CSE 546: REINFORCEMENT LEARNING

Multi-Agent Reinforcement Learning

Nitin Kulkarni: 50337029

Sai Reddy: 50315301

Sehaj Grover: 50336839

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

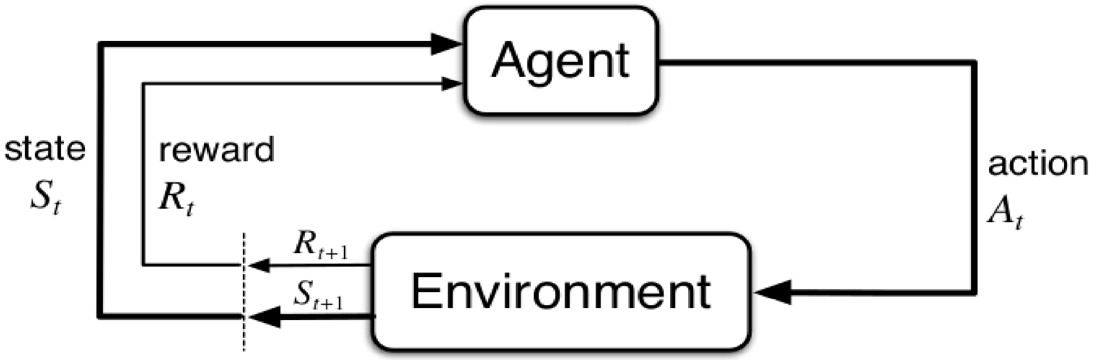
Project Description:

Our aim is to build a Multi-Agent Multi Objective environment, solve it using Tabular and Deep RL methods, and apply the Deep RL methods on an existing MARL environment (Predator-Prey).

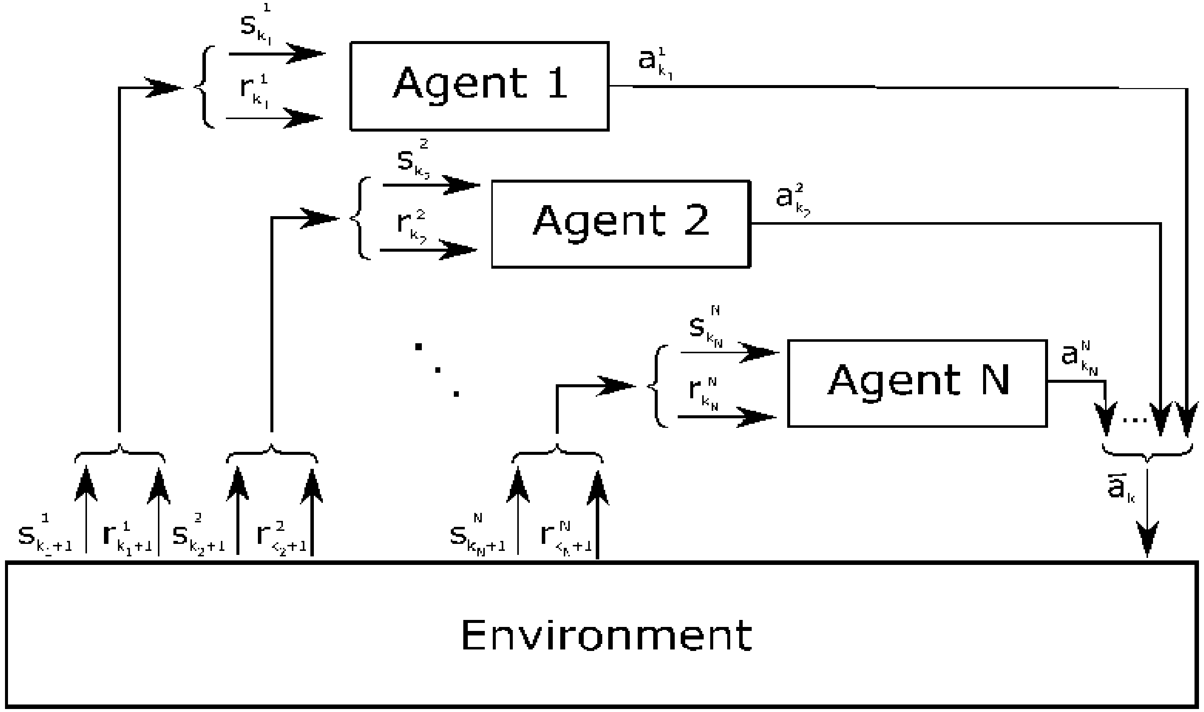
For Tabular methods, we implemented Q-learning and for Deep RL methods we implemented DQN, Double DQN, and Advantage Weighted Regression.

BACKGROUND

What is Multi-Agent RL?



Single Agent RL



Multi-Agent RL

Why is Multi-Agent RL Challenging?

- Joint Action Space
- Game-Theoretic Effects
- Credit Assignment
- Lazy Agent Problem
- Non-Markovian Nature of Environments
- Non-Unique Learning Goals
- Non-Stationarity
- Scalability
- Various Information Structures

IMPLEMENTATION

Algorithms:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Algorithm 1 Advantage-Weighted Regression

$\pi_1 \leftarrow$ random policy
$\mathcal{D} \leftarrow \emptyset$
for iteration $k = 1, ..., k_{\max}$ **do**
  add trajectories $\{\tau_i\}$ sampled via π_k to $\mathcal{D}$
  $V_k^{\mathcal{D}} \leftarrow \arg\min_V \ \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left|\left|\mathcal{R}^{\mathcal{D}}_{\mathbf{s},\mathbf{a}} - V(\mathbf{s})\right|\right|^2\right]$
  $\pi_{k+1} \leftarrow \arg\max_\pi \ \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\mathcal{D}}_{\mathbf{s},\mathbf{a}} - V_k^{\mathcal{D}}(\mathbf{s})\right)\right)\right]$
end for

Q-Learning

Advantage Weighted Regression

Algorithms:

