The Second Neural MMO Challenge: Learning in Massively Multiagent Open Worlds

Joseph Suarez*, Sharada Mohanty§, Jiaxin Chen†, Hanmo Chen†⋆,
Clare Zhu£, Xiu Li⋆, Julian Togelius‡, Phillip Isola*
*MIT, †Parametrix.AI, §AICrowd,
⋆Tsinghua Shenzhen International Graduate School, Tsinghua University,
£Stanford University, ‡New York University

November 7, 2023

Abstract

Neural MMO [SDIM19] 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

1 Contact Information

- Joseph Suarez, MIT, jsuarez@mit.edu
- Jiaxin Chen, Parametrix.AI, jiaxinchen@chaocanshu.ai
- Hanmo Chen, Tsinghua Shenzhen International Graduate School, Tsinghua University chm20@mails.tsinghua.edu.cn

2 The Proposed Neural MMO Challenge description

2.1 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 multi-modal 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.

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

2.2.1 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

2.2.2 Agents

Input: On each game tick, agents 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
- **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

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**		

2.2.3 Foraging

Foraging 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.
- **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.

2.2.4 Combat

Combat (Figure 1) 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.

Figure 1: Example combat behavior

Figure 2: Achievement system with point thresholds for the first round.

2.2.5 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 2 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.

2.3 Competition Design

2.3.1 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

Table 1: Configurations across Rounds

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.

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.

2.4 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 2.2.5, 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.

2.4.1 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 2.2.5), which will be used to update the TrueSkill 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.

2.5 Baselines, code, and material provided

`Neural-MMO` includes two scripted baselines and a pre-trained recurrent network with associated training code. Evaluation results for all three models are available on the project 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://github.com/NeuralMMO/baselines
- **Source Code:**
  https://github.com/NeuralMMO/environment

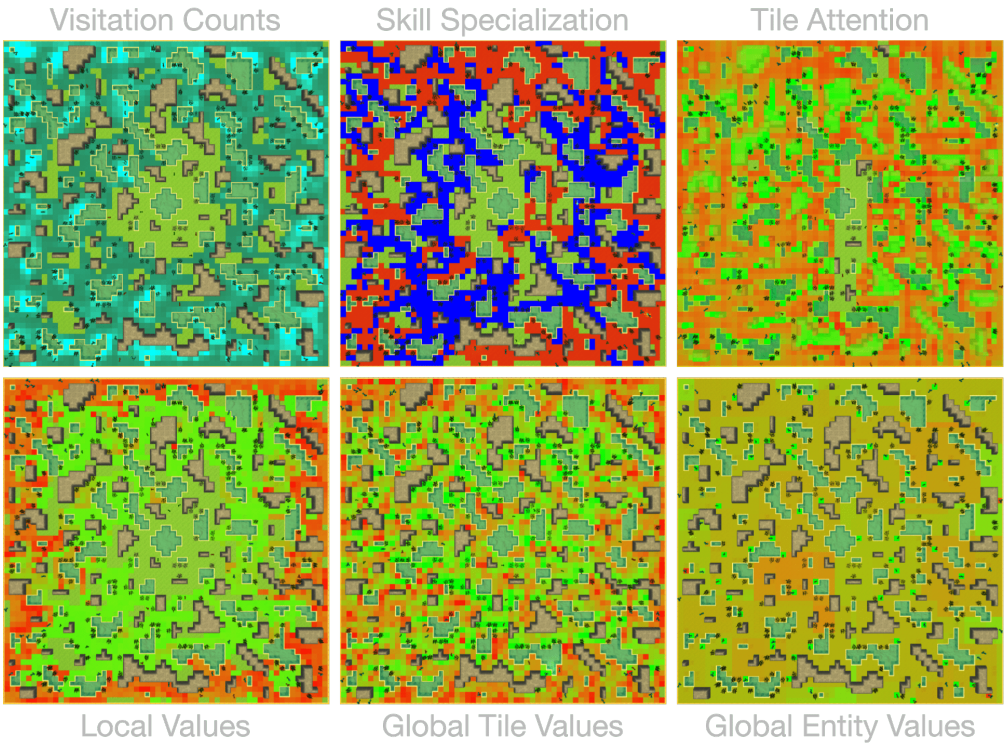

Figure 3: Overlays for visualizing learned agent behaviors.

- **API:**
  https://neuralmmo.github.io

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).

3 Organizational aspects

3.1 Protocol

3.1.1 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.

3.1.2 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.

Submission Mechanism. Participants can pack their evaluation codes and the model into an archive file through the AICrowd UI page. On the successful evaluation of the submission, the scores and any relevant artifacts(generated media, etc) are added automatically to the leader-board.

3.2 Rules

1. Do not make any effort to intentionally overwhelm our evaluation servers.
2. Participants may mark up to three submissions for inclusion in the tournament at any given time.
3. Participants may adjust their 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 the winners individually if they 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.

3.3 Challenge Structure

3.3.1 Competition Rounds

The Neural-MMO Challenge provides a unique opportunity for participants to explore robustness and teamwork in a massively multi-agent setting with opponents not seen during training. This Challenge consists of two rounds as described in Section 2.3.1. There are no qualifiers; participants can submit to any or all of the rounds independently.

3.3.2 Evaluation Stages

We will evaluate your agent in two stages.

Stage 1: Verses Scripted Bots We will evaluate your agent against scripted baselines of a variety of skill levels. Your objective is to earn more achievement points (see Evaluation Metrics) than your opponents. We will estimate your agent's relative skill or match-making rank (MMR) based on several evaluations on different maps. We generate these maps using the same algorithm and parameters as provided in the starter kit, but we will use another random seed to produce maps outside of the direct training data.

