Creating a Curiosity-Driven Artificial Agent with the Ability to Replicate the Exploratory Behaviors of Primates in a Dynamic Reward Environment

Hanseul Kim

Sungkyunkwan University

Department of Biomedical Engineering

Creating a Curiosity-Driven Artificial Agent with the Ability to Replicate the Exploratory Behaviors of Primates in a Dynamic Reward Environment

이 논문을 글로벌바이오메디컬공학 학사 학위논문으로 제출합니다.

2023 년 12 월

성균관대학교 성균융합원 글로벌바이오메디컬공학과

성명: 김 한 슬 2018311299

이 논문을 위 학생의 학사학위 논문으로 인정함.

지도교수 유 승 범 (인)

Creating a Curiosity-Driven Artificial Agent with the Ability to Replicate the Exploratory Behaviors of Primates in a Dynamic Reward Environment

Abstract

주제어 : Reinforcement Learning , Representational Similarity Analysis, Exploration/Exploitation,

Dynamic Environment, Curiosity

Contents

<Table of contents>

제 1장 Introduction

1. Introduction

1.1 Background

Recent advancements in Reinforcement Learning(RL) especially in Deep Reinforcement Learning made it possible for the development of super-human-performing artificial agents such as Deep Blue(1997), and AlphaGo(2016) on complex games such as chess and Go. (Silver et al. 2018) The current implementation of reinforcement learning has limitations of the high cost of sample efficiency(Yu 2018)and low generalizability across different environments(Packer et al. 2018)and even in a single environment with changing dynamics. (Padakandla 2020) These limitations lead to behavioral differences between natural intelligence and artificial intelligence agents which make it difficult to develop human-like artificial intelligence agents in dynamical enviornments.

This problem of sample efficiency and low generalizability of RL agents is shown predominantly as limited adaptive behavior of artificial intelligence agents in dynamical environments which includes changing environment state or perturbations such as wind or lighting. (Nagabandi et al. 2018). Current research on reinforcement learning in computer science tried to solve this problem using model-based RL (Kaiser et al. 2019), curiosity-based RL(Pathak et al. 2017), and successor RL(Kulkarni et al. 2016)which showed better sample efficiency and generalizability. However, there has been limited studies comparing these agents to natural intelligent agent such as primates or human.

Although it is possible to handcraft an RL agent with designer designer-selected metric to optimize which leads to an optimal agent that maximizes cumulative rewards of an environment, it is a different story to build an agent that can show adaptive behavior that is alike a natural intelligence agent. In this research by using primates navigational path data and comparison metrics between natural intelligence and artificial intelligence behaviors, tried to find out which method of reinforcement learning is needed to show the most similar explore/exploit behavior like natural intelligence.

Since there has been a limited study that compares behavior data between natural and artificial intelligence agents, this research will be a stepping stone to building artificial agents that can show adaptive behavior of humans. Also, this research will give insight

into computational and algorithmic components of natural intelligence by comparing multiple implementations of reinforcement learning.

1.2 Research Goal

The goal of this research is to build a reinforcement learning agent that can show adaptive behavior like a natural intelligence agent in an environment with changing reward dynamics. This research will compare multiple reinforcement learning models such as Model-free Q learning, Model-based Q learning, and Curiosity-based reinforcement learning.

This research will investigate the effects of each algorithm showing optimal behavior on two different complexity of the environment. The simple form is a 3 by 3 grid maze with a single fixed goal location and the most complex one is a 2D grid-maze with multiple cross sections and changing reward dynamics.

제 2장 Prelinimares

1 Preliminares

1.1 Maze Environment with Changing Reward Dynamics

Typical reinforcement learning algorithm considers finite stationary MDP(Markov decision process)(Sutton & Barto, 2018) which formally defined as a tuple $M = <S, A, P, R>$ in finite space, where S are a set of states of the environment, A is set of actions of the agent and $P: S * A * S \rightarrow [0, 1]$ is the transition probability of the environment. Finally, R is a function typically linking state, action to reward given the environment. In this paper, we model the reward function as $R: S*A \rightarrow r$ linking reward to the current state.(Padakandla et al., 2019)