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

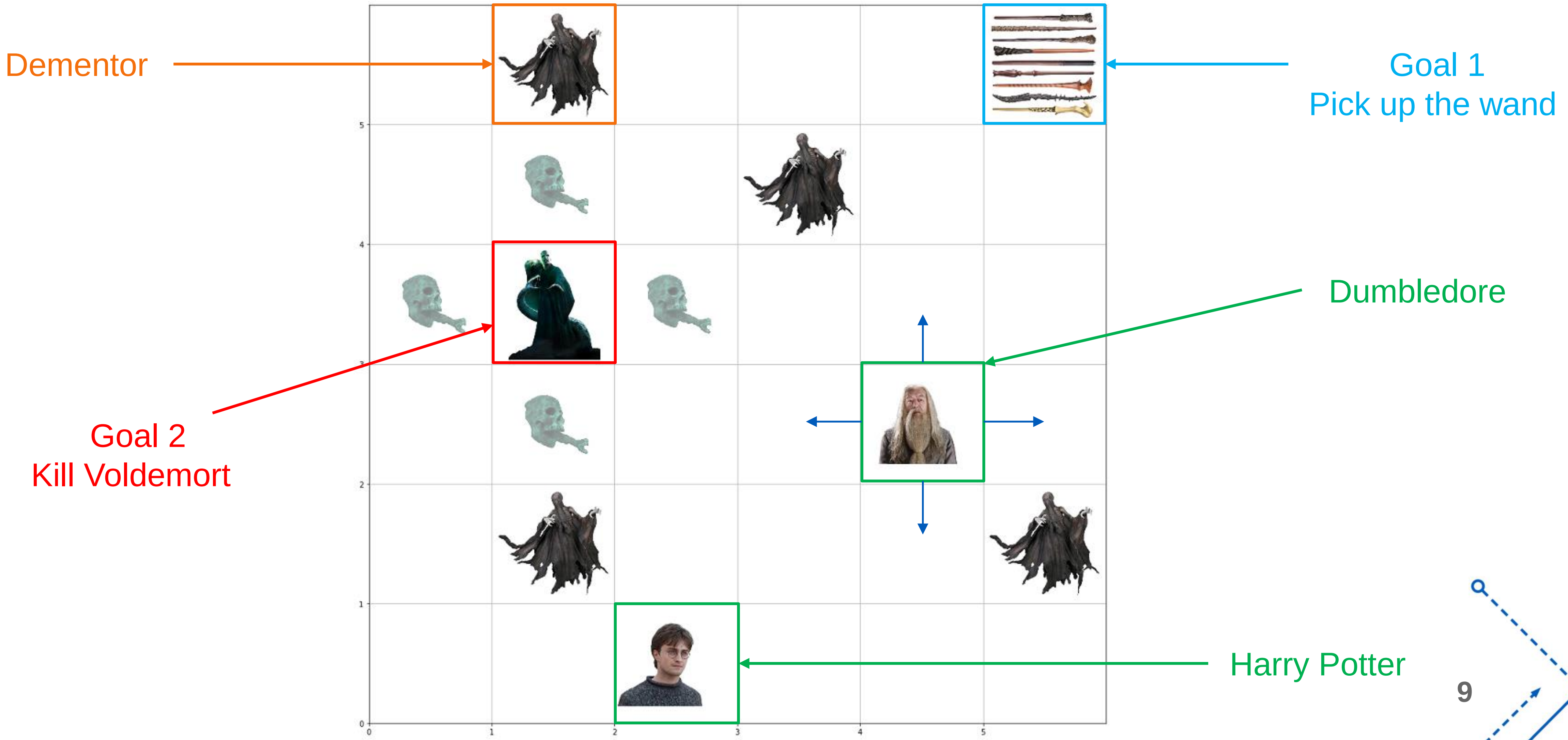
DQN

Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau << 1$
for each iteration **do**
  for each environment step **do**
    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$
    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
  for each update step **do**
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute target Q value:
      $Q^*(s_t, a_t) \approx r_t + \gamma\, Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$
    Perform gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$
    Update target network parameters:
      $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$

Double DQN

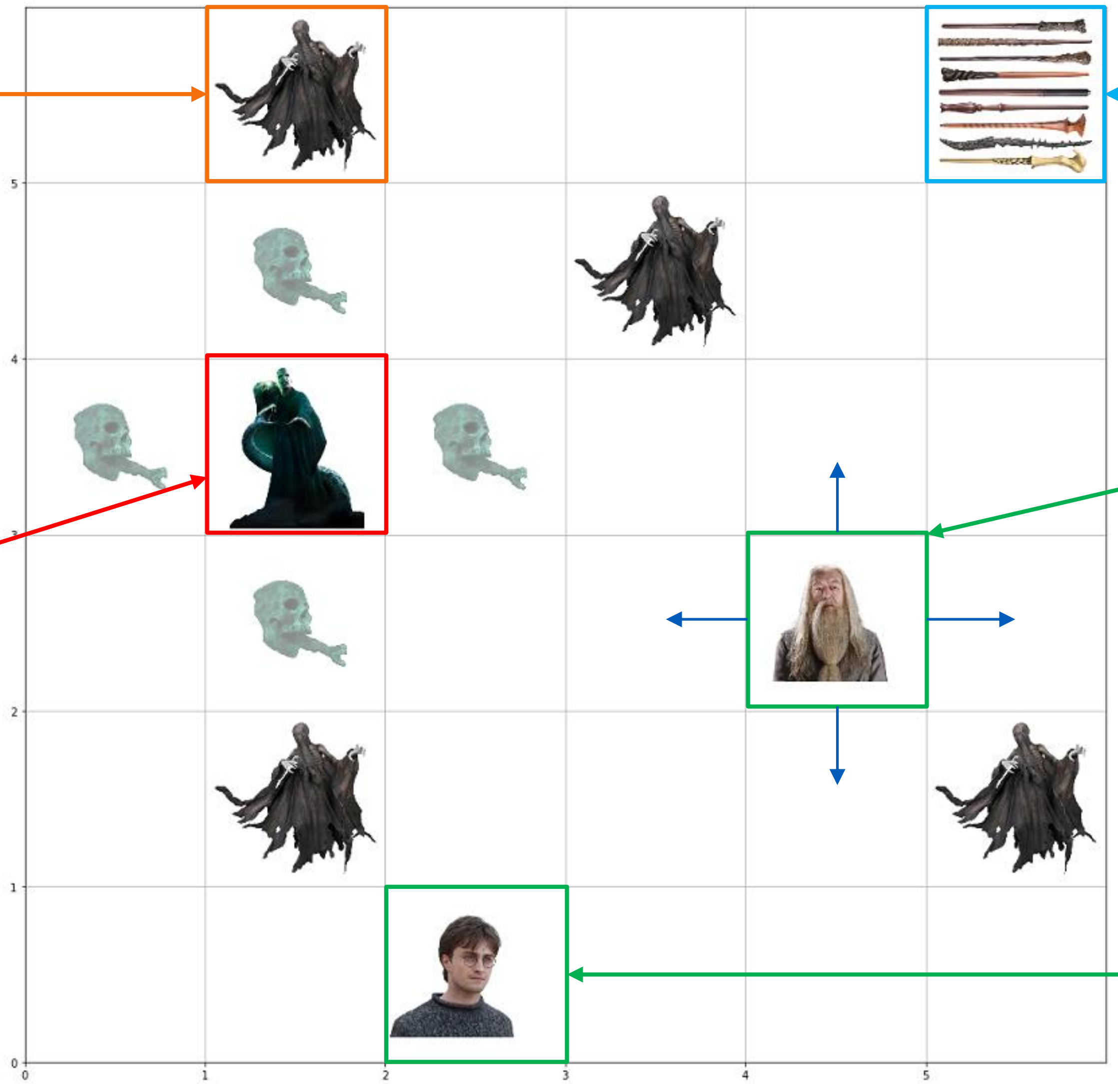
Environment: “Harry Potter in the Grid World”



Reward Dynamics

- Going closer to the wand +1
- Going father from the wand -1
- Picking up the wand +10

Stepping into the Dementor block -10



Dumbledore

Going closer to Voldemort +1
Going farther from Voldemort -1
Killing Voldemort +25
Stepping into the Voldemort block -10

