Understanding Behaviors Learnt by Multi-agent RL for Massively Multiplayer Online Games

Siyu Shen
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
ss019@ie.cuhk.edu.hk

Ke Zhang
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
zk019@ie.cuhk.edu.hk

Abstract

MMORPG (massively multiplayer online role-playing game) are a kind of game that enables a large and variable number of agents to compete and even fight for finite resources. To survive and win in these games, agents need to have complicated strategies under different cases of competition and cooperation. Training agent in these games can be hard because the number of agents could be large and the environment need to support them. The project Neural MMO[7] by Openai provides a platform for such games. Based on that project, we will train the agents and study what behavior pattern they have learned under different cases and at different training stages. We found that some behaviors, like avoiding fatal lava and going through traversable tiles, are easier to learn and can occur at early time; but looking for food and water, and making intelligent attacks are hard to learn by the agent under short training time.

1 Introduction

Reinforcement learning agents have learnt to do many tasks, but it still has a long way to go before it can think like humans and do complex tasks with competition and cooperation in a complicated environment. This complexity was reflected in a particular genre of games, MMORPGs, in which a large and variable number of agents compete and even fight for finite resources. It will be interesting to see how agents learn to make decisions, and what intelligent strategies they have found in such an environment. The project Neural MMO[7] by Openai provides a platform for such games by supporting a large number of agents to interact with environment and each other. This platform is quite large, and to train a full-fledged agent is almost impossible for us with only a normal desktop computer. Thus in this project, we will train the model for some time and compared the behaviors they have learned at different stages of learning.

2 Related Work

The recent advancement of reinforcement learning has enabled a single agent to act for its own interests in many cases. However, there are many cases in real life where many agents are involved and affected by each other's behaviors. These can be better modelled as multi-agent reinforcement learning problems.

For example, in [2], agents in two teams are playing hide-and-seek game in a complex environment with various objects. It was showed that the agents not only learned how to use tools, but also learned to cooperate between team members, and use countermeasures to cope with a new measure from the opponent team. In [5], sequential social dilemmas are introduced to model the real-world social dilemmas. Several agents are independently trained using deep-Q networks and the effects of cooperation and competition are showed. In [3], the authors used multi-agent reinforcement learning techniques to control traffic lights at several linked intersections. They used independent DQN for training and achieved better performance than baseline methods. [4] studied how a group of robot swarms with limited sensing capabilities can learn to accomplish sophisticated tasks. To simplify the training, they used a actor-critic approach where the critic has global state information. Paper [8] proposed a large-scale DRL training platform enabling millions of intelligent agent to act and learn. The agents perform a simple predator-pray behavior and their findings are align with the self-organization theory studied in population biology.

Looking at the emergent behaviors learned by agents can be interesting and enlightening. Most papers will show that, and the original paper of Neural-MMO[6, 7] has also observed the behavior of agents. They showed the value function of learned map, which forms the pattern that central area is higher than borders (because far away from lava). For attacks, they showed agent-agent dependencies of targeting agent at surroundings, and the targeting strategies showed that they will intelligently use melee attack at nearer places and mage at more faraway areas. The model they showed went through very long time of training on more than 100 worlds, but I can only compare to one hundredth of it, because training time is limited and my CPU only support 4 worlds at most. Thus I am looking at a primitive stage of learning, and I will focus more on the progress of learning at different training times.

3 Environment

Our environment is based on the Neural MMO project developed by Openai[6, 7]. The word "MMO", or "MMORPG", stands for Massively Multiplayer Online Role-Playing Games. In human MMOs, developers aim to create balanced mechanics, while players aim to maximize their skills in utilizing them. Thus, one important purpose of this platform is to discover game mechanics that support complex behavior and agent populations that learns to make use of them.

Environment state. In this environment, state is represented by grid of tiles in the map. Each tile unit holds a particular texture with various properties. Types of textures include forest, scrub, grass, stone, lava, and water. Grass and forest tiles are traversable, while stone and water are not, and lava will kill an agent if stepping onto it.

Agent state and reward. For agents, there is no other objectives except survival. A reward $R_t = 1$ for each time step is provided to the agent while they are alive in the map. Initially, agents spawn at a random location along the edges of the environment. In order to remain healthy, agents must obtain food and water from time to time. Food can be collected (i.e., food bar will grow) when an agent is on a forest tile, and water can be collected if the agent is adjacent to a water tile. Water tile are like rivers and can provides unlimited water, but forest tiles have a limited supply of food, and will turn into grass if their capacity are used up (grass will also grow back to forest after some time). If agents starve or are thirsty, their health bar will shrink and finally reach zero (dies and be deleted from the environment). This means that agents will compete for food tiles while periodically refilling their water supply.

