CS698R: Deep Reinforcement Learning Project

Final Project Report

Team Name: Runtime Error
Project Title: Foraging in a Field
Project #: 3
Team Member Names: Roll No
Dibyojyoti Sinha: 180244
Milind Nakul: 180420
Prakhar Pandey: 180527
Samarth Varma: 180655

1 Introduction

Foraging has been an essential activity for survival in the wild. In our case, the environment contains patches of berries that are non-renewable, the agent loses reward for every movement and dies after a certain amount of time. The goal of our agent is to find an optimum foraging policy that maximizes the reward in a static environment where the resources are limited and there exists a moving cost but no cost when the agent is at rest. The agent can choose to leave the current patch of berries to explore for more patches or stop moving and conserve its rewards collected so far. This setting can be compared with the consumption of rare natural non-renewable resources, using these do generate wealth but also requires an effort to search and extract.

2 Related Work

Kolling and Akam [2017] explores model-based average reward reinforcement learning to provide a common framework for understanding different foraging strategies. The Marginal Value Theorem (Charnov [1976]) predicts the optimal patch leaving strategy when the rewards within the patch decrease monotonically. However, MVT makes sub-optimal decisions when the rewards may be stochastic or when the current reward is not a good estimator of the future rewards. Moreover, MVT does not account for the cost of searching for new patches and the changing trade-off landscape between spending time searching for a new patch and getting smaller but positive rewards by consuming the current patch. The latter makes more sense when the agent is aware that the episode is about to end.
The agent is required to navigate the environment, and thus it is essential to remember the positions of the already explored regions and patches. Ramezani and Lee [2018] explore a memory-based reinforcement learning algorithm to autonomously explore in an unknown environment. One such memory-based algorithm is Episodic Memory DQN (EMDQN) proposed by Lin et al. [2018]. citedrqn papers talks about the implementation of working memory which would help reinforcement learning agents to solve their environments and gain greater rewards. They propose a change in the architecture of the DQN algorithm and introduce a LSTM layer at the end which acts as a working memory layer for this agent.

In Volodymyr Mnih [2013], the authors have used the Deep Q Network algorithm to train an agent to play several Atari 2600 games. It uses a Convolutional Neural Network along with the Q-Learning algorithm to train the agent. Hado van Hasselt and Silver [2016] uses the Double Deep Q Learning algorithm to overcome the maximization bias of the traditional Q-Learning algorithm and trained an agent to play Atari games. Ziyu Wang [2016] introduced a new neural network architecture for model-free reinforcement learning. It uses two estimators one for the state value function and one for the state-dependent action advantage function. The dueling network automatically produces separate estimates of the state value function and state-advantage function, without any extra supervision.

Christopher Stanton [2018] paper explores the complexities of the sparse reward space and how to tackle the challenges put forward by it. The authors talk about dividing the environment into smaller grids and giving the agent a curiosity reward for exploring each new grid. This helps the agent in exploring more of the environment and this prevents it from getting stuck at a local optimum.

3 Problem Statement

As described above, our goal is to find an optimal solution for collecting berries from a field. The field has berries spread in randomly distributed patches. The agent's goal is to collect the maximum amount of juice (which will be its reward) in a given time frame by moving in 8 directions (N, NE, E, SE, S, SW, W, NW). The agent can increase the amount of juice by collecting more berries, and it will lose the juice while moving around in the field (at a constant rate). Depending on the size, there are four kinds of berries in the field. The bigger the berry, the more is the amount of juice the agent will get by collecting it (linear relation between berry size and the amount of juice it contains).