Harry Potter

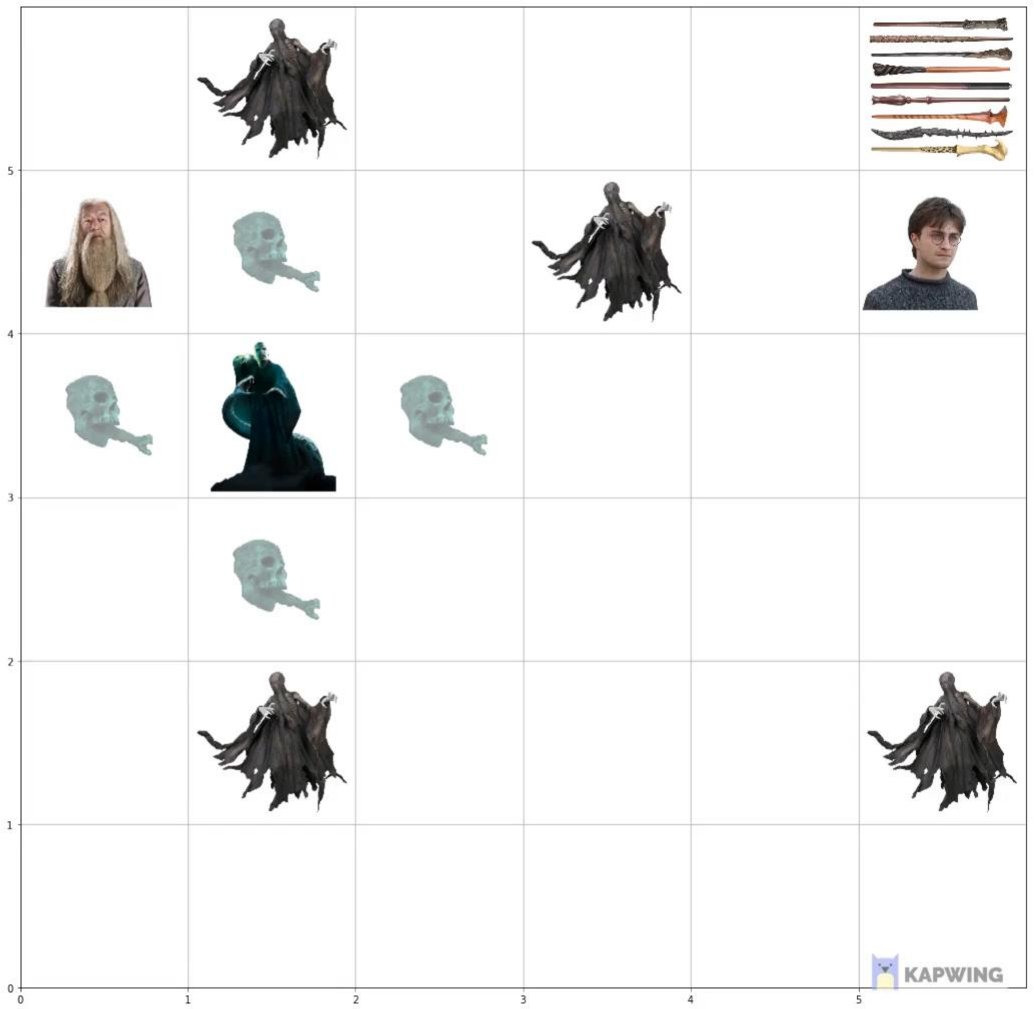
Challenges with Multiple Objectives:

- Environment and training setup.
- A single Q-table / Neural network won’t work.

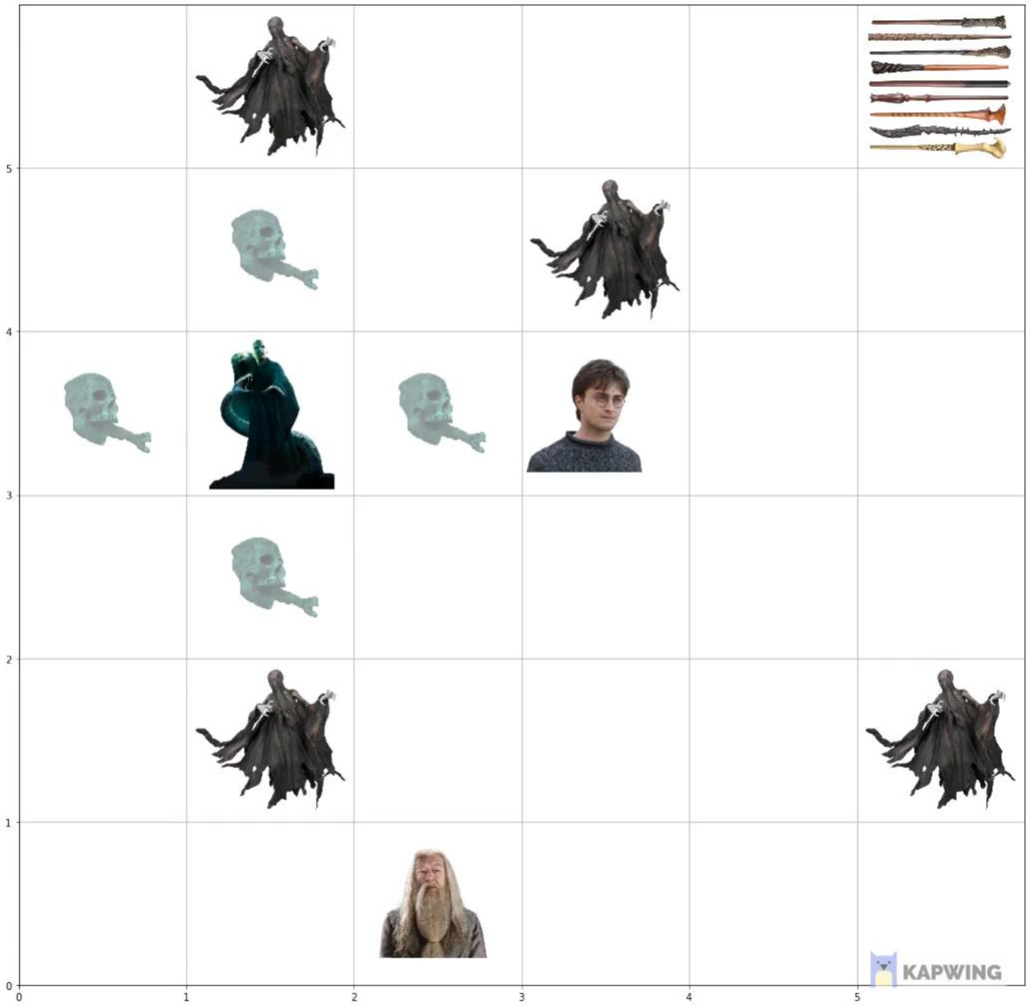
Suggested Solution:

- Ensure that if an objective needs to be completed before another you don’t give the agents the rewards for the second objective until the first is completed.
- Use as many Q-tables / Neural networks as there are objectives.
- While training ensure that each Q-table / Neural network is updated for the appropriate objectives. (Especially challenging with offline RL.)