We define a stationary reward environment as a consistence reward function R which doesn't change with respect to the training episode number $i \in N$. And we define a non-stationary reward environment as a family of MDPs parameterized by $\theta \in \Theta$

$M_\theta = <S, A, P, R_\theta>$ where S, A, P are the state, action spaces, and transition probability of the environment and R_θ is a function from before with added parameter θ. We define changing points in episode number at which changes to the reward function are made as $I_{t \geq 1}$ which changes the parameter of the reward function accordingly as t increases. So, the non-stationary dynamics of the reward functions is following.

$$R(s,\ a) = \begin{cases} R_{\theta_0}(s,\ a), & \text{for } i < I_1 \\ R_{\theta_1}(s,\ a), & \text{for } I_1 \leq i < I_2 \\ \vdots & \end{cases}$$

1.2 Deep-Q-Learning with ε-greedy Policy

Given the MDP environment (stationary or not), the task of Q-learning agent is to determine the optimal policy that maximizes the total discounted expected reward by a factor $\gamma (0 < \gamma < 1)$.(Watkins & Dayan, 1992) We accomplished this by training a

Deep-Q-Network(DQN) that estimates this discounted expected reward as Q values. (Mnih et al., 2013)

$$Q^*(s,\ a) = E_{s' \in\ S}\left[R(s',a) + \ \gamma * \max_{a \in A} Q^*(s',a)\right]$$

This Deep Neural Network of DQN(with weight q) can be trained by minimizing a sequence of loss functions that changes for each iteration

$$L_i(\theta_i) = E_{s,\ a \sim p(\bullet)}\left[(y_i\ -\ Q(s,a;\theta_i))^2\right]$$

Where $y_i = E_{s' \in S}[R(s',a) + \gamma * \max_{a'} Q(s',a';\theta_{i-1}) \| s,a]$ is the target Q value for iteration i and $p(\bullet)$ being probability distribution of state, actions pair of agent's past history.

After learning this $Q(s,a)$ distribution, the DQN agent selects action based on ε -greedy policy that chooses greedy strategy $\max_a Q(s,a;\theta)$ with probability (1-ε) and selects random $a \in A$ with probability ε.(Mnih et al., 2013)

1.3. Limitation of Deep-Q-Learning Algorithm with ε-greedy Policy

Since the DQN algorithm is designed to converge in a stationary MDP environment with fixed transition probability and reward function, it shows the following limitation in a dynamic environment with changing rewards.

a. Sample inefficacy while updating changed reward function due to weak inductive bias(Botvinick et al., 2019)

b. Not exploring the environment states when ε gets low enough.

Since the target Q value for iteration i, y_i contains only one argument of the changed reward function $R_{\theta_i}(s',a) \Longrightarrow R_{\theta_{i+1}}(s',a)$ and only one-step next Q value of $\max_{a'} Q(s',a';\theta_{i-1})$ update to the changed reward function is slow and sample inefficient.

Also, because ε-greedy policy controls exploring behavior using ε value alone, when ε values get low enough and the reward function changes DQN agent doesn't explore the state of the changed reward in the environment which leads to the agent getting stuck in non-optimal behavior. This kind of behavior is unlike that of natural intelligence which has shown flexibility in changing explore/exploit decisions in a dynamic

environment.(Addicott et al., 2017; Behrens et al., 2007),(Knox et al., 2011)

1.4 Related Work to Solve Current Limitation of DQN

There are three widely used current solutions to sample inefficacy while updating to the changed reward function of the environment. One of the possible solutions is using batch reinforcement learning which brings the benefits of stability and data-efficiency of the learning process(Lange et al., 2012). Another way is to use internal simulation of the learned environment model called model-based reinforcement learning.(Kaiser et al., 2019) The other solution to this sample inefficacy can be separating the reward function estimate from the neural network Q value. This can be accomplished using Deep Successor Reinforcement Learning(DSR)(Kulkarni et al., 2016). Using the DSR framework, the Q value function at a given state, action can be expressed as a dot product between the vector of discounted future state occupancies and the state-reward vector. The discounted future state occupancies can be estimated using a neural network and the state-reward mapping vector can be updated using the artificial agent's experience.