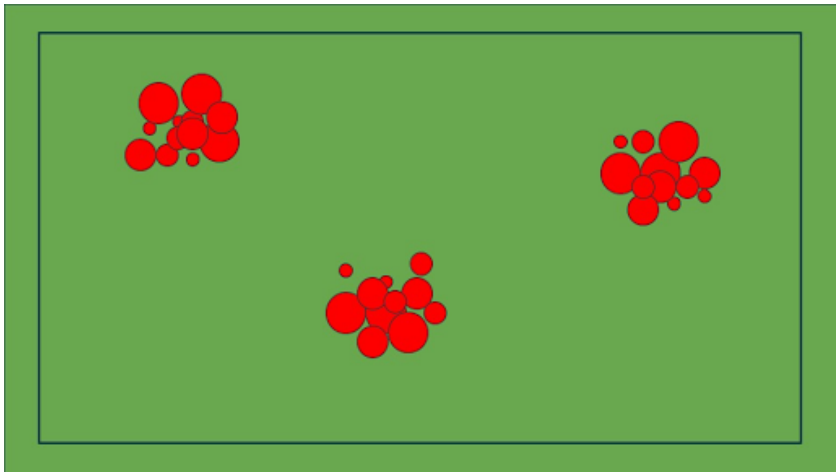


Figure 1: Visualization of the Problem Description

4 Environment Details and Implementation

Following are the technical details of the environment:

- Field size: (20000, 20000)
- Patch size: (2600, 2600)
- Berry sizes: (10, 10), (20, 20), (30, 30) and (40, 40)
- Number of patches: 10
- Number of berries: 20 berries of each size in each patch ,i.e. Total 800 berries, 200 berries of each size
- Total time: 5 min = 300 sec
- Agent speed: 400 pixels/sec, Agent Size : (10,10)
- Max steps: (Agent speed)*(Total time) = 120000
- Action space: (No move, N, NE, E, SE, S, SW, W, NW)
- Area visible to the agent: (1920, 1080)
- Drain rate: $\frac{0.5d}{120*400}$, where 'd' is the distance moved in pixels
- Reward factor: $\frac{s}{1000}$, where 's' is the size of the berry collected, initial juice : 0.5

Table 3 describes the MDP of the environment:

State space (S)	Set of all the matrices of size 35*5
Initial states (S_θ)	Singleton set (Initial state is fixed)
Action Space (A)	Set of size 9 where action 0 corresponds to no-move and actions 1 to 8 correspond to a move in one of the 8 directions
Transition function (T)	The probability of landing in a particular state (s') when the agent is in state (s) and takes action (a) is either 1 (for one state) or 0 (for all other states)
Reward (R)	$-1 * drain\_rate + \sum_{collectedberries} berry\_size * reward\_factor$
Horizon (H)	Finite Horizon (120000) - Agent is aware that the task will terminate in finite time steps

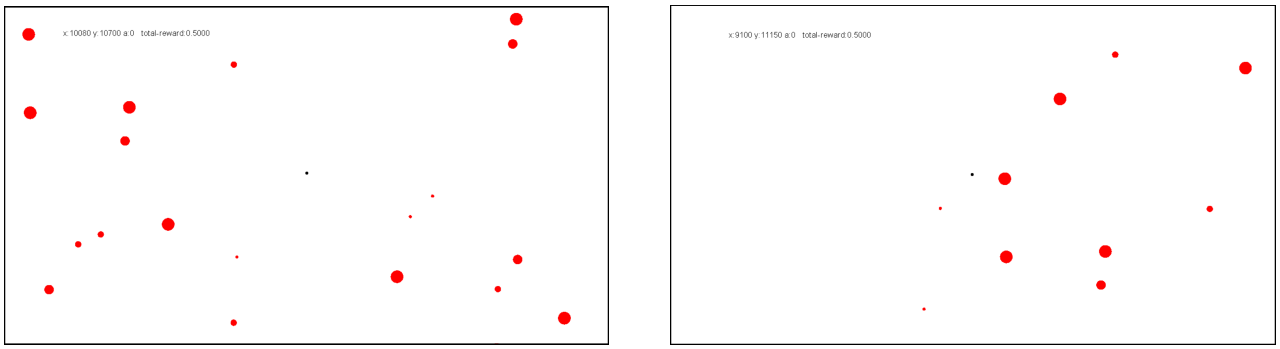
Table 1: MDP for the environment

The structure of the observation received from the environment is as follows:

- Matrix of size 35*5 having the following structure

Boolean value specifying if the information stored in this row is of a berry	X component of the unit vector pointing towards the berry	Y component of the unit vector pointing towards the berry	Distance from the agent to the berry	Berry size

Table 2: Observation structure 1



(a) (b)

Figure 2: Rendered Images of the Environment

The above images show the implemented environment for two different locations. The berries and the agent are both implemented as squares.