Agent actions. At each time step, agents may move one tile (towards North/South/East/West) and make an attack. There are 3 types of attacks: melee(short distance, big damage), range(larger distance and smaller damage), and mage(smallest damage, but has control effect of freezing the opponent temporarily). Being attacked will directly deduce the agent's health value. This enables the agent to develop different intelligent strategies of attack and defense in face of different scenarios.

Observation, action and reward. For each agent, each time step t, the reward for the agent is $R_t = 1$ if the agent is still alive at time t. The observation of the agent is a 15x15 square crop of surrounding map tiles. The properties also lists the occupying agents on the tiles. The action of an agent at each time step consists of a movement from the set of possible movements (north, south, east, west, Pass) and an attack from 3 types of attacks (melee, range, mage).

For each agent, the reinforcement learning problem is to obtain the best policy π^* which maximizes the expected total reward ($G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$).

4 Methodology

4.1 Experiments

We managed to train the agents using their given RL structure code for some time. First the NN was only a simple network with no hidden layers. Then thinking that this single layer may not have enough ability to learn complex policies, we changed the network structure (in `ann.py`) to one with 3 layers, as shown in Figure 1. The size of input h was set to the default value 32 in both trainings. (This can be changed because the input to this network is not the raw actions, but some packet that has been processed by the network defined in `IO` module.)

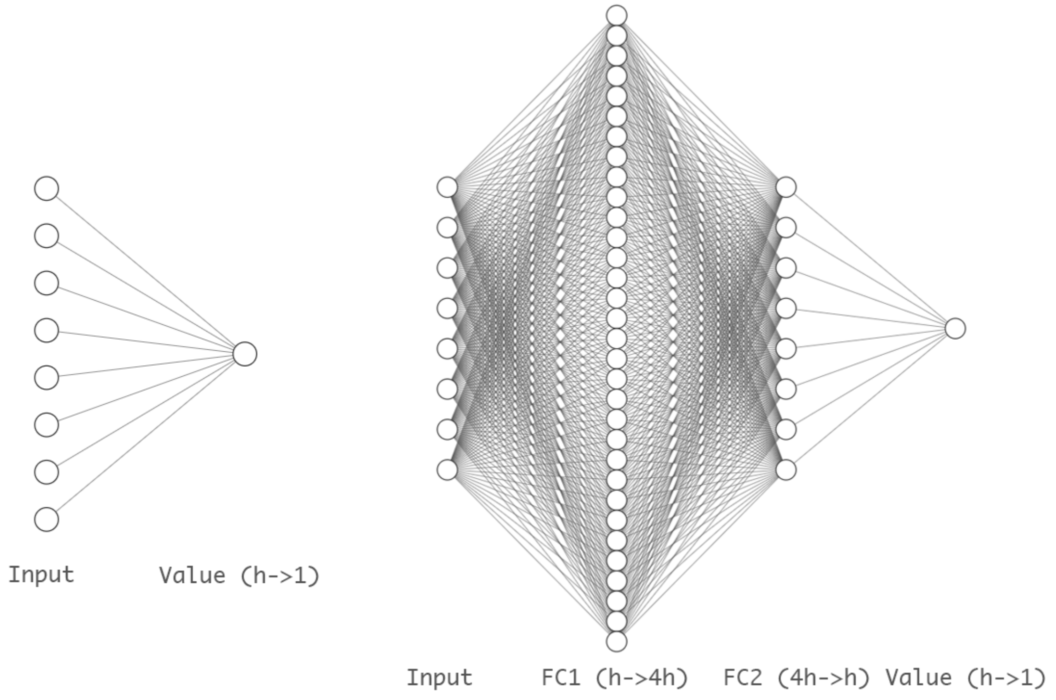


Figure 1: Network structure in two experiments.

With both network structures we trained the network for some time and observe their statistics. As shown in Figure 2, the learning curve is quite similar for the first 1000 server ticks. Later the curve goes up very slowly and it was not shown. One thing to notice is that the multi-layer NN actually learned not as well as the single-layer one, for the average and best lifetime of agents are all shorter than that trained by single-layer NN with same server ticks. However, we guess as long as the model has overcome the bad weights initialized, it may eventually be better because it can bare more complex behaviors. For training hours, for those 1103 ticks as shown in the picture, the simple NN took 12 h 56 min and the multi-layer one took 12 h 59 m, with negligible difference.

