Simulation of Predator Prey Dynamics using Deep Reinforcement Learning

(CS 275: Artificial Life for Computer Graphics and Vision - Course Project)

Rahul Dhavalikar
University of California, Los Angeles
Email: rahul.dhavalikar@cs.ucla.edu

Anoosha Sagar
University of California, Los Angeles
Email: anoosha.sagar@cs.ucla.edu

Maithili Bhide
University of California, Los Angeles
Email: maithili.bhide@cs.ucla.edu

Akshay Sharma
University of California, Los Angeles
Email: akshaysharma23@cs.ucla.edu

***Abstract*— In this project we simulate predator-prey dynamics in a multi-agent environment where both the predator and the prey evolve together. Each of our agents is assigned a deep reinforcement learning model (DQN) trained using Tensorflow to determine its actions. The goal of each of agent is to maximize its reward in this mixed cooperative and competitive environment. We are using OpenAI gym to simulate the environment and have explored different scenarios with varying number of agents(eg. 1v1, 1v2, 2v1). We also introduced additional artefacts in the environment such as obstacles and food to observe the change in the dynamics of our predator-prey agents. Currently, our environment returns a list of observations based on the actions performed by the agents in constrained space. From our simulation of various scenarios, we observe that the green agents adopt strategies such as splitting up in order to confuse the red agent and effective use of its speed to gather food. Similarly, red agents evolve to create triangular formations and man-to-man marking strategies to corner the green agents and gain rewards for colliding with them.**

***Keywords* - Reinforcement Learning, DQN, Predator-Prey Environment, Multiagent Environment, OpenAI Gym**

I. Introduction

Predator-prey dynamics constitute one of the most fundamental relationships in nature. The predator and prey are a part of the same environment and evolve different abilities such as stealth, speed, camouflage in order to survive as a species. This relationship was studied as early as 1910 and formalized by the Lotka-Volterra predator-prey model[1] to explain the dynamics of natural populations of predators and prey, such as the lynx and snowshoe, moose and wolf or more simply rabbits and foxes.

Fig. 1 A predator-prey example in nature

This fundamental survival relationship of predation is the motivation of our project. Inspired by the work of researchers at UC Berkeley and OpenAI on Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments[2] we have implemented several predator-prey scenarios that are commonly observed in nature. Our aim is to vary the environment and number and type of agents to represent the different conditions in nature and observe the changes to the evolution of our predator-prey agents. The results of each of these scenarios will be presented in this report.

II. Literature Review

A. Deep Reinforcement Learning

Tasks related to physical control have continuous (real valued) and high dimensional action spaces. An approach to extending reinforcement learning methods designed for discrete action spaces to the continuous domain is simply discretizing the action space. In [3] is presented a deep learning model to learn control policies directly from high-dimensional sensory input. The model is a convolutional neural network, trained with a variant of Q-learning trained to play 7 different Atari games.

Deep Q networks combine Q-learning with a flexible deep neural networks. Three major components present in DQN -

- Addition of convolutional/ dense layers
- Experience replay mechanism
- Utilization of a separate target network

The variant of online Q-learning used in this paper combines stochastic mini-batch updates with experience replay memory to ease the training of deep networks for RL. Storing an agent's past experiences and randomly drawing training batches prevents the network from only learning about what it is immediately doing in the environment, and allow it to learn from a more varied array of past experiences making it more robust in the task.

B. Double DQN

Deep Q-Learning (DQN) suffers from overestimations in some games in the Atari domain. In [4], researchers have proposed a Double Q-learning algorithm that can be

generalized to work with large-scale function approximation. This algorithm reduces the overestimations in the Atari domain and leads to better performance on several other games. In Double DQN, this is achieved by decomposing the max operation in the target into action selection and action evaluation. The primary network to chooses an action, and the target network generates the target Q-value for that action.

$$Q\text{-}Target = r + \gamma Q(s', argmax(Q(s', a, \Theta), \Theta'))$$

This decoupling of the action choice from target Q-value generation reduces overestimation, trains the model faster and more reliably. This helps agents learn optimal policy even if suboptimal actions were regularly given higher Q-values than optimal actions.

C. Dueling DQN

In [5], a new neural network architecture for model-free reinforcement learning is proposed. It makes use of two separate estimators - one for the state value function and one for the state dependent action-advantage function. It then combines them back into a single Q-function at the final layer. This new structure generalizes learning across actions without imposing any change to the underlying reinforcement learning algorithm. More robust estimates of state value by decoupling it from the necessity of being attached to specific actions.