The solution to the limitation of low explorative behavior rather than adjusting ε value of the random exploration is reward shaping. In reward shaping designer of the artificial intelligence agent adds a new "internal reward" to the agent which is independent of the reward function of the environment agent's training in.(Ladosz et al., 2022) This "internal reward" function can be made with a count-based system which gives larger rewards in low-count states. Another method can be making a predictor system that predicts when given and gives a reward proportional to the error of predicted and actual which leads to the exploring behavior of the uncertain state of the agent.(Ladosz et al., 2022)

However internal reward function of the agent must be designed by the designer to fit the desired behavior. Also due to the count-based and predicted error-based internal reward function having noisy reward due to count number and predicted error dynamically changing along the agent's experience, an under-optimized reward model might lead to noisy reward and in turn lead to insufficient exploration and turns to unstable training loop.(Li et al., 2023) Also balancing two external reward of the environment and internal reward is crucial because this instability of internal reward function might affect neural network which is estimating the cumulated sum of both rewards.

2. Curiosity-Based Agent with Maximum Entropy Update

2.1 Limitation of Current Curiosity-based Reinforcement Learning Models

The current implementation of curiosity-based reinforcement learning models tries to solve problems of sparse external reward, lack of exploring behavior, and low generalizability in unseenenvironments.(Pathak et al. 2017) This is achieved by shaping a reward function such as a counter-based reward which gives a higher reward for unvisited states based on the agent's memory or internal prediction-based reward functions that give a reward based on the error of the agent's prediction which gives higher reward to novel states. (Pathak et al. 2017)

However, there are three kinds of problems in these current curiosity solutions. The first one is dealing with a reward function that is unstable. Since RL is designed to solve a Markovian Decision Process problem(Bellman 1957) in which reward functions are fixed and cumulative reward is mathematically proven to converge, unstable reward functions might nothave converging cumulative reward.

The second problem arises in environments with a dynamical reward function. Since the current implementation of the curiosity model is designed to explore the unvisited sparse states, changes in reward in the visited state cannot be easily explored by these agents.

The final limitation is in deciding the ratio of internal and external rewards. Since the current implementation of curiosity-driven RL optimizes maximizing the linear combination of internal and external rewards, this ratio of the importance of each reward must be crafted and fitted to each environment.

2.2 Curiosity-based Maximum Entropy Reinforcement Learning

We devised a new curiosity-based maximum entropy reinforcement learning model shown in Figure 1. The model consists of two policies which are exploit and explore. Each policy is trained on different reward functions of the environment given external or curiosity-driven internal rewards. The agent uses the exploit policy to choose every online action and the exploit policy is updated using external reward. Then internal

curiosity reward is calculated in offline model-based simulation to update explore policy. Finally, the effect of internal reward to online exploit policy is only possible through maximum entropy update between exploit and explore policies. (update only happens when the reward is unknown)

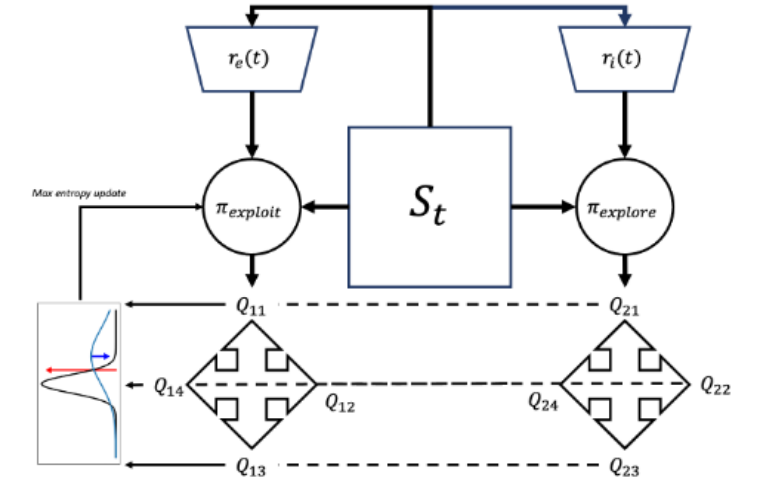

Figure 1 Curiosity-based Maxent RL

This model could solve the problems of the current curiosity-driven model in three ways. One is the buffer effect of explore-policy for unstable internal reward function. Online exploit-policy is not directly affected by internal reward functions, we expect the model to be more stable. Two, exploring past visited states using internal simulation and reward. Since the model uses internal simulation to generate curiosity rewards, the past visited states in the online environment can be revisited and rewarded in the offline internal simulation. Finally, since an update to the online exploit policy is not a linear combination of internal and external rewards, these reward ratios do not need to be fitted to each environment.