4.2 Understanding the Behavior.

To examine what behavior the agents have learned during training, we run some tests using the models saved at different ticks. During the test we use some of their internal API, add some logging codes to their project (in `God.py`) to log all the observations and actions at each tick. Then we wrote some scripts to analyse these logs and visualize the behavioral patterns of interest.

Movements. First we took a look at their moving behaviors at each tile. For simplicity we use the map No.0 that was generated by the Neural-MMO project upon setup. A processed map was shown in Figure 3. It was a 60 * 60 grid map surrounded by a border of grass and then 9-tile thick border of lava, which result in 80 * 80 in total. For convenience of referencing the locations we added grid lines and labelled row and column numbers. As mentioned in section 3, stepping into forest and beside water should be generally more favorable because agents can collect necessary water and food to

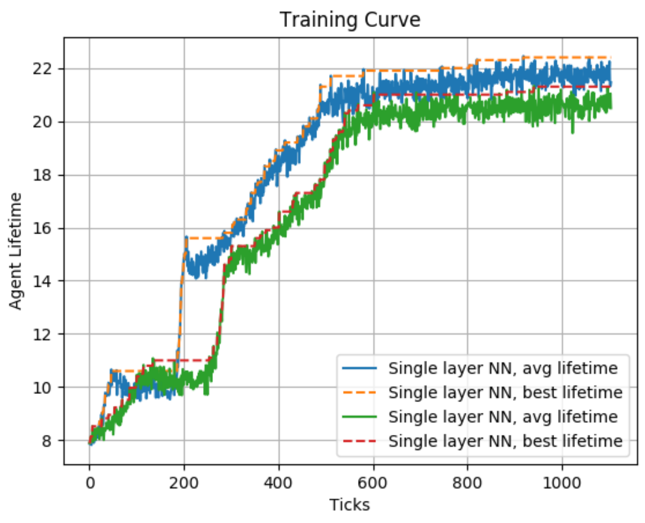


Figure 2: Learning curve with two network structures.

maintain health. Stones are not traversable and thus there's no point to move towards it. Lava are the most dangerous one because agents will die immediately if stepped into it, thus agents should learn to not reach out of the central area.

We took 3 models saved at tick 500, tick 1000 and tick 4500 respectively. Their agent average lifetimes are 18.44, 20.77 and 22.47 ticks.

At 500 ticks, we draw the chance of moving N/S/E/W in different positions in Figure 4. Because the map is too large, we only show the cases when agents are in the first row of grass (row 9 in Figure 3), from east to west (column 9 - 70), at each position. The table above lava shows frequency (in percentage) of each movement at the corresponding tile below. It can be seen that in a overall manner, agents like to move towards south, and the next preference is west. Moving towards south is reasonable, because forests and water are all in the south, and moving north instead will result in dying in lava. In most cases here should be no distinct difference in moving east or west, but agents move west probably because of the initialization of parameter still plays a role. It is also noticeable that in several tiles, agents still have certain frequency to go north and jump into lava, especially when moving south will not give it forest (column 9, 63). Moreover, the agents have also started to learn to avoid untraversable tiles, because their averaged chance of going south is smaller when the south tile is stone (40%) or water(32%), compared to traversable tiles of grass(46%) and forest(47%).

At training 1000 tick, the difference is little although their average lifetime has been longer. Generally, the probability of going south has increased, but not very much. The chance of going south has increased if the southern tile is water (32% to 53%), which seem like a degradation. What's worse, the chances of going north has increased a bit, especially in the left part (column 9 to 15). But good news is in the 9th column, the agent has improved in that its chance of going west into lava has deducted. It seems that agents are learning to go to a beneficial position, but the learning is not steady. As we only examined the moving behavior on a row, the improvement on lifetime could be due to the improvement of strategies in other positions.

Finally we look at the behaviors on 4500 training ticks, as shown in Figure 6. The biggest difference is that it finally overcome the influence of initialized parameters and learn to also go east. Perhaps due to this reason, the probability of going north has reduced in all cases of the southern tile. Moreover, the the agents don't go north that often. But besides this we can't see more improvements in their moving behaviors.