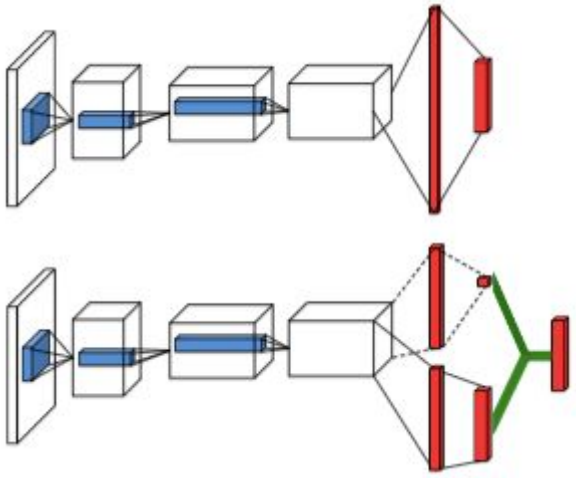


Fig. 2 A popular single stream Q-network (top) and the dueling Q-network (bottom) [5].

The DQNs share a common feature learning module in this architecture. This approach gives dramatic improvements over existing deep RL approaches in the Atari domain.

III. Environment

We use the multi-agent extension of the OpenAI Gym framework[6] to set up our predator-prey environment. The framework defines the state and action space for every agent based on the environment, allowing us to focus on training the agents to act intelligently and simulate predator-prey dynamics. The framework is turn based, i.e., each agent performs an action (including no action) at every time step. The environment also provides each agent with a reward and an observation describing the environment's state. The goal of each agent is to maximize its own reward and this leads to the development of various cooperative strategies which will be described in the further sections. The environment also provides location details of the agents and other artefacts in the environment such as food or obstacles.

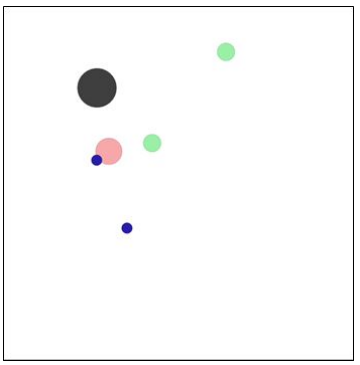


Fig. 3 Representative image of our environment with 1 predators (red agents), 1 prey (green agent), 2 food items (blue), and 1 obstacle (black).

In the predator prey environment, slower (red) agents chase faster (green) adversaries. In addition, there are food items (blue) which the green agents wish to consume and obstacles (black) which can be used by the green agents to distract and confuse the red agents. Multiple agents can exist on either team and the goal of each agent is to maximize its own reward in this mixed cooperative and competitive setting. The environment returns a list of states, one per agent, once each agent performs an action. An agent's state consists of the (1) agent's position and velocity, (2) the relative position of any landmarks or food, (3) the relative position of all other agents. The agents receive a negative reward if they leave the arena and the environment is reset with the next episode beginning in a random configuration. This trains our agents to operate within the confines of the environment. An episode also ends and is reset if it has completed 3000 steps. The episodes do not end when the green and red agents collide, instead we keep a track of the number of collisions in the episode.

All red agents receive a positive reward if they are able to successfully intercept any green agent. The green agents receive a corresponding negative reward if they are intercepted by the red agents. To help training converge faster, we also enabled an L2 penalty based on the distance formula. Red agents are given a negative reward proportional to their distance from green agents to prevent them from drifting aimlessly during the early stages of training. Green agents are given a positive reward proportional to their distance from red agents to similarly help them train faster. Green agents additionally get a positive reward based on their distance from the food and a higher positive reward for actually eating the food. There is also a negative reward for the green agents at the boundary of the arena, so that the green agents do not exit the arena easily. The negative reward proportional to distance for the red agent is not only based on its own distance from green agent but is also dependent on the distances of other red agents from green agents. Similarly when any one of the red agent collides with a green agent, all the red agents receive the

same positive reward. This kind of a reward system where the actions of one agent affect the rewards for other agents of its kind leads to formation of cooperative strategies as is discussed later and is evident from the simulation.

The action space consists of a list of actions, one per agent. Each agent's action is described by 5 numbers, which represent whether an agent should stay put or move up/down/left/right (this is specified by the OpenAI Gym environment).

IV. Network Architecture and Training

Each agent in our multi-agent predator-prey environment is trained using a deep Q network model developed in Tensorflow. Each model receives as input the current state, next state, action and reward of the agent it is modeling at every time step. For each episode we are keeping track of the episode rewards, collision count of agents and number of steps of the episode. The goal of our model is to minimize the loss (MSE) and thereby maximize the reward of the agents.

