificial Open Worlds workshop, as an extended abstract at AAMAS 2020, and as an invited speaker at several academic laboratories and research groups. Previously, he completed his BS in computer science with an emphasis on artificial intelligence at Stanford University and published a language model at NeurlPS (then NIPS) 2017.
Dipam Chakraborty is an ML Engineer at AIcrowd Research, and has previously worked as an ML Engineer at Intel. Dipam was part of one of the winning teams of the NeurIPS 2020 Procgen competition. Dipam holds Bachelors and Masters degrees in Electronics and Communication Engineering from NIT Rourkela.
Phillip Isola received a Bachelors of Science in Computer Science from Yale, a Ph.D in Brain & Cognitive Sciences at MIT working under the supervision of Ted Adelson, and a postdoctoral term in the EECS department at US Berkeley with Alyosha Efros. He spent a year working as a visiting research scientist at OpenAI and is now an assistant professor in EECS at MIT studying computer vision, machine learning, and AI. His laboratory focuses on studying why we represent the world the way we do and how we can replicate these abilities in machines.

3.2 Resources provided by organizers, including prizes

Authorship: We will release a manuscript summarizing the benchmark and key finding. The top three teams will be invited to contribute material detailing their approaches and be included as authors. Honorable mention recipients will be invited to contribute a shorter section summarizing their work with their names attached. We will help revise submitted figures and writing in collaboration with the winners.

AIcrowd will take a lead in the preparation of the starter-kits for smooth onboarding of the participants. AIcrowd will provide staff for the implementation and maintenance of the automated evaluation setup. AIcrowd will provide staff for supporting and assisting participants during their code submission processes. AIcrowd will provide staff for communication and community engagement activities around the competition. AIcrowd will take a lead in securing sponsorship to cover the costs of compute for the evaluation requirements of the competition. In the event that AIcrowd does not manage to secure compute sponsorship for the competition, AIcrowd will commit to covering the costs of the compute associated with the evaluation of the submissions. AIcrowd will take a lead in securing sponsorship for awarding prizes to the top teams.

3.3 Support requested

The results of the competition will be announced at NeurIPS 2021. We plan to have short presentations from the top teams and interesting solutions during the conference. We also plan to present a summary of the competition at the conference, including the participation statistics, results and our analysis of the solutions. We request NeurIPS to support by providing the video-conferencing platform for these presentations, or advertising about the platform where they will be presented.

References

[1] Karl Cobbe et al. "Leveraging Procedural Generation to Benchmark Reinforcement Learning". In: *CoRR* abs/1912.01588 (2019). arXiv: 1912.01588. URL: http://arxiv.org/abs/1912.01588.

[2] *Flatland: Challenges*. URL: https://www.aicrowd.com/challenges/neurips-2020-flatland-challenge/.

[3] William H. Guss et al. "MineRL: A Large-Scale Dataset of Minecraft Demonstrations". In: *CoRR* abs/1907.13440 (2019). arXiv: 1907.13440. URL: http://arxiv.org/abs/1907.13440.

[4] William H. Guss et al. "The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors". In: *CoRR* abs/1904.10079 (2019). arXiv: 1904.10079. URL: http://arxiv.org/abs/1904.10079.

[5] Cinjon Resnick et al. "Pommerman: A Multi-Agent Playground". In: *CoRR* abs/1809.07124 (2018). arXiv: 1809.07124. URL: http://arxiv.org/abs/1809.07124.

[6] B. Schölkopf, J. Platt, and T. Hofmann. “TrueSkill™: A Bayesian Skill Rating System”. In: *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. 2007, pp. 569–576.

[7] Joseph Suarez et al. “Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents”. In: *arXiv preprint arXiv:1903.00784* (2019).

[8] Yaodong Yang et al. “An Empirical Study of AI Population Dynamics with Million-agent Reinforcement Learning”. In: *CoRR* abs/1709.04511 (2017). arXiv: 1709.04511. URL: http://arxiv.org/abs/1709.04511.

[9] Lianmin Zheng et al. “MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence”. In: *CoRR* abs/1712.00600 (2017). arXiv: 1712.00600. URL: http://arxiv.org/abs/1712.00600.