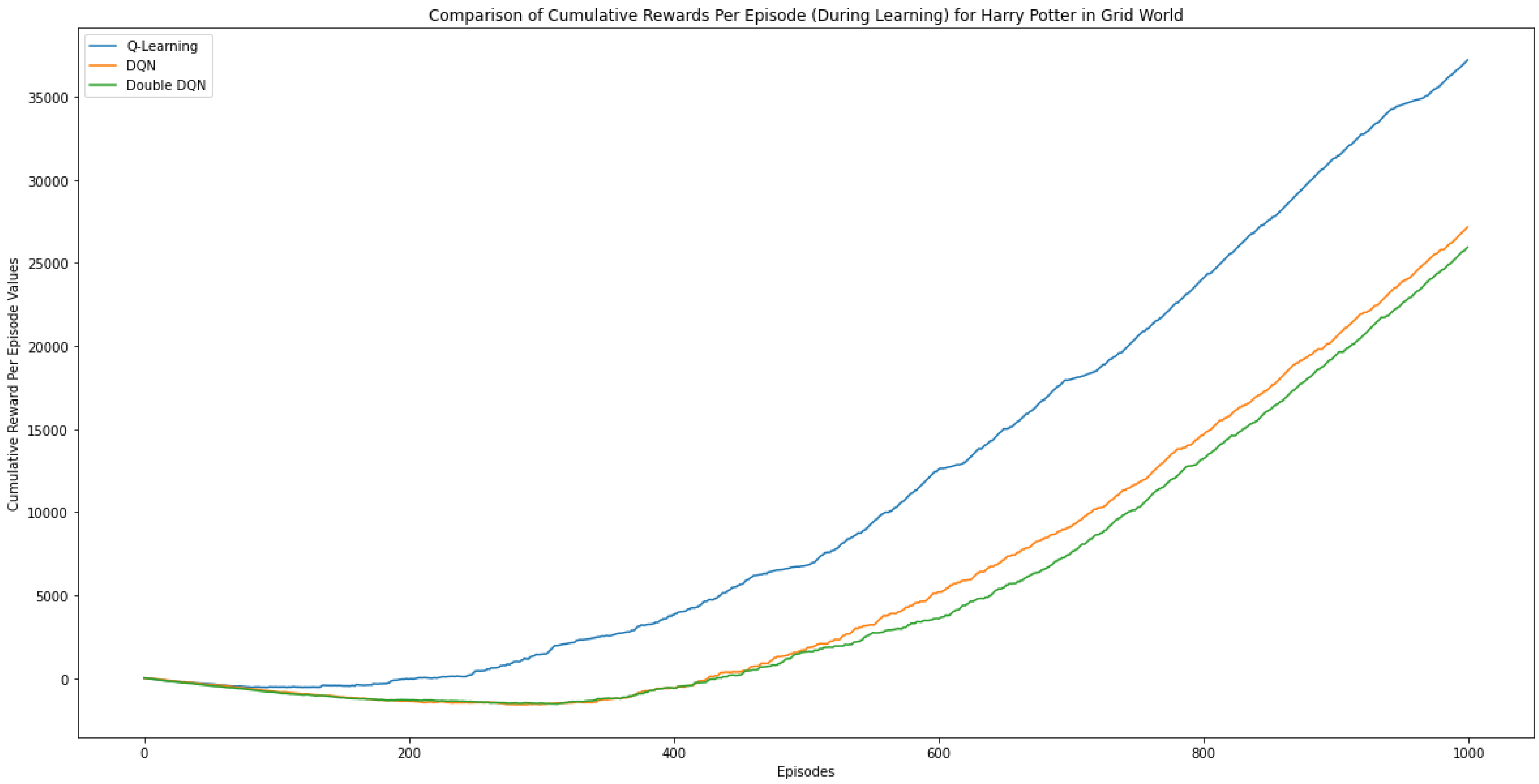
Random Agents



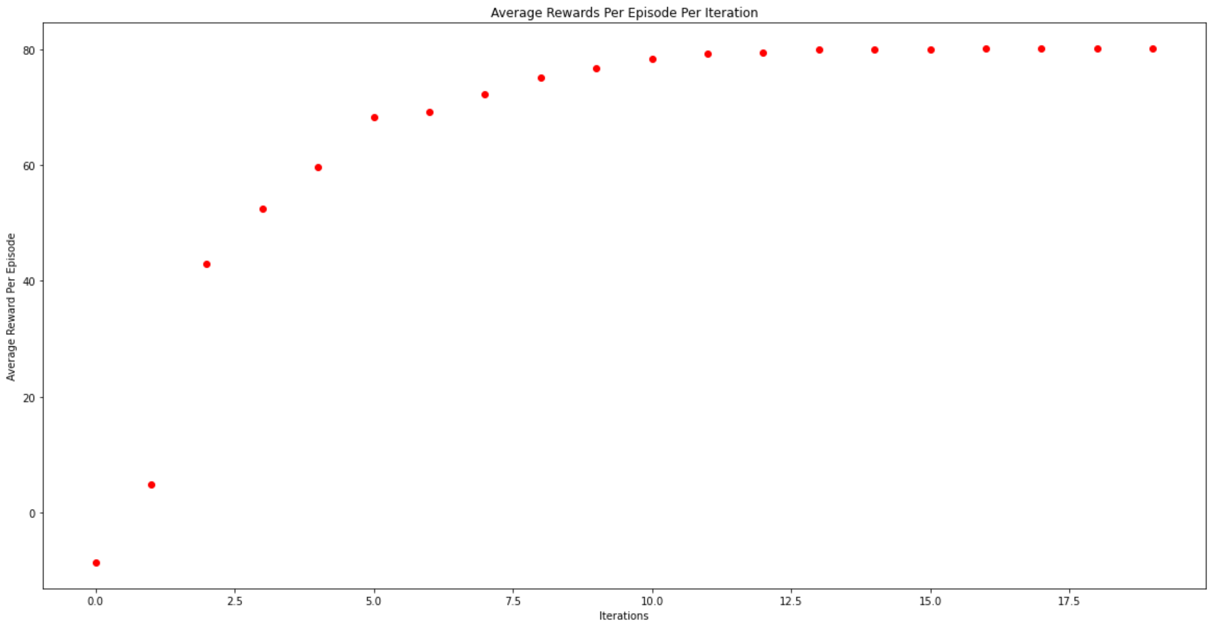
Trained Agents

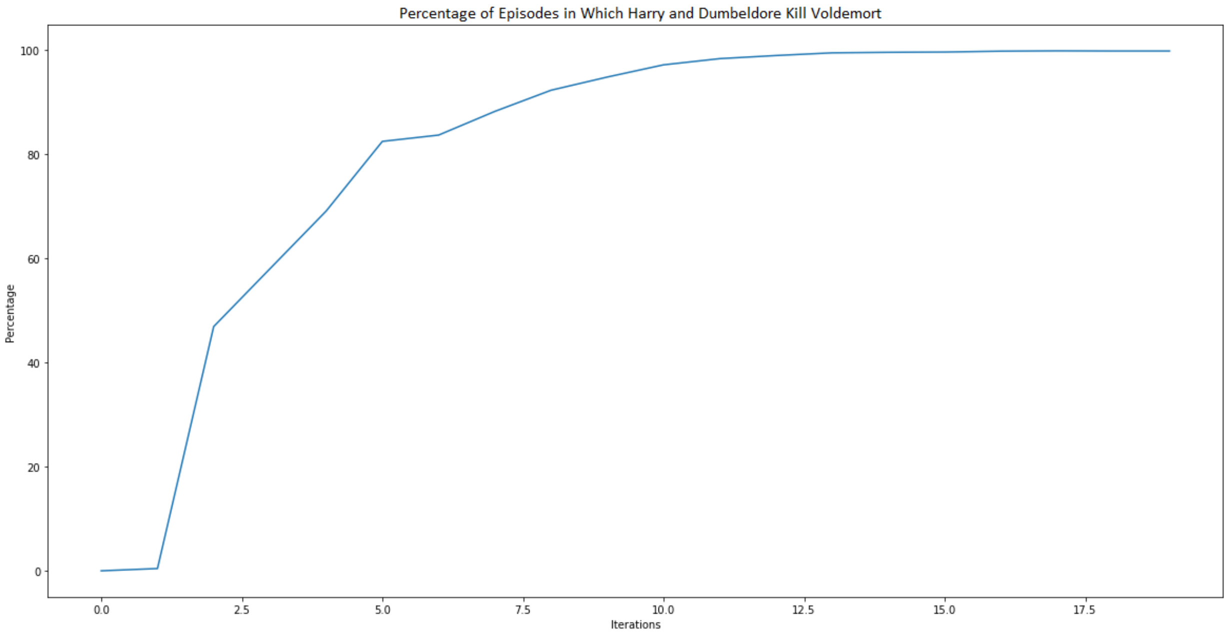


Results

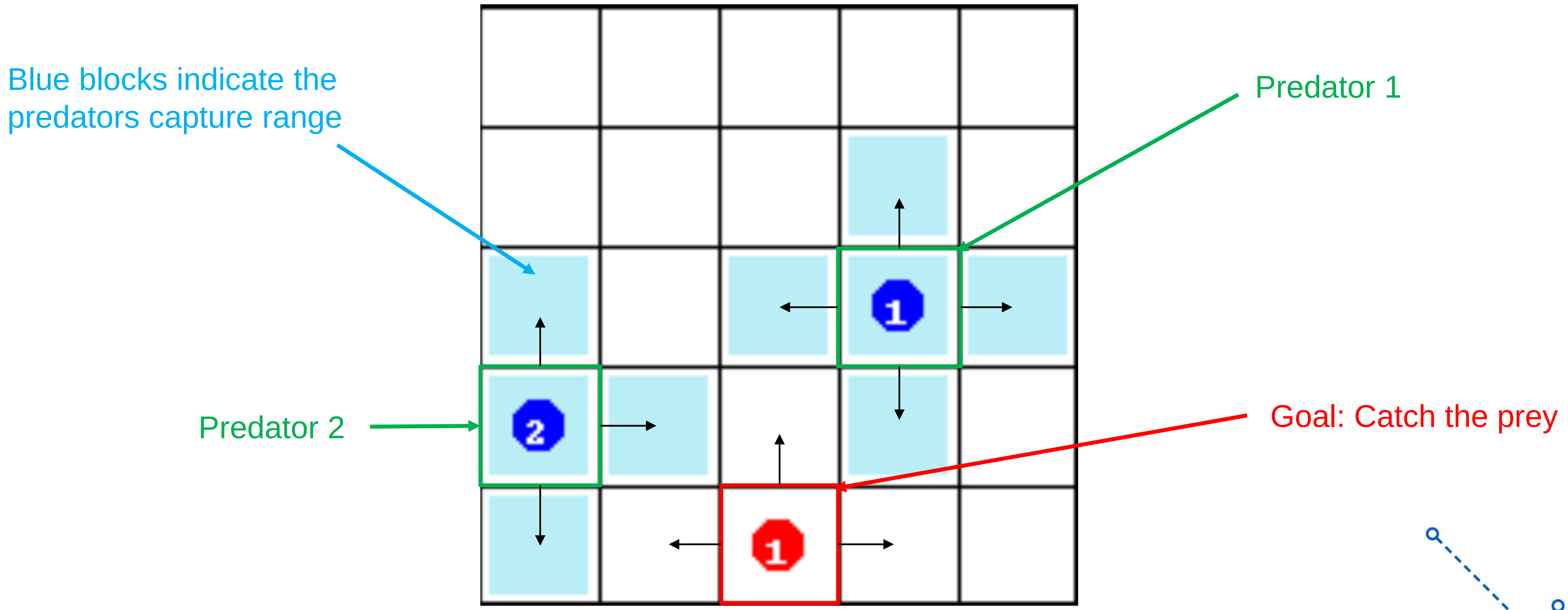

Comparison of Cumulative Rewards Per Episode (During Learning) for Harry Potter in Grid World
Q-Learning
DQN
Double DQN
Cumulative Reward Per Episode Values
35000
30000
25000
20000
15000
10000
5000
0
0
200
400
600
800
1000