Our deep reinforcement learning model has 2 dense layers, each with 50 hidden units. The first dense layer receives an input as the current state of environment.The output from the first dense layer is fed to the second dense layer. The output from the second dense layer is split into 2 separate streams - (1) Advantage Stream (2) Value Stream. We then perform a Xavier initialization on these streams in order to get Advantage Weights and Value Weights. These weights are combined together to get the final Q-values. We then calculate the error between the target Q-values and the predicted Q-values and use the mean squared error to optimize the loss of our model. All models are trained with the Adam Optimizer and with a learning rate of 0.0001 for 20,000 episodes each.

There is an experience replay buffer associated with each agent to learn from its past experiences. The scenarios starts with some pre-train steps where the agents take random steps for some initial experience and exploration of the arena. The actual training of the model begins after these pre-train steps. We also allow the agents to take random actions at some steps. Initially, for more exploration, the chance of random actions for the agents are high but as the model is trained and gets better, we reduce the chance of random actions and instead use the actions predicted by the network.

V. Experiments & Results

We have experimented with different scenarios by varying the number of agents and by adding additional entities in our environment. In each of the scenarios below, we have described the environment and noted some interesting patterns and observations from the simulation. We also keep a track of the number of steps, the average rewards per agent and the number of collisions in each episode. We plot a graph for these results and use moving average trend lines to observe some trends in these graphs. The results are compiled after this section.

A. *Simple 1 vs 1*

In this scenario, there is one prey (green agent) and one predator (red agent). We can see that the predator continuously chases the prey to gain reward while the prey has to keep running for gaining reward. As the green is faster than the red agent, we see it develop interesting patterns to avoid being hit. Despite the green agent being faster than the red agent, the red agent still manages to hit the green agent as evidenced in the collisions graph in our results section. The moving average trend line shows that the overall episode duration(number of steps) increases showing that the agents learn to stay inside the arena while trying to maximize their rewards.

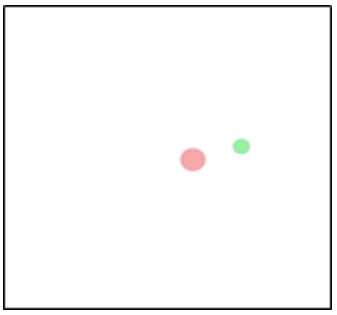


Fig. 4 Simple 1vs1 environment

B. *Simple 1 vs 2*

In this scenario, there is one prey (green agent) and two predators (red agents). Because of the reward system we observed some cooperative strategies by the red agents in our simulations to capture the prey. The red agents corner the green agent by getting into a triangular configuration. Then instead of both rushing in, one the red agent goes in to attack the prey while the other one stays back to intercept the green agent if it escapes. This scenario is quite similar to the predator-prey dynamics observed in nature. As expected we get higher collisions in this scenario as compared to the previous 1 vs 1 scenario.

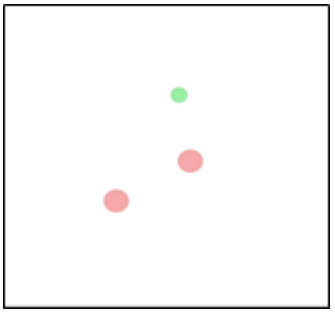


Fig. 5 Simple 1vs2 environment

C. *Simple 2 vs 1*

In this scenario, there are two prey (green agents) and one predator (red agent). In our simulation we observed that the green agents try to stay together away from the red agent but

when the red agent comes in to attack, they split up and move in opposite directions. We observed that the predator in this case focuses on only one of the prey and chases it constantly. This helps the predator to ensure it intercepts at least one prey.

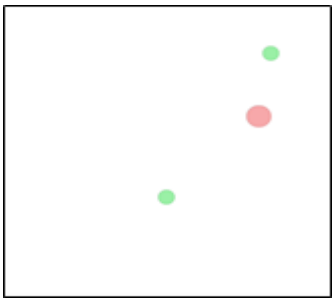


Fig. 6 Simple 2vs1 environment

D. *Complex 1 vs 1*

In this scenario, there is one prey (green agent) and one predator (red agent). In addition to the agents, we also introduced an obstacle (black) and food (blue). This is similar to the 1 vs 1 scenario but since food gives the green agents a positive reward, the agent try to get the food whenever possible while avoiding the red agent as the negative reward for colliding with the red agent is double as compared to the positive reward it will get for getting the food. In this experiment, we also observed that the green agent tries to use the obstacle as a shield from the red agent and tends to move around the obstacle instead of randomly as in the initial 1vs1 scenario. Also due to the presence of food, the green agent spends more time in the arena thereby significantly increasing the number of steps for each episode as can be observed from the graph. This has led to a complete reduction in the episodes where the green arena exits the arena to avoid being chased by the red agent as visible in the rewards graph for this scenario.