제 3장 Experiment Design

1.Task Design

1.1 Maze Enviornment

To compare the behavioral difference in solving the navigational problem of natural intelligence and multiple models of artificial intelligence, we designed both a 3D virtual reality foraging task and a simplified 2D version of the same task to train multiple reinforcement learning agents. The maze layout is shown in Figure 3.1, black lines outline the path of the maze, and Triangles show the location of small rewards which is consistent throughout training and testing. (The colors of the triangle shows different probability of a small reward, Green: a small certain reward, Black and Magenta: a small risky reward. All small reward has the same expected reward of 150uL)

Figure 3.1 Experimental setup for rhesus monkey

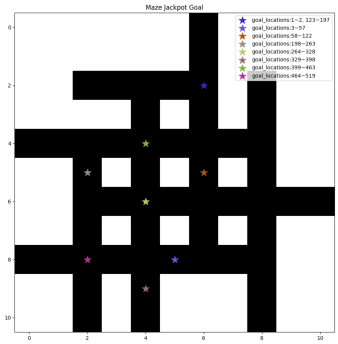

Figure 3.2 Changing jackpot reward location by trials

The maze also consisted of a jackpot reward(1200uL) whose location changes depending on the trial number (different locations selected on different days of rhesus monkey training)

The same task was converted to a simple 2D version for training artificial intelligence agents.The 2D version of the game also had the same fixed small reward location (shown in green in Figure 3.3) with a reward size of 1 and changing jackpot location by trials with a reward size of 8 (shown in red in Figure 3.3). (different reward size but same scale as the 3D version of the game)

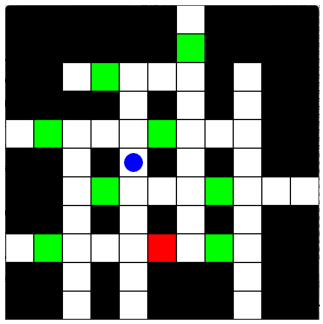

Figure 3.3 2D Version of the game made in pygame

1.2 Training Sequence

Before testing the navigational behavior of both artificial agent and rhesus monkeys, all agent was pre-trained on the environment with no reward to get familiar with the layout of the maze. After the pretraining rhesus monkey was trained with different jackpot locations day by day and this was translated to different jackpot locations based on the same trial number.

1.3 Data Analytic Metric

Data analytic metrics for the artificial agent can be categorized into two parts, one is metrics showing optimality of the agent's navigational behavior shown by total reward, length, and reward rate (total reward/length for each trial). Another metric was designed to calculate the similarity between artificial agents and rhesus monkeys' behavior. This was done by calculating and comparing explore/exploit behavior similarity using the following metrics in Table 1.

Table 1 Explore/Exploit metrics

Explore/Exploit Metric	Metric Formula (per trials)
Map Exploration Percentage	$\frac{count(visited\ \ state)}{count(total\ \ state)}$
Exploration Entropy	$\sum_{s\ \in\ S} -p(s) * \log(p(s))$
Shannon info value of action	$\sum_{s\in S}\sum_{a\in A} -p(a;s) * \log(p(a;s))$

Also, to compare the choices of the rhesus monkey and artificial agent, we converted the behavior data of monkeys in the 3D maze to a state-action pair of the 2D game. After this conversion, every state was inputted to the neural network of agents after each trial to calculate the choice similarity percentage of actions between rhesus monkey and artificial agents.

2. Reinforcement Learning Models Training

Reinforcement learning agents were trained in two environments. One is an environment with a fixed jackpot reward and a fixed starting location for the agent. Another is changing dynamic reward environment with reward locations shown in Figure 3.2 . The total count of each reward distribution is shown below.