5 Proposed Solution

The first solution proposed to solve the research problem is the Deep Q Networks. Since the problem we are trying to solve is very similar in nature to the Atari games, the Deep Q Networks should be able to solve our environment. To tackle the computational complexity of the DQN algorithm, we propose a slight change in the architecture of the neural network. Instead of using Convolutional Neural Networks, we use the following network architecture-

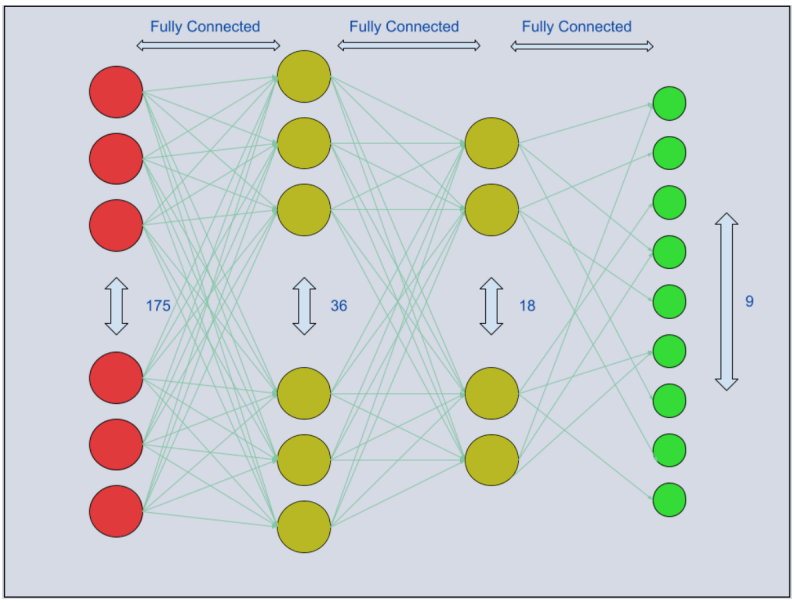


Figure 3: Neural Network Architecture

The Neural Network Architecture consists 175 input nodes. The step function of our environment returns a 35*5 numpy array having the following information:

1. Boolean (0/1) : Specifying if the info stored in this row corresponds to a berry
2. Float : The X and Y coordinate of the berry present in the 1920*1080 pixel window visible to the agent
3. Float : Specifying the distance from the agent to the berry
4. Float : Specifying the size of the berry

These values are returned corresponding to all the berries visible to the agent. This NumPy array thus helps us reduce the computational complexity since we can replace the Convolutional Neural Network Architecture with a Deep Neural Network.

Another challenge we faced while using the DQN algorithm was that the reward space was very sparse in our case, and random actions generally do not lead to rewards. Hence the agent does not learn much while doing this random exploration. We used the following three strategies to tackle the sparse reward problem:

1. **Heuristic Policy** : We start exploring the environment with a heuristic policy so that the agent is able to explore more of the environment. We divide the 1920*1080 pixel window visible to the agent into 8 sectors corresponding to each action. We compute a discounted average of the berry juice for each sector, discounted by the distance of berry from the agent. We make the agent move towards the maximum amount of average juice. It provides a benchmark for our agent.
2. **Motivation Approach** : In this approach, we augment the sparse reward signal with a dense reward signal to motivate our agent to explore more of the environment. As the agent succeeds in these tasks, it learns about the environment better than learning just from the sparse reward signals. We provide the motivation with the following two sub-approaches:
   - **Closest Berry** : In this approach we find the berry closest that the agent can see. The additional reward is given by :
   $$additional\ reward = e^{-k*distance}, \tag{1}$$
   where k is a hyperparameter and distance is the distance to the closest berry.

- **Curiosity** : In this approach in order to motivate the agent to explore new states and actions we keep a novelty reward. We divide the environment into 100*100 grids. There is a curiosity reward associated with exploring each grid for the very first time.

$$R_c = \beta * R + (1 - \beta) * I_c, \tag{2}$$

Here R_c represents the curiosity reward, R represents the game reward and I_c represents the indicator function denoting whether the current grid has already been explored. β is the hyperparameter which controls the preference between current game reward and the curiosity reward

6 Experiments

We ran the following experiments:

- Deep Q Networks
- DQN with closest berry as motivation
- DQN starting with a Heuristic Policy
- DQN with curiosity approach

7 Results and Analysis

We trained the Vanilla DQN for about 200 episodes. The reward improves after every episode however it does not converge in the given number or episodes. We could not train it for a larger number of episodes due to computational limitations.