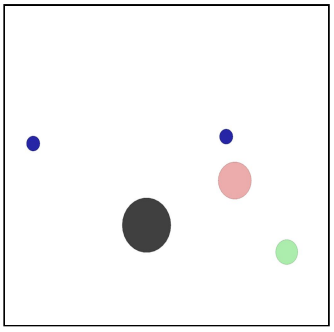


Fig. 7 Complex 1vs1 environment with obstacle and food

E. *Complex 1 vs 2*

In this scenario, there is one prey (green agent) and two predators (red agents) along with food and obstacles in the environment. We observed that the episodes are relatively shorter in this scenario due to the presence of two red agents which aggressively pursuit the green agent. The faster green agent tries repeatedly to get closer to the food and eat it and succeeds many at times but soon one of the red agents starts marking the green agent making it harder for it to eat more food. On the other hand, the second red agent provides cover and stays behind to prevent the green agent from moving too far. On many occasions, both the red agents corner the green agent and repeatedly hit it causing it to leave the arena and end the episode. In this scenario, we observe many collisions between the red and green agents due to the aggressive pursuit and cornering strategies adopted by the red agents.

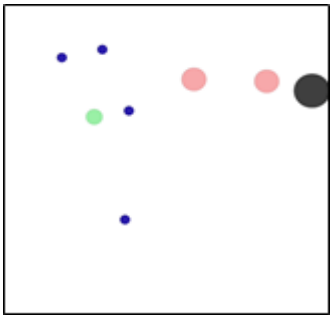


Fig. 8 Complex 1vs2 environment with obstacle and food

F. *Complex 2 vs 1*

In this scenario, there are two preys (green agents) and one predator (red agent) along with food and obstacles in the environment. In our simulation of this scenario, we observed that the primary instinct of the red agent is to attain a position on the field from where it can easily chase either of the two green agents. As the reward for maintaining a large gap between the green and red agent is greater than the reward for being closer to the food, the green agents spend most of the time maintaining the distance between them and the red agent and seek out food less frequently. As observed for the previous complex scenarios, the episode length is considerably longer with maximum episodes reaching the maximum threshold of 3000 steps.

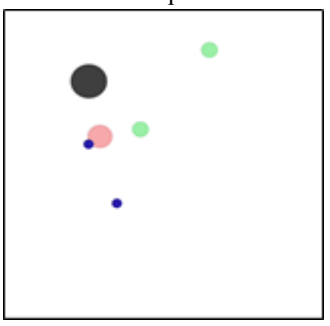


Fig. 9 Complex 2vs1 environment with obstacle and food

VI. Summary

To summarize, we observed the effects of our reward system on the behavior of our predator-prey agents in a variety of different scenarios. From the results and videos, we demonstrate the cooperative and competitive strategies evolved by the agents in our environments. The results that we obtained are similar to the dynamics we observe in nature.

The review of different reinforcement learning methods provided us with the motivation to design our deep reinforcement learning model for the agents.

As future work, we can have some additional actions for the agents like a sprint action for the red agents to increase its speed for a short duration of time to catch the green agent. We can also add more elements in the environment like forests where the green agents can hide. We can also explore how other reinforcement learning algorithms like policy gradient algorithms, actor-critic algorithms can perform in these scenarios while also having different algorithms for the red and green agents.

References

[1] Wikipedia contributors. (2018, March 3). Lotka–Volterra equations. In Wikipedia, The Free Encyclopedia. Retrieved 18:33, March 21, 2018, from https://en.wikipedia.org/w/index.php?title=Lotka%E2%80%93Volterra_equations&oldid=828624027

[2] Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in Neural Information Processing Systems. 2017.

[3] *Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).*

[4] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." AAAI. Vol. 16. 2016.

[5] Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." arXiv preprint arXiv:1511.06581 (2015).