Episodes

Results

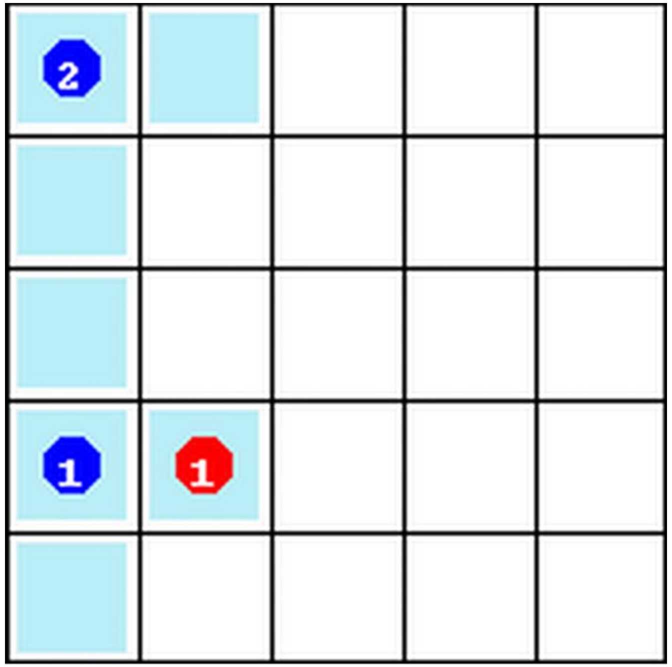

Average Rewards Per Episode Per Iteration
Average Reward Per Episode
Iterations
80
60
40
20
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Percentage of Episodes in Which Harry and Dumbeldore Kill Voldemort
Percentage
Iterations
100
80
60
40
20
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

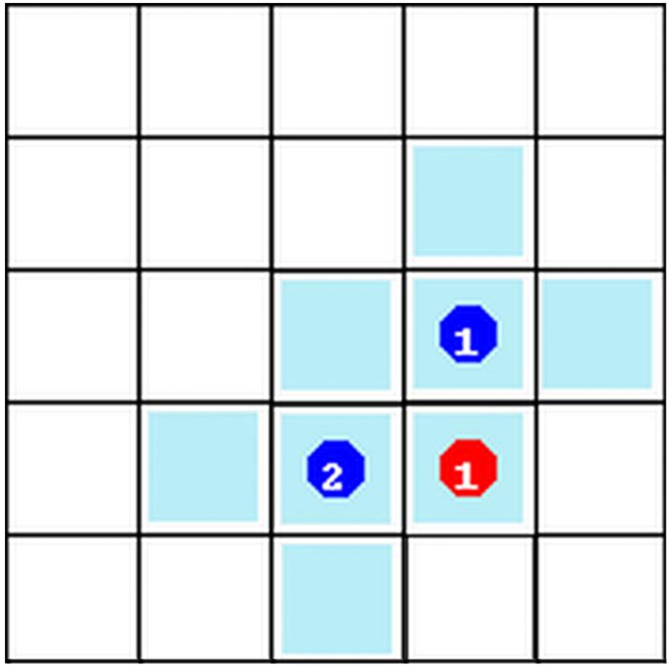
Environment: “Predator-Prey”