Attacks. Then we examine the preferences of agents using different attack strategies. As said, there are 3 types of attacks: melee (10 damage at 1 range), range (2 damage at 1-2 range), and mage (1 damage + freeze at 1-3 range). 1 range stands for the area with distance 1 including the diagonal tiles, that is, the 3 * 3 tiles centered at agent. It feels that the melee attack is dominantly preferable

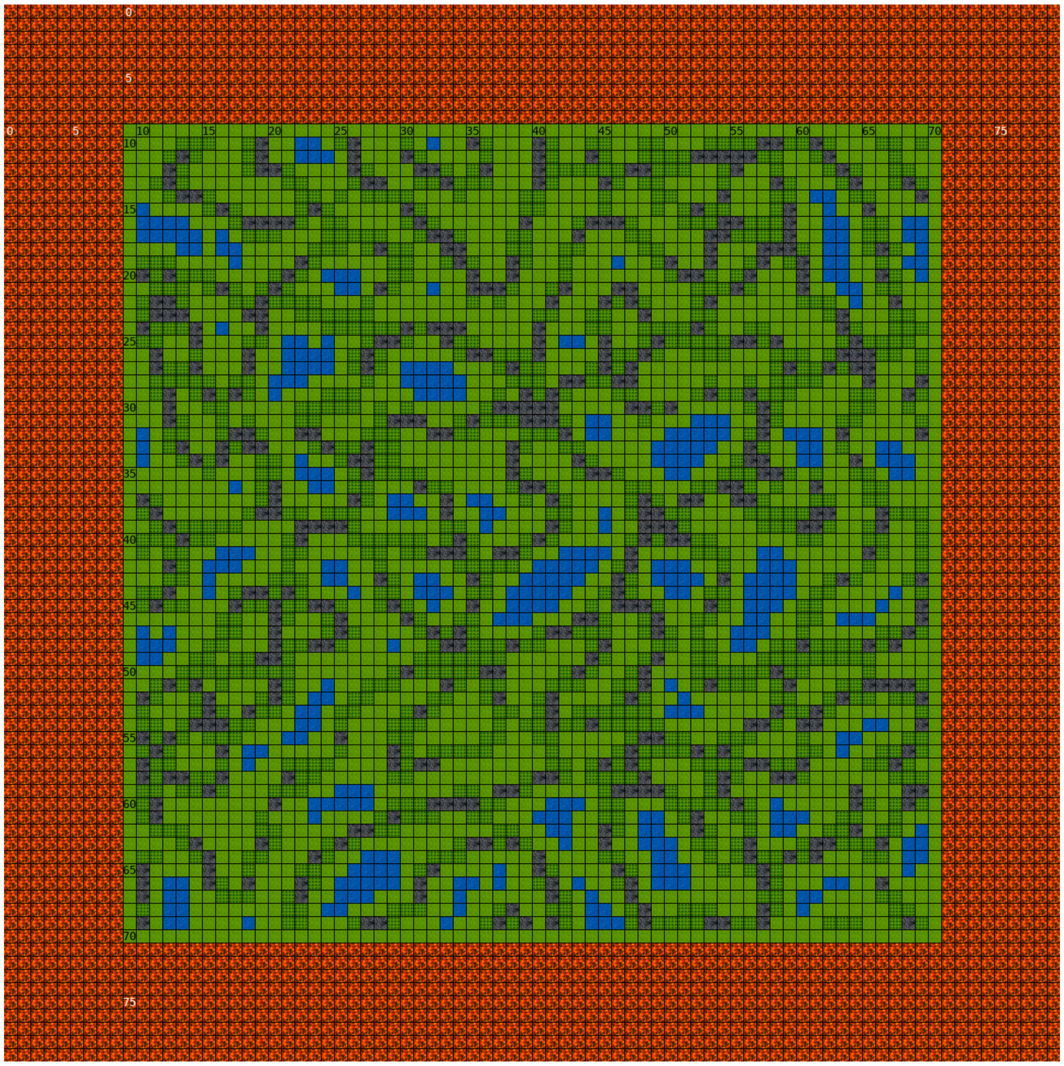
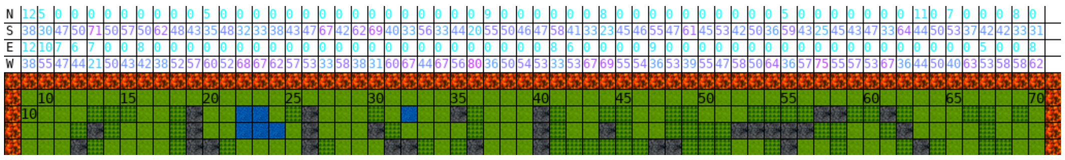