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

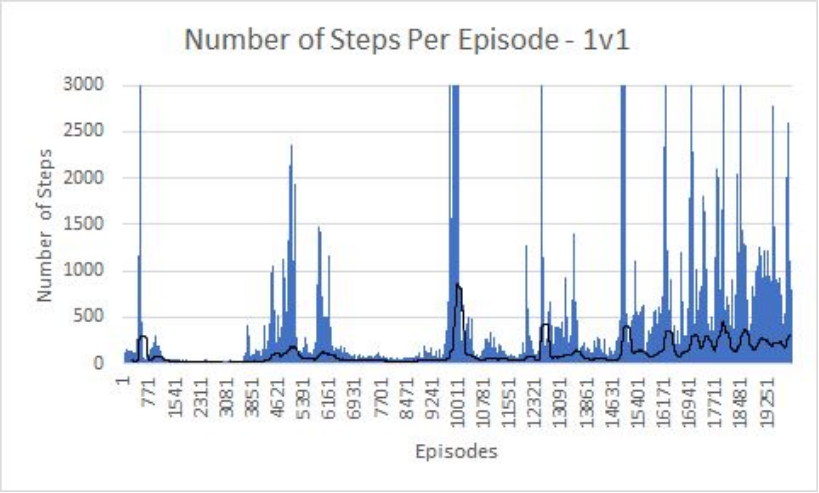


Fig. 10 Simple 1vs1 environment

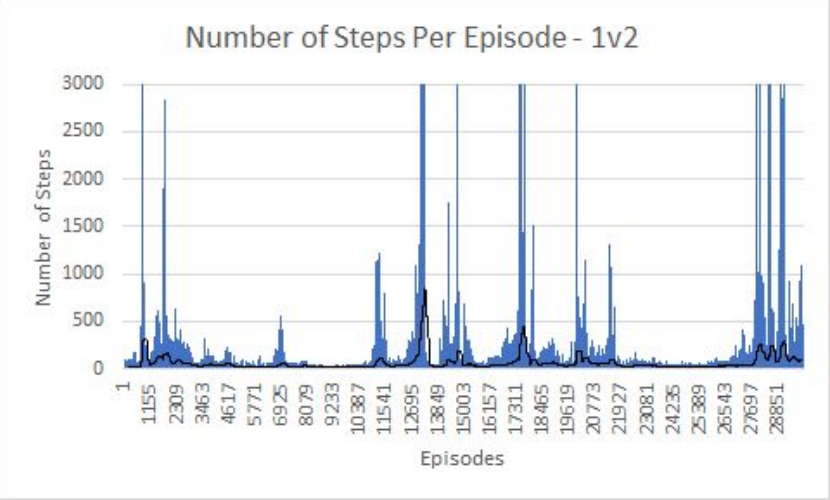


Fig. 11 Simple 1vs2 environment

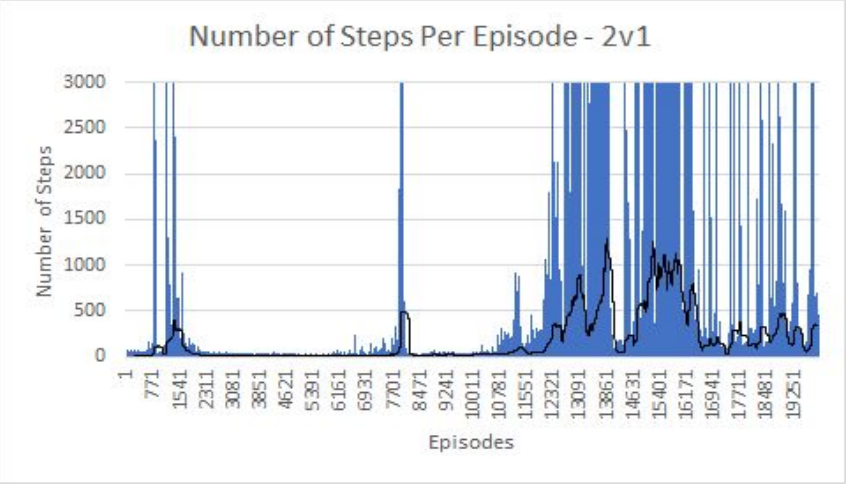


Fig. 12 Simple 2vs1 environment