In the second plot we see that when the closest berry was provided as motivation to the agent then the reward improves but it is still not able to converge in 200 episodes. This agent performs better than the Vanilla DQN algorithm.

In the plot for DQN agent with heuristic start policy we see that the agent converges to a cumulative reward of about 0.48 in about a 150 episodes. However the problem with this algorithm is that the agent does not explore the environment much and stops after just exploring a single patch of berries.

In order to tackle the problem of the agent not exploring the environment much we introduce the curiosity reward in order to motivate the agent to explore more grids in the environment. However this algorithm is computationally very complex and hence the agent does not learn much in the 200 episodes that it was trained.

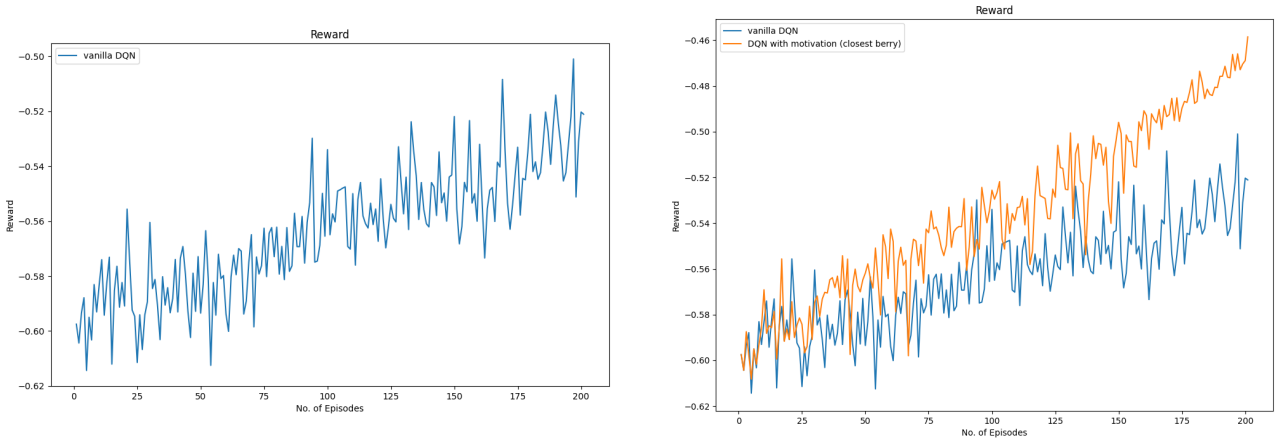


Figure 4: Left: Vanilla DQN, Right: Comparison of Vanilla DQN and Vanilla DQN with Motivation