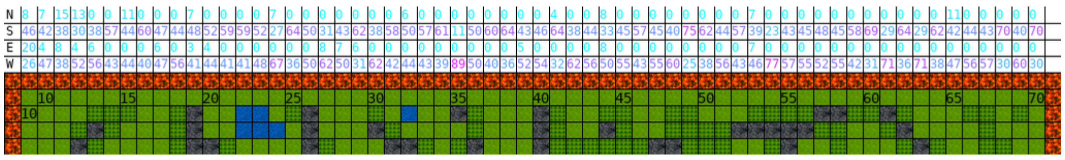
Figure 3: The map for testing.



N	12	5	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0
S	38	30	47	50	71	50	57	50	62	48	43	35	48	32	33	38	43	47	67	42	62	69	40	33	56	33	44	20	55	50	46
E	12	10	7	6	7	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	38	55	47	44	21	50	43	42	38	52	57	60	52	68	67	62	57	53	33	58	38	31	60	67	44	67	56	80	36	50	54

N	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	11	0	7	0	0	0	8	0
S	47	58	41	33	23	45	46	55	47	61	45	53	42	50	36	59	43	25	45	43	47	33	64	44	50	53	37	42	42	33	31
E	0	8	6	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	8
W	53	33	53	67	69	55	54	36	53	39	55	47	58	50	64	36	57	75	55	57	53	67	36	44	50	40	63	53	58	58	62

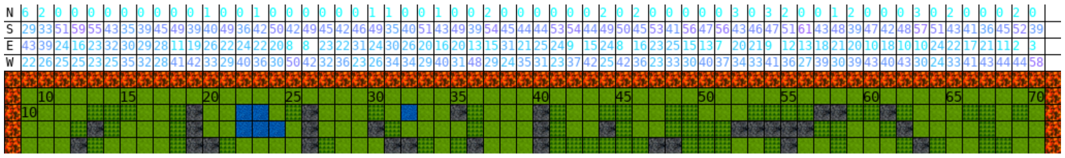
Figure 4: Agents' chance of moving to each direction at training tick 500.



N	8	7	15	13	0	0	11	0	0	0	7	0	0	0	0	7	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0
S	46	42	38	30	38	57	44	60	47	44	48	52	59	59	52	27	64	50	31	43	62	38	58	50	57	61	11	50	60	64	43
E	20	4	8	4	6	0	0	0	6	0	3	4	0	0	0	0	0	0	8	7	6	0	0	0	0	0	0	0	0	0	5
W	26	47	38	52	56	43	44	40	47	56	41	44	41	41	48	67	36	50	62	50	31	62	42	44	43	39	89	50	40	36	52

N	0	4	0	0	8	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0
S	46	64	38	44	33	45	57	45	40	75	62	44	57	39	23	43	45	48	45	58	69	29	64	29	62	42	44	43	70	40	70
E	0	0	0	0	8	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	54	32	62	56	50	55	43	55	60	25	38	56	43	46	77	57	55	52	55	42	31	71	36	71	38	47	56	57	30	60	30

Figure 5: Agents' chance of moving to each direction at training tick 1000.



N	6	2	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0	2	0	0
S	29	33	51	59	55	43	35	39	45	49	39	40	49	36	42	50	42	49	45	42	46	49	35	40	51	43	49	39	54	45	44
E	43	39	24	16	23	32	30	29	28	11	19	26	22	24	22	20	8	8	23	22	31	24	30	26	20	16	20	13	15	31	21
W	22	26	25	25	23	25	35	32	28	41	42	33	29	40	36	30	50	42	32	36	23	26	34	34	29	40	31	48	29	24	35

N	0	0	0	0	2	0	2	0	0	0	0	0	3	0	3	2	0	0	1	2	0	0	0	3	0	2	0	0	0	2	0
S	44	53	54	44	49	50	45	53	41	56	47	56	43	46	47	51	61	43	48	39	47	42	48	57	51	43	41	36	45	52	39
E	25	24	9	15	24	8	16	23	25	15	13	7	20	21	9	12	13	18	21	20	10	18	10	10	24	22	17	21	11	2	3
W	31	23	37	42	25	42	36	23	33	30	40	37	34	33	41	36	27	39	30	39	43	40	43	30	24	33	41	43	44	44	58

Figure 6: Agents' chance of moving to each direction at training tick 4500.

at a short distance, because the high damage can kill the opponent at once (although agent face the danger that the opponent also use this attack). And at larger distance, proper use of the mage attack can freeze the opponent and then it can come closer and launch more damaging attacks without being hurt.