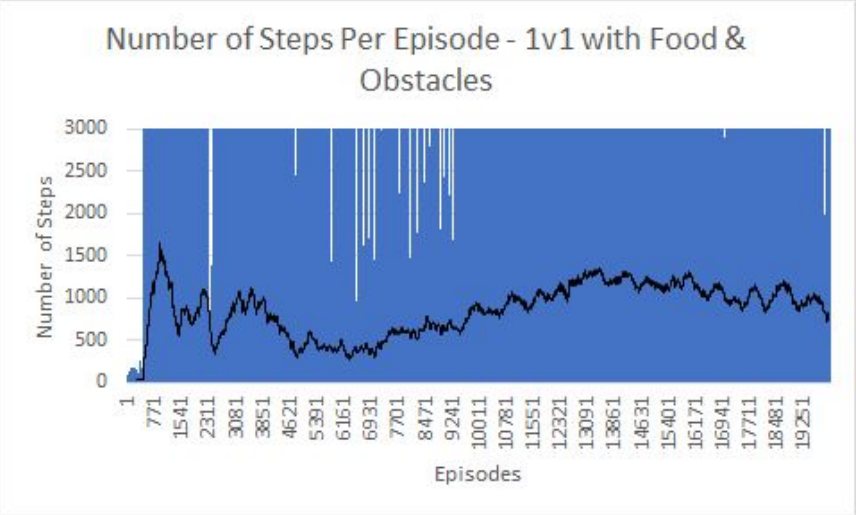


Fig. 13 Complex 1vs1 environment with obstacle and food

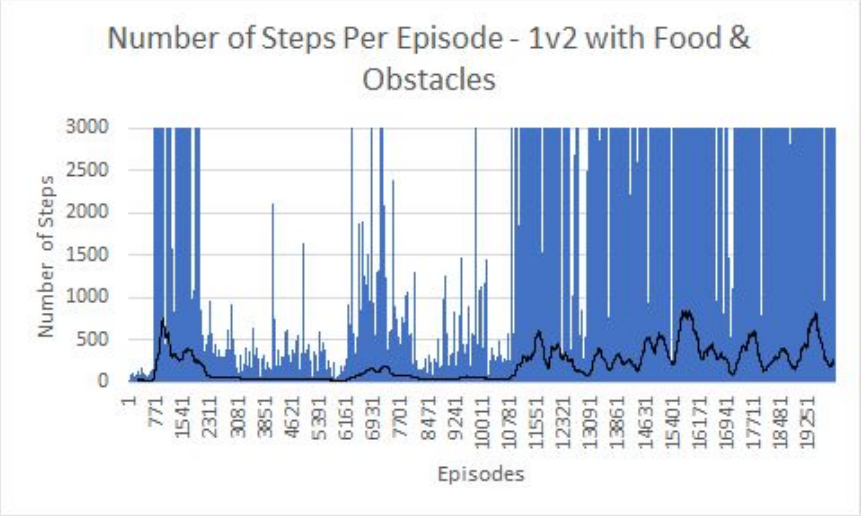


Fig. 14 Complex 1vs2 environment with obstacle and food

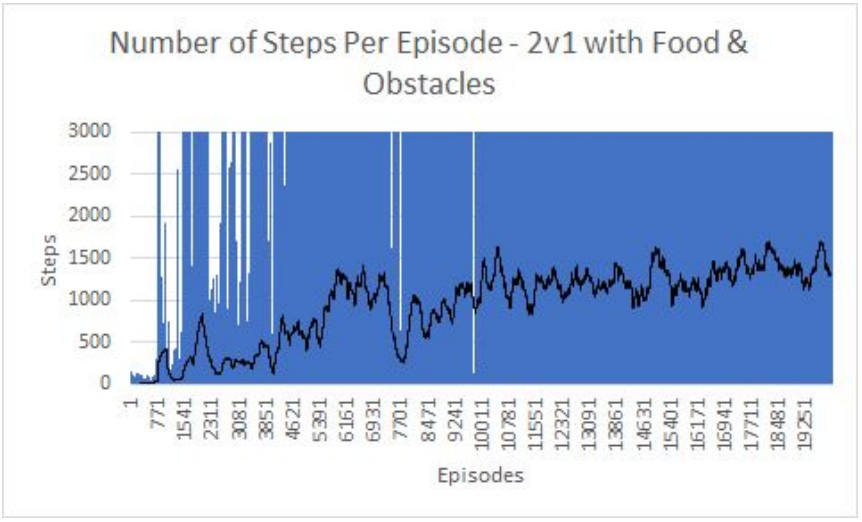
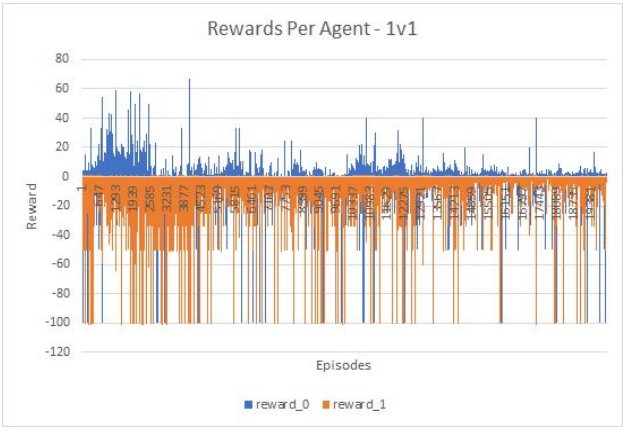