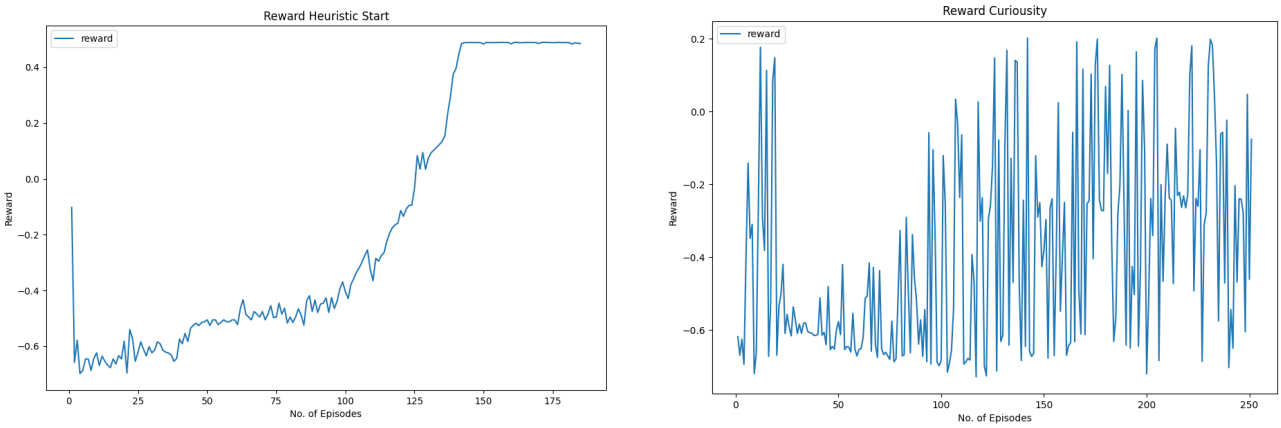


Figure 5: Left: Heuristic Policy as initial policy, Right: Curiosity Search

8 Future Directions and Conclusions

We could incorporate working memory in our agent using a LSTM layer in our Neural Network architecture. This is useful since our research problem is a partially observable Markov Decision Process.

Starting with Heuristic Policy leads the agent to a local optimum to a cumulative reward of 0.48. However, the agent does not explore anything outside the first patch it starts within.

Our DQN algorithm with the motivational reward also demonstrates a learning curve, which is slow and requires a high amount of computational power to train. Introducing more dense motivational rewards will facilitate a faster learning rate and encourage the agent to explore the environment.

9 Member Contributions

Below are the member contributions till now for the project.

Dibyojyoti Sinha	Environment Rendering and Testing, Curiosity Search Implementation ,Environment Optimization, Implementation of Heuristic Agent.
Milind Nakul	Base DQN Agent Implementation, DQN Agent with Heuristic start policy, Curiosity Search and related research work
Prakhar Pandey	Environment Implementation , DQN Class Approach and Motivational Reward
Samarth Varma	Base DQN Agent Implementation, Motivational Reward, Rendering and Environment Testing and Result Analysis

Table 3: Contributions

References

Eric L. Charnov. Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2):129–136, 1976. ISSN 0040-5809. doi: https://doi.org/10.1016/0040-5809(76)90040-X. URL https://www.sciencedirect.com/science/article/pii/004058097690040X.

Jeff Clune Christopher Stanton. Deep curiosity search: Intra-life exploration can improve performance on challenging deep reinforcement learning problems. *arXiv preprint arXiv:1806.00553*, 2018.

Arthur Guez Hado van Hasselt and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence, 30(1)*, 2016.

Nils Kolling and Thomas Akam. (reinforcement?) learning to forage optimally. *Current Opinion in Neurobiology*, 46:162–169, 2017. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2017.08.008. URL https://www.sciencedirect.com/science/article/pii/S0959438817301393. Computational Neuroscience.

Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks. *arXiv preprint arXiv:1805.07603*, 2018.

Amir Ramezani and Deokjin Lee. Memory-based reinforcement learning algorithm for autonomous exploration in unknown environment. *International Journal of Advanced Robotic Systems*, 15:172988141877584, 05 2018. doi: 10.1177/1729881418775849.

David Silver Alex Graves Ioannis Antonoglou Daan Wierstra Martin Riedmiller Volodymyr Mnih, Koray Kavukcuoglu. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Matteo Hessel Hado van Hasselt Marc Lanctot Nando de Freitas Ziyu Wang, Tom Schaul. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2016.

A Appendix

The various hyperparameters used during the experiments and their specifications are as follows:

- γ : The value of the discount factor which was used is 0.999
- ϵ : The starting value of the ϵ was 0.9 and it was decayed to 0.05 over the various episodes. This worked quite well in our case.
- Batch Size : We have experimented with two different batch sizes, 64 and 128. Both provided us with very similar results. However, batch size of 256 had diverging results.
- Episodes : We have trained our models over about 250 episodes. The training time for there 250 episodes was about 15 hours (RTX 3060 6GB)
- β : We have used the value of 0.25 for this hyperparameter. This is the parameter which decides between the curiosity reward and the game reward which is obtained. The higher values of this parameter signify that the game rewards are given more weightage over the curiosity reward of exploring new grids in the environment.
- No. of grids : For the curiosity reward we have used grids of size 100*100.
- Update Frequency : For the DQN algorithm we have experimented with three values of the update frequency, 2, 10 and 25. All of these provide very similar results in our case.

B Experimental Details

C Extra Plots

D More Information, etc.