We count the times of each attack type when some other agent is around itself. Figure 7, 8 showed the frequency of each attack type. It is drawn as the relative position to the agent, which is located at the center. It is strange that we only see attacks happened within 1 range, but in theory range and mage attack should be able to happen at 2 and 3 range. Within 1 range, at 500 training tick, the agents seem to prefer melee attack to range and mage, and this is clever because melee attacks have higher damage. But at 4500 training step, they seem to have no preference on the attack types. We may have missed to do some configurations on their project, otherwise the agents are not likely to show learning behavior like this but also keeps increasing their lifetime.

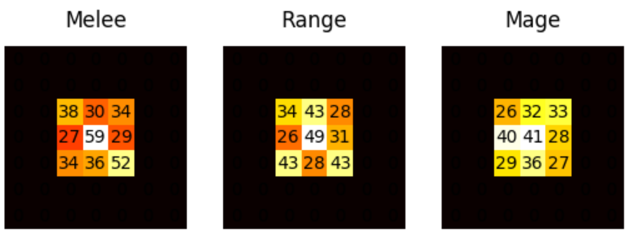


Figure 7: Count of attack at training tick 500.

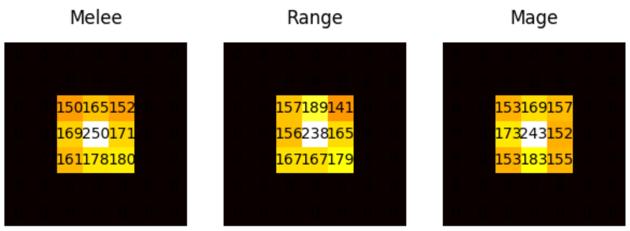
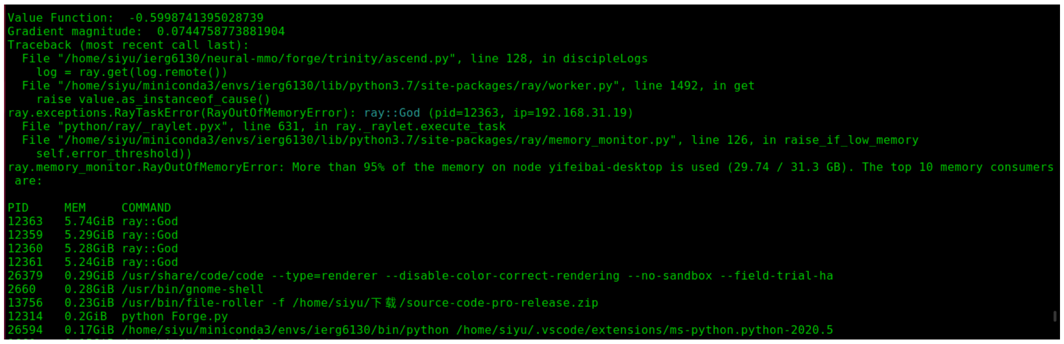


Figure 8: Count of attack at training tick 4500.