Fig. 15 Complex 2vs1 environment with obstacle and food



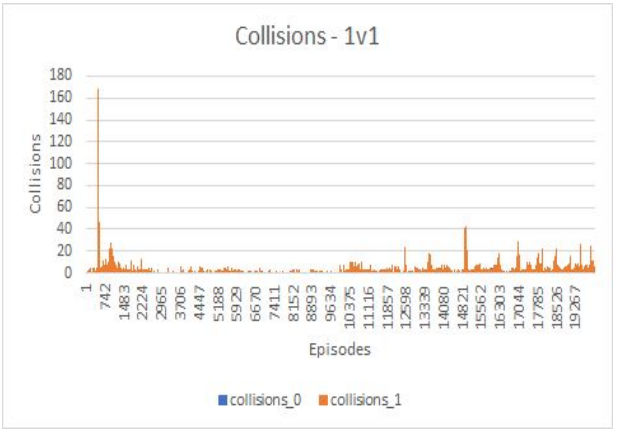
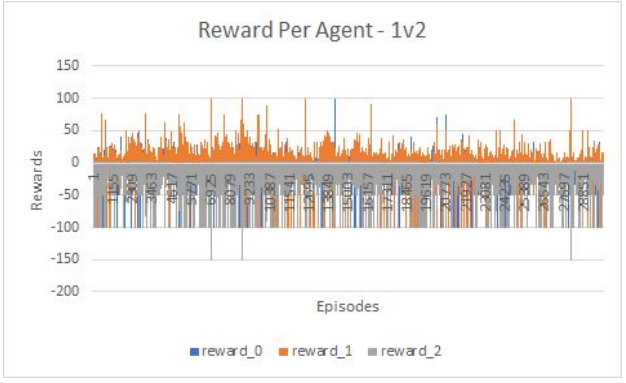


Fig. 16 Simple 1vs1 Environment



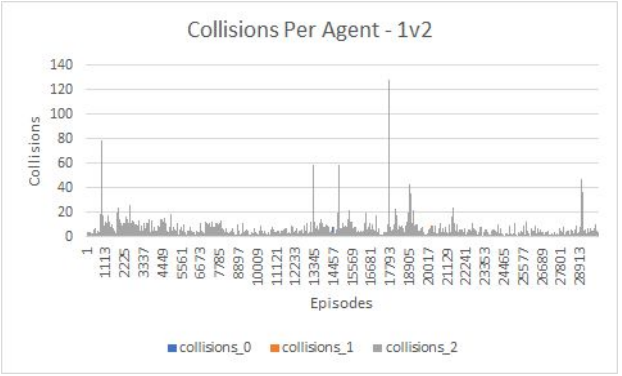
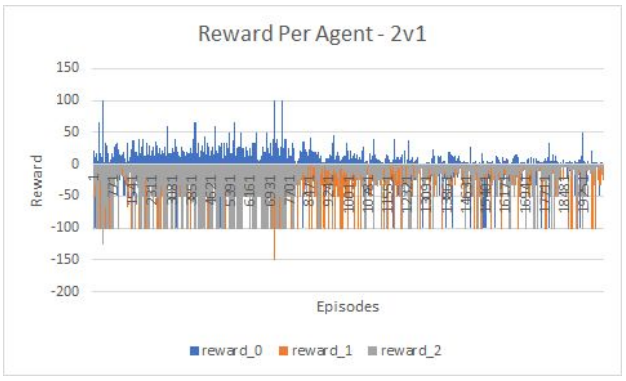


Fig. 17 Simple 1v2 Environment



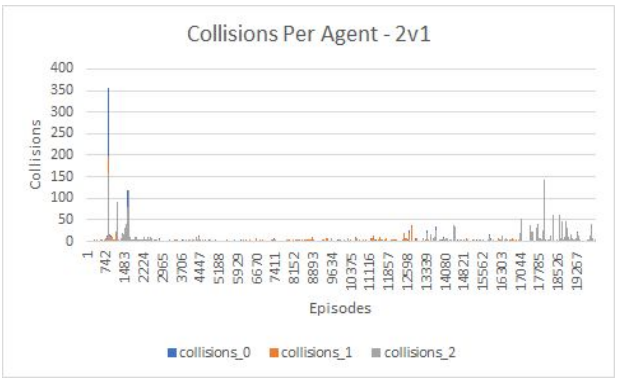
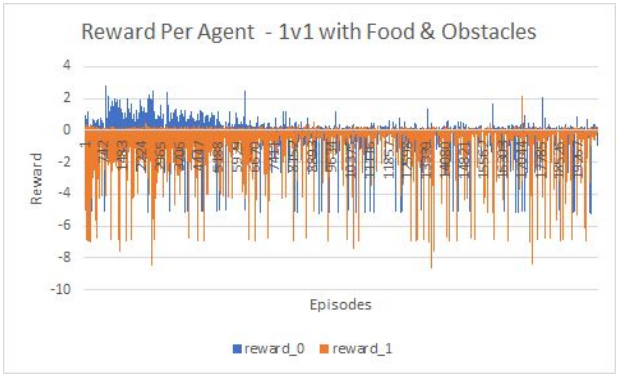