Reward Dynamics

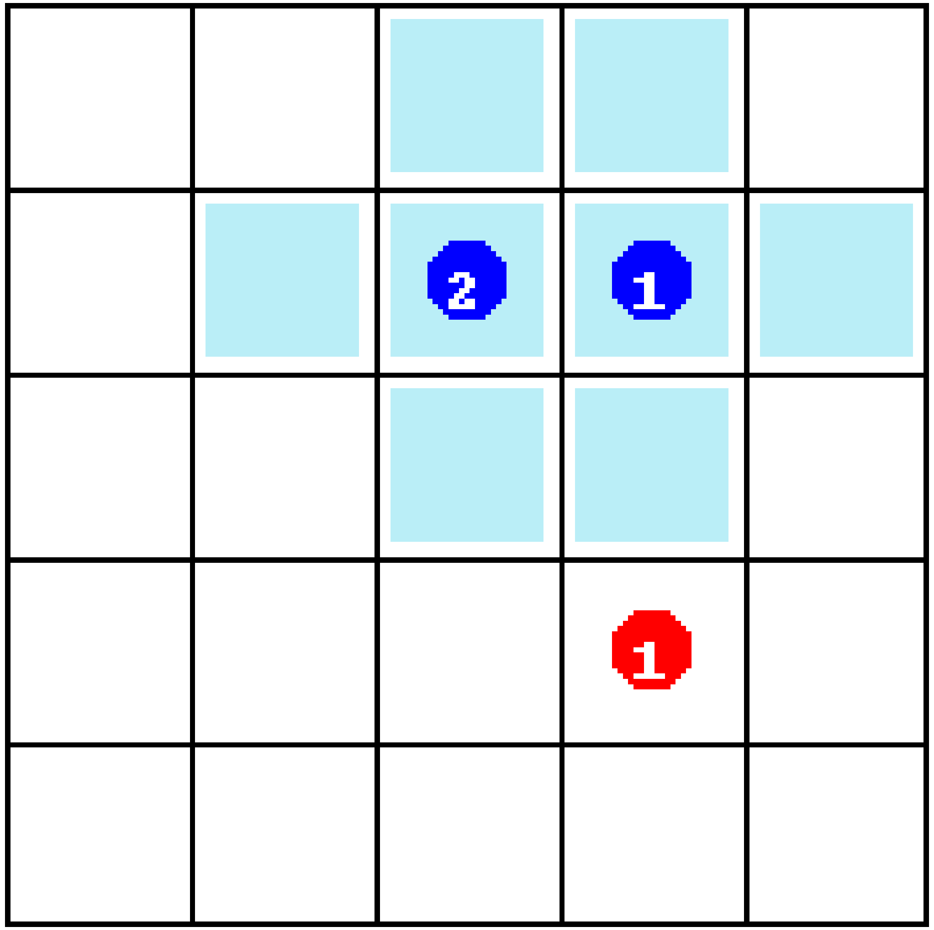


A single predator catches the prey: -0.5

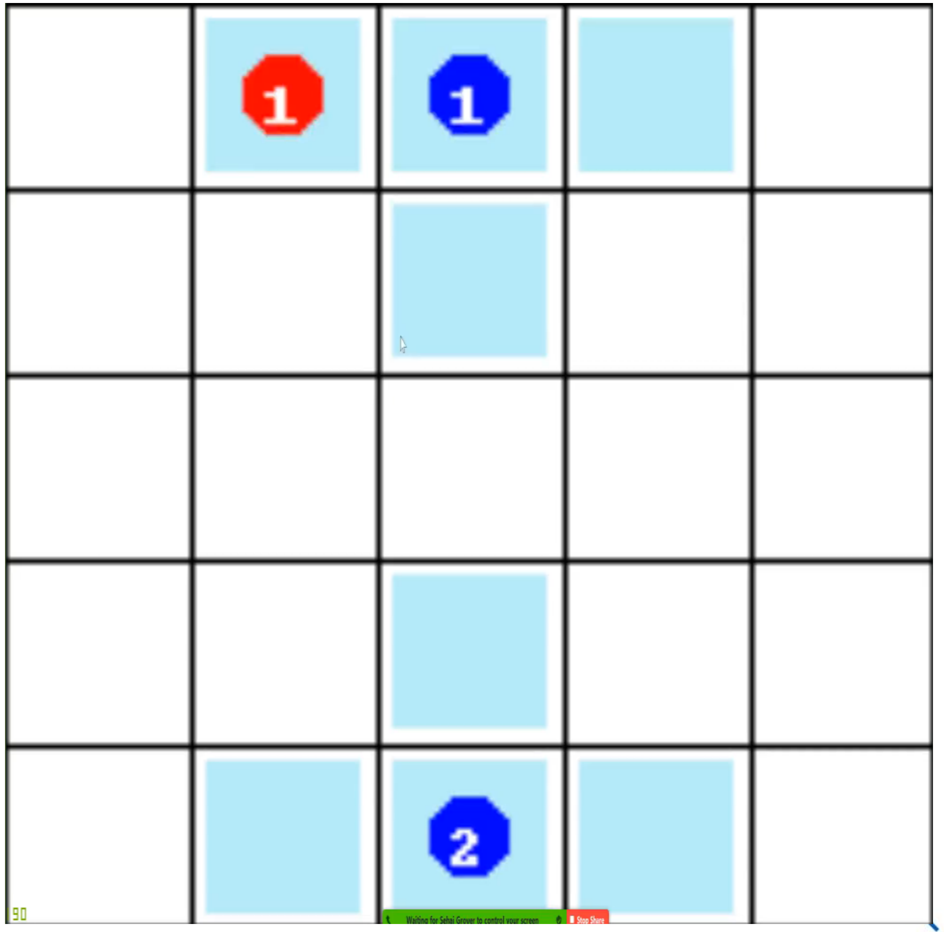


Both predators catch the prey: +5