5 Discussion

The contents of this final report is a bit different from the aims declared at the proposal. This is because we underestimated the hardware requirement of this project and have to choose different perspectives to explore. Although their blog said the baseline can be trained on a single desktop CPU on its blog[1] (actually a older version of article said a single core), we found out later that it requires more than 32G of RAM and a quite performant CPU to enable training. On my consumer-level machine for daily research with 8G RAM, it basically cannot run at all, as it still require much space under my minimal configuration of the parameters. After moving to a 32G RAM machine, the training starts well but soon crashed after about 3 hours (see Figure 9), because the occupied RAM has gradually increased from about 70% proportion at beginning to 95% soon. In theory it is possible to train on cloud server, however, as we intended to add new features to the whole project in the proposal, this involves changes in the Unity client, and debugging is too hard without a graphical interface. Thus I have to drop the aims stated in the proposal, and do something that is lighter, i.e., accomplishable with a command-line server and my desktop.



```
Value Function:  -0.5998741395028739
Gradient magnitude:  0.0744758773881904
Traceback (most recent call last):
  File "/home/siyu/ierg6130/neural-mmo/forge/trinity/ascend.py", line 128, in discipleLogs
    log = ray.get(log.remote())
  File "/home/siyu/miniconda3/envs/ierg6130/lib/python3.7/site-packages/ray/worker.py", line 1492, in get
    raise value.as_instanceof_cause()
ray.exceptions.RayTaskError(RayOutOfMemoryError): ray::God (pid=12363, ip=192.168.31.19)
  File "python/ray/_raylet.pyx", line 631, in ray._raylet.execute_task
  File "/home/siyu/miniconda3/envs/ierg6130/lib/python3.7/site-packages/ray/memory_monitor.py", line 126, in raise_if_low_memory
    self.error_threshold))
ray.memory_monitor.RayOutOfMemoryError: More than 95% of the memory on node yifeibai-desktop is used (29.74 / 31.3 GB). The top 10 memory consumers
 are:

PID     MEM     COMMAND
12363   5.74GiB ray::God
12359   5.29GiB ray::God
12360   5.28GiB ray::God
12361   5.24GiB ray::God
26379   0.29GiB /usr/share/code/code --type=renderer --disable-color-correct-rendering --no-sandbox --field-trial-ha
2660    0.28GiB /usr/bin/gnome-shell
13756   0.23GiB /usr/bin/file-roller -f /home/siyu/下载/source-code-pro-release.zip
12314   0.2GiB  python Forge.py
26594   0.17GiB /home/siyu/miniconda3/envs/ierg6130/bin/python /home/siyu/.vscode/extensions/ms-python.python-2020.5
```

Figure 9: Training crashed on a 32G RAM machine.

We looked at the moving and attacking behaviors in a separate way. In fact, the attack and move behaviors are correlated to each other, for example, if the agents saw another agent in its neighborhood, it may try to move towards (ready to attack) or away from (protect itself) that agent. Also, use what type of attack may also be influenced by what direction the agent sees of greater value. We didn't take that into consideration when processing the log data because we think the effect is not large at current stage. After all, because as agents are not so many and die quite soon, their chance of attacking each other are not high, which can be seen in the amount of attacks we have collected.

We also would like to examine more intelligent behaviors which may emerge when the training time is much longer, but because of limited training time, our model saved at tick 4500 can only survive 23 ticks which is just minor improvement compared to model at 500 ticks. Motivated by the previous paragraph, it could be interesting if we can first disable the attack behaviors and only train to move, and then later train to attack, and combine them together. Without the interference, agents may be able to learn better and faster. This could be done if we can put more time into it. While doing the analysis we found they also provided some logging functions and simple scripts, but we didn't use that because understanding details of their large project takes time and writing by myself will be faster.

6 Conclusion

This project tried to study the behaviors learned by agents in the Neural-MMO environment and take a look at how they gradually learned more intelligent strategies. Because our limited training time and training power, we only managed to observe the cases at quite early training stage, when the agents have average lifetime of more than 20 ticks.

By looking at their moving behaviors in part of the map, we found that agents have basically learned to avoid stepping into fatal lava at very early stage (500 ticks), and slightly knows to go to traversable tiles to avoid wasting life. However, the preference for nutrition (forest, water) is not clear. For attacks, the analysed data showed no strategies in choosing different attack types. These are generally consistent with the fact that the agents are still short-lived. They may be lucky to cross a forest and find waters, because there are quite a lot of them in the map.

References

[1] Neural mmo: A massively multiagent game environment. `https://openai.com/blog/neural-mmo/`. (Accessed on 05/20/2020).

[2] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[3] J. A. Calvo and I. Dusparic. Heterogeneous multi-agent deep reinforcement learning for traffic lights control. In *AICS*, pages 2–13, 2018.

[4] M. Hüttenrauch, A. Šošić, and G. Neumann. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*, 2017.

[5] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

[6] J. Suarez, Y. Du, P. Isola, and I. Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019.

[7] J. Suarez, Y. Du, I. Mordach, and P. Isola. Neural mmo v1. 3: A massively multiagent game environment for training and evaluating neural networks. *arXiv preprint arXiv:2001.12004*, 2020.

[8] Y. Yang, L. Yu, Y. Bai, Y. Wen, W. Zhang, and J. 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*, pages 2133–2135. International Foundation for Autonomous Agents and Multiagent Systems, 2018.