Fig. 18 Simple 2vs1 Environment



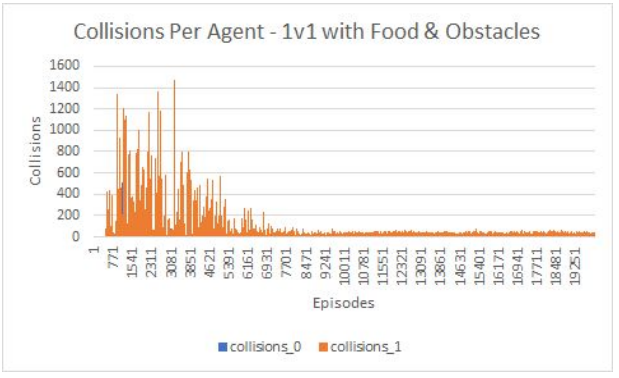
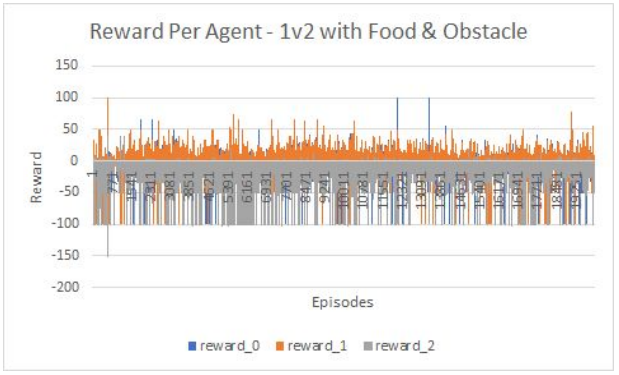


Fig. 19 Complex 1vs1 environment with obstacle and food



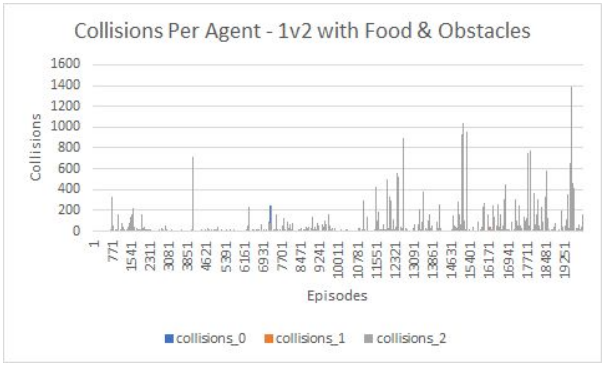
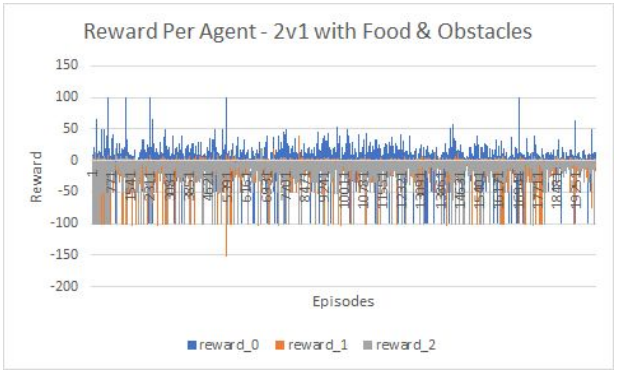


Fig. 20 Complex 1vs2 environment with obstacle and food



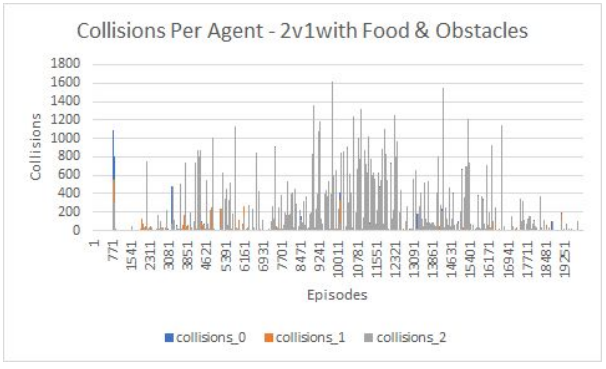


Fig. 21 Complex 2vs1 environment with obstacle and food