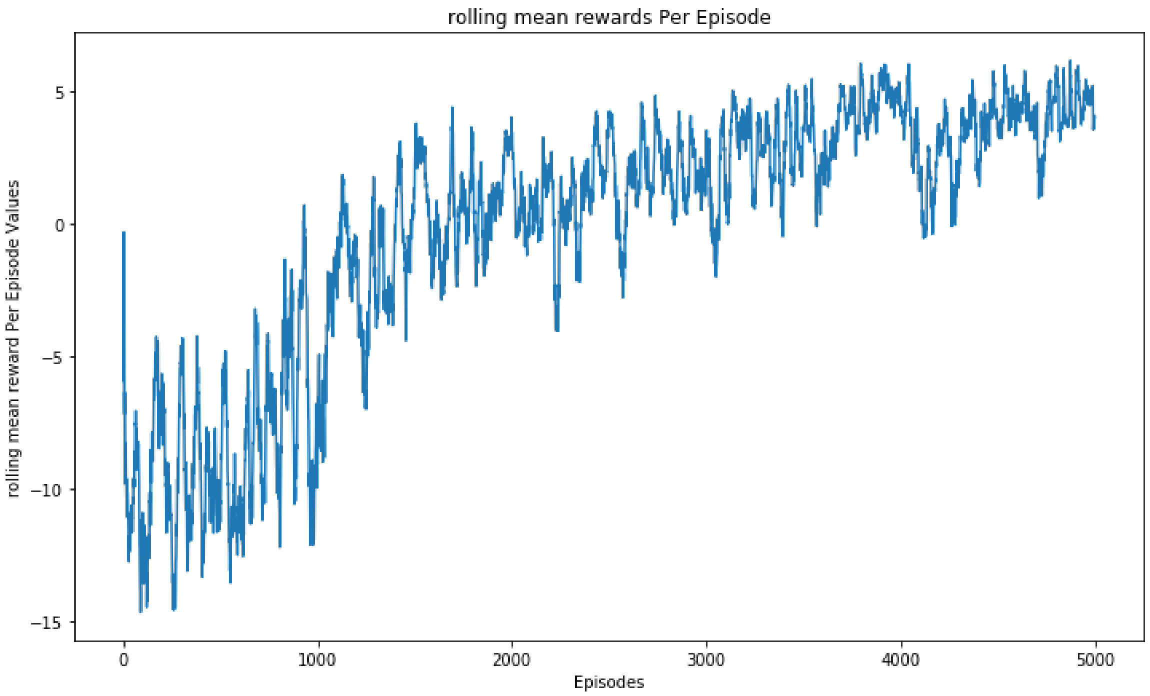
Random Agents

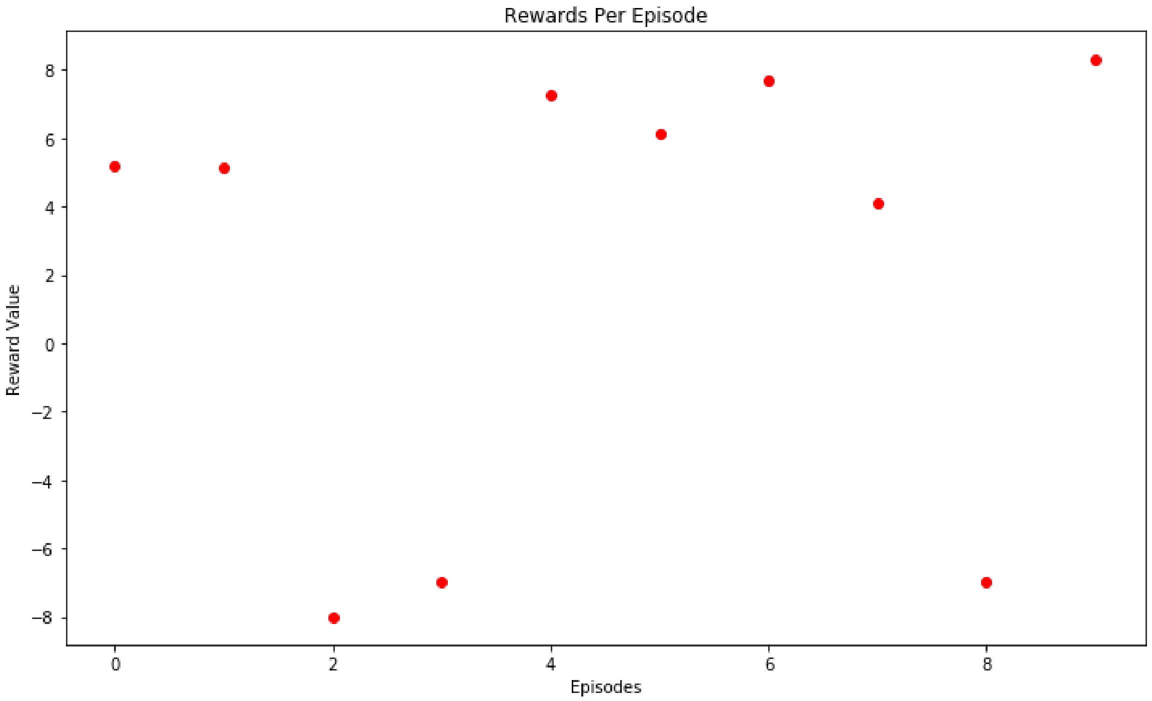


Trained Agents



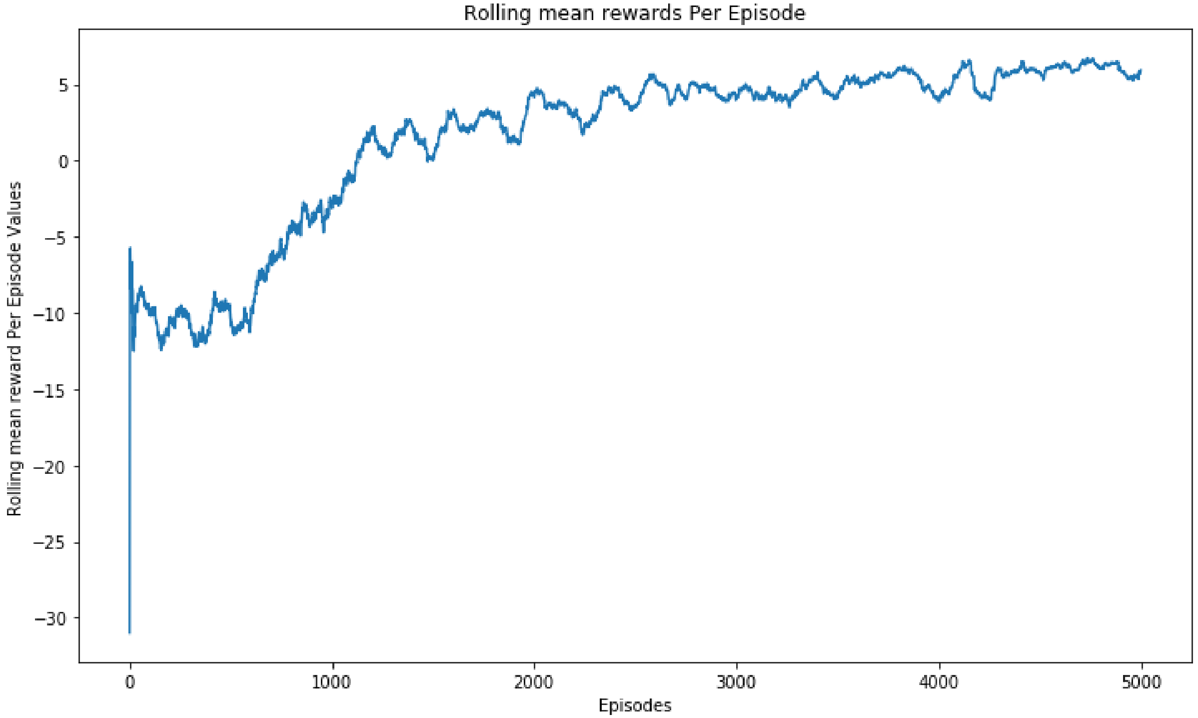
Results:

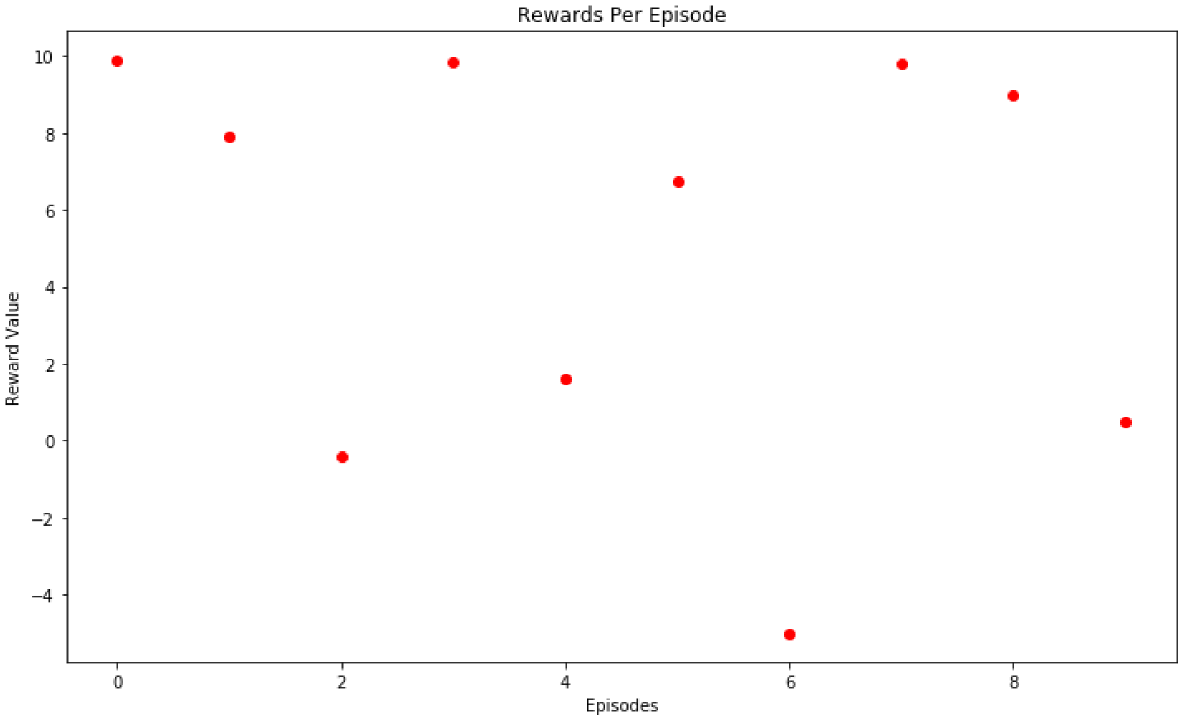




Percentage of episodes in which the Predators catch the Prey: 70.0 %

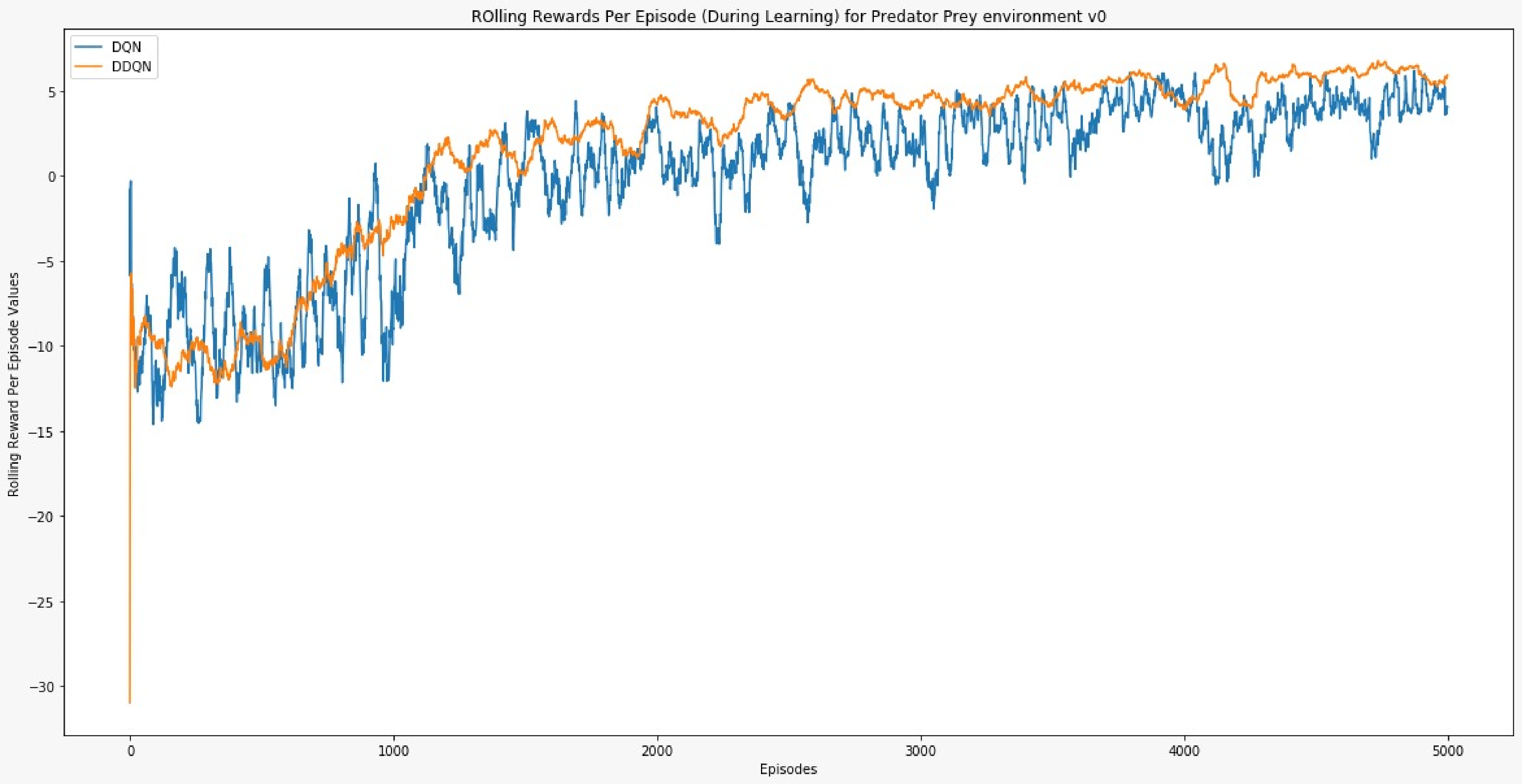
Results:



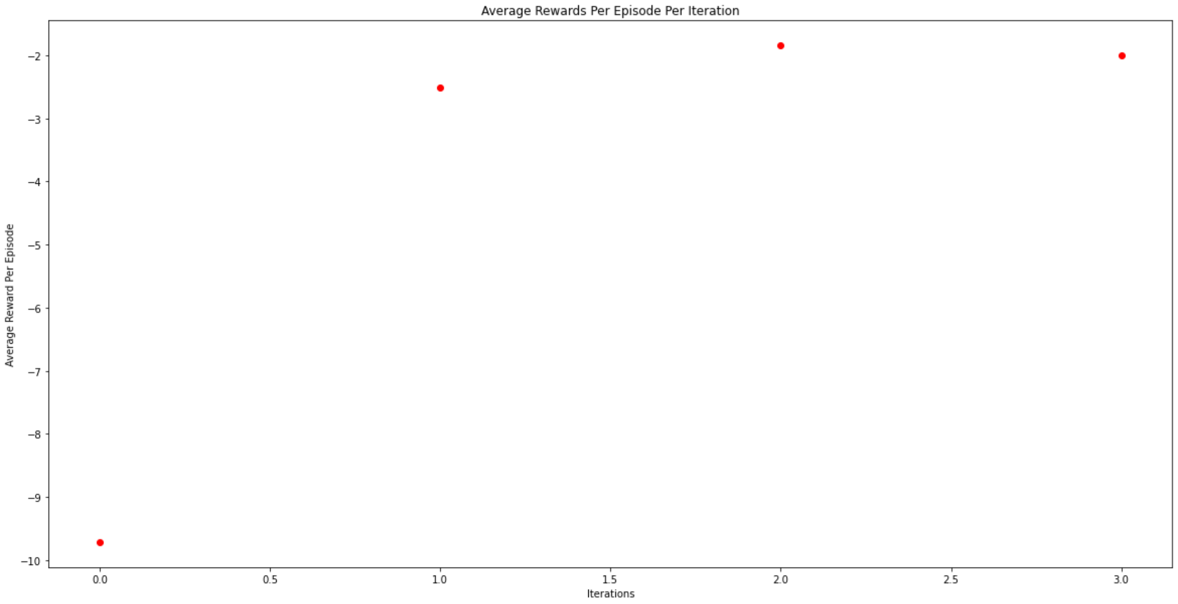


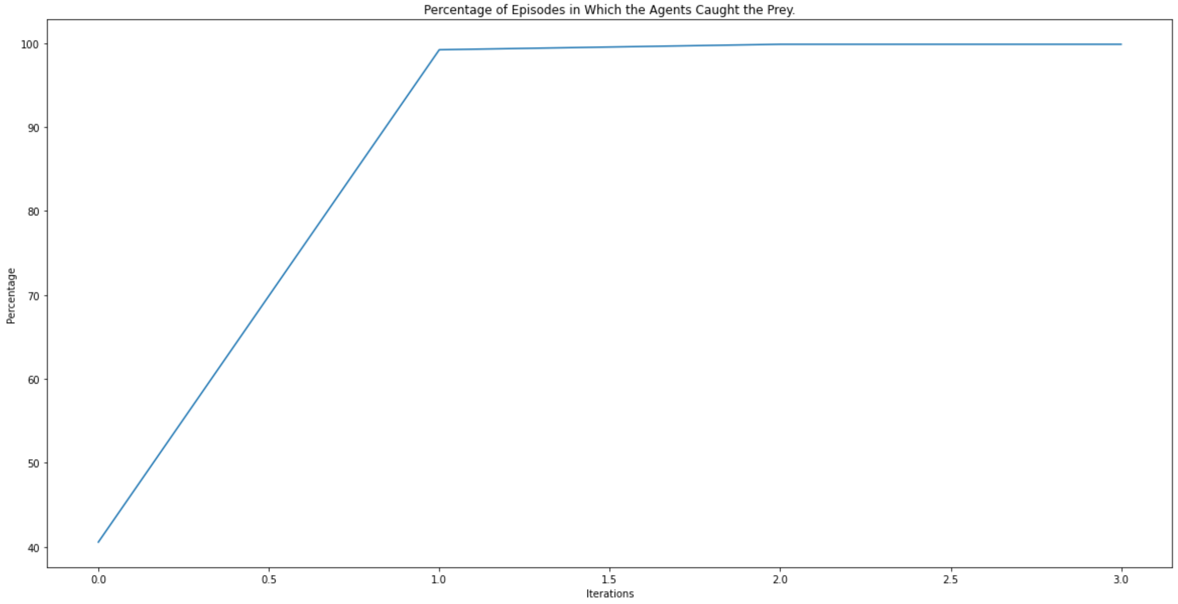
Percentage of episodes in which the Predators catch the Prey: 90.0 %

Results


ROlling Rewards Per Episode (During Learning) for Predator Prey environment v0
DQN
DDQN
Rolling Reward Per Episode Values
Episodes
5
0
−5
−10
−15
−20
−25
−30
0
1000
2000
3000
4000
5000

Results:





Percentage of episodes in which the Predators catch the Prey: 98.0 %

Key Observations / Summary

- The key feature for the MARL is for the agents to collaborate to achieve the goal.
  - In the “Harry Potter in Grid World” environment both Harry and Dumbeldore have to attack Voldemort at the same to defeat him as individually they aren’t strong enough to defeat Voldemort.
- To solve multiple objectives in the environment, we must implement different value approximation functions for each objective.
  - When implemented using a single value approximation function, the learning isn’t correct because depending upon the current objectives a different action must be performed in the same state.
  - When implemented using a different approximation function per objective, the agents are able to learn the optimal policy.

THANK YOU!