Stage 2: Verses Other Participants We will evaluate your agents against models submitted by other participants as well as our baselines. When there are only a few submissions at the start of the competition, we will take a uniform sample of agents for each tournament. Once we have enough submissions from the participants, we will run tournaments by sampling agents of similar estimated skill levels. Your objective is still to earn more achievement points than your opponents.

4 Participants

The number of teams is expected to be about 50 and the number of participants is expected to be about 100, with teams consisting of up to 4 participants. The estimation is based on the previous Neural MMO competition organized by AIcrowd[1] and the NetHack Competition [2]. We expect to the champion team to attend IJCAI-ECAI 2022 in person but only if the team member is willing.

5 Prizes

We plan to provide cash prizes for winners of both Round 1 and Round 2. The money will be sponsored by Parametrix.AI and the total cash pool is 20,000 US Dollars. The prizes are designed as follows.

- Round 1
  - No.1: $ 3000
  - No.2: $ 2000
  - No.3: $ 1000
  - Community Prize: $1000
- Round 2
  - No.1: $ 5000
  - No.2: $ 3000
  - No.3: $ 2000
  - No.4-No.5: $ 1000
  - Community Prize: $ 1000

Additionally, we plan to set up a Community Prize participants who substantially contribute to the Neural MMO community and help foster the challenge.

6 Source Code

The Neural MMO environment is open sourced[3] with MIT license. Also we will give a starter kit and a baseline for the starters. We encourage the participants to open-source their code after the competition but it is not forced.

7 Timeline in Milestones

The competition has 2 phases. Timeline of the competition 4:

- 2 March: Send out call for participation.
- 20-31 March: Test period, to see if the baselines work for participants.
- 10 March: Phase 1 starts.
- 24 March: Phase 2 starts.
- 28 May: Submission Deadline for Phase 1.
- 10 June: Submission Deadline for Phase 2.
- 25 June: Winner Announcement for Phase 1 & 2.

[1] https://www.aicrowd.com/challenges/the-neural-mmo-challenge
[2] https://www.aicrowd.com/challenges/neurips-2021-the-nethack-challenge
[3] https://github.com/NeuralMMO/environment

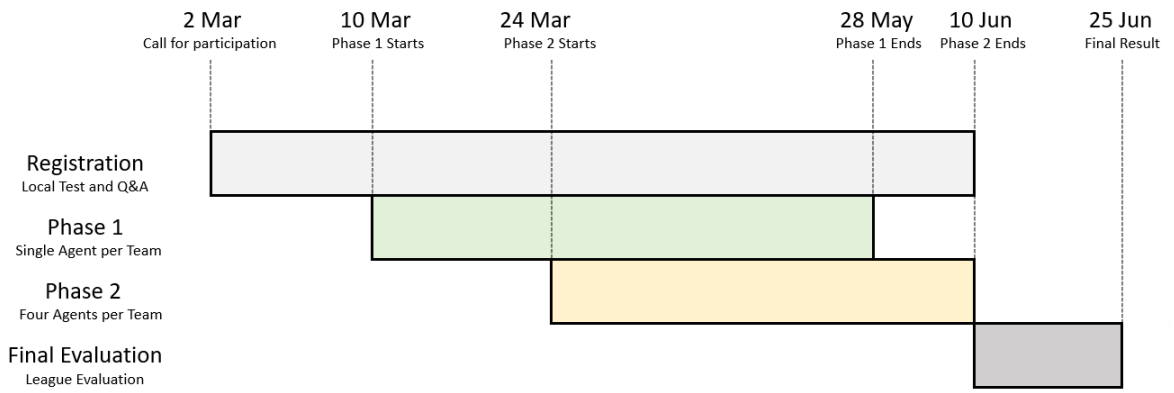

Figure 4: Timeline for the Neural MMO Challenge.

8 Post-Proceedings / Publications

We are interested in preparing one paper of our challenge including the most interesting agents / algorithms / techniques developed by participants.

8.1 Resources Needed during the Conference

- Floor space in square meters (minimum nice to have)? Nice to have 50 square meters.
- Number of beamers with screens required (minimum nice to have)? Nice to have 1.
- Number of monitors required (minimum nice to have)? Nice to have 2.
- Number of tables (minimum nice to have)? Nice to have 50.
- Number of chairs (minimum nice to have)? Nice to have 50.
- Number of electricity sockets and watts (minimum nice to have)? Nice to have 4.
- Wifi accessibility (must-have or nice-to-have)? Must-have
- Private space (must-have or nice-to-have)? In other words, can your competition or challenge be held in open space, e.g., the lobby? Nice to be held in open space.

9 Organizer Backgrounds and Roles

- Joseph Suarez: Creator of Neural MMO, organizer in the first Neural MMO Challenge and co-organizer in this challenge
- Sharada Mohanty: AICrowd founder, organizer in the first Neural MMO Challenge and co-organizer in this challenge
- Jiaxin Chen: Senior Research Scientist at Parametrix.AI, co-organizer of this challenge
- Hanmo Chen: Winner of first Neural MMO Challenge, in the AI master program of Tsinghua Shenzhen International Graduate School, research intern at Paramatrix.AI
- Xiaolong Zhu: Senior Director at Parametrix.AI, competition advertising and university relations
- Xiu Li, Professor at Tsinghua Shenzhen International Graduate School, advising on challenge organization
- Clare Zhu: Data scientist involved in early versions of Neural MMO, advising on accessibility to non-RL methods/backgrounds

- Julian Togelius: Associate Professor, NYU, expert in procedural generation with experience on Neural MMO
- Phillip Isola: Assistant Professor, MIT, Advisor of Neural MMO for 3+ years

References

[SDIM19] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019.