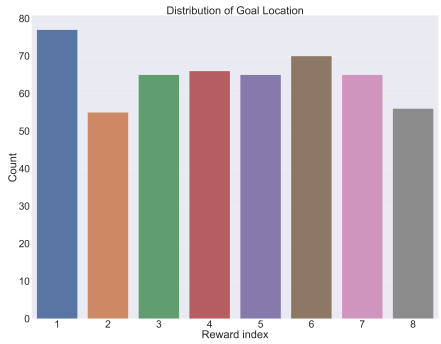

Figure 3.4 Distribution of goal location count

We trained 4 different types of reinforcement learning agents, single-step, batch learning, model-based, and curiosity-based DQN agents. The training number of each reinforcement learning algorithm is shown in the table below.

Table 2 Training number of different RL algorithm

Algorithm name	Fixed Reward Environment	Changing Reward Environment
Single-step DQN	8(1 * 8 reward positions)	0
Batch Learning DQN	8(1 * 8 reward positions)	5
Model-based DQN	X	2 * 6(different simulation num)
Curiosity-based DQN	X	6

제 4장 Results

1. Experimental Results

1.1 Rhesus Monkey Shows Optimizing Behavior in 3D Maze

The optimization behavior of the rhesus monkey by shortening the episodic length in the Vr-maze trial with the first fixed reward position is shown in Figures 4.1.a, 4.1.b. Although difference across categories in first fixed reward position in figure 4.1.b (early-middle p-value = 0.234, middle-late p-value = 0.78, early-late p-value = 0.247), But showed better behavioral difference relative to latter trials figure 4.1.c (early-middle p-value = 0.378 , middle-late p-value = 0.820, early-late p-value = 0.333).

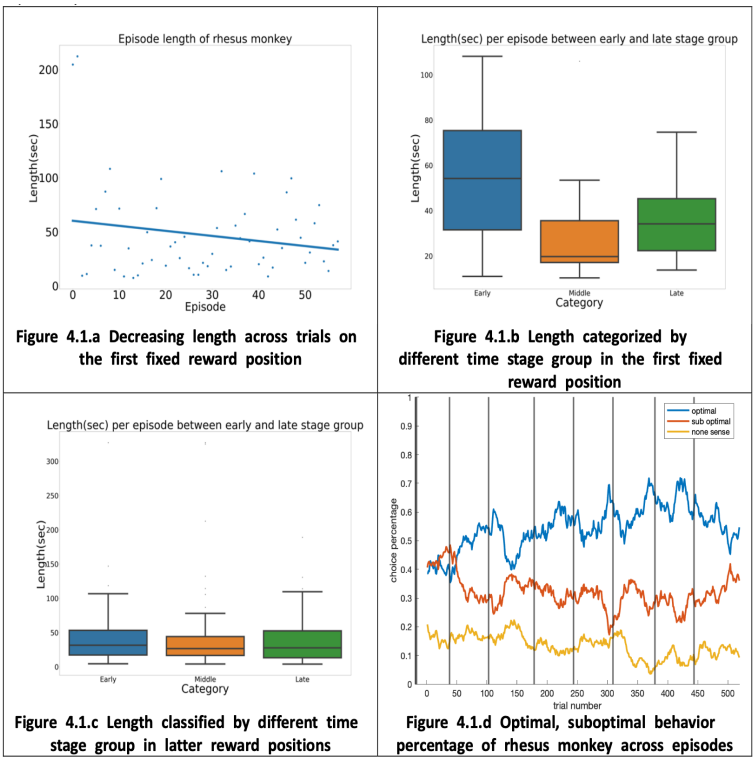

Figure 4.1.a Decreasing length across trials on the first fixed reward position

Figure 4.1.b Length categorized by different time stage group in the first fixed reward position

Figure 4.1.c Length classified by different time stage group in latter reward positions

Figure 4.1.d Optimal, suboptimal behavior percentage of rhesus monkey across episodes

Figure 4.1 Optimizing behaviors of the rhesus monkey (first reward positions)

Also, each directional action was categorized as optimal (getting closer to jackpot-goal and sub-goal), sub optimal(getting closer to jackpot goal but not sub-goal) and other(none-sense). The result in figure 4.1.d showed subject's favoring optimal behavior to sub-optimal and sub-optimal to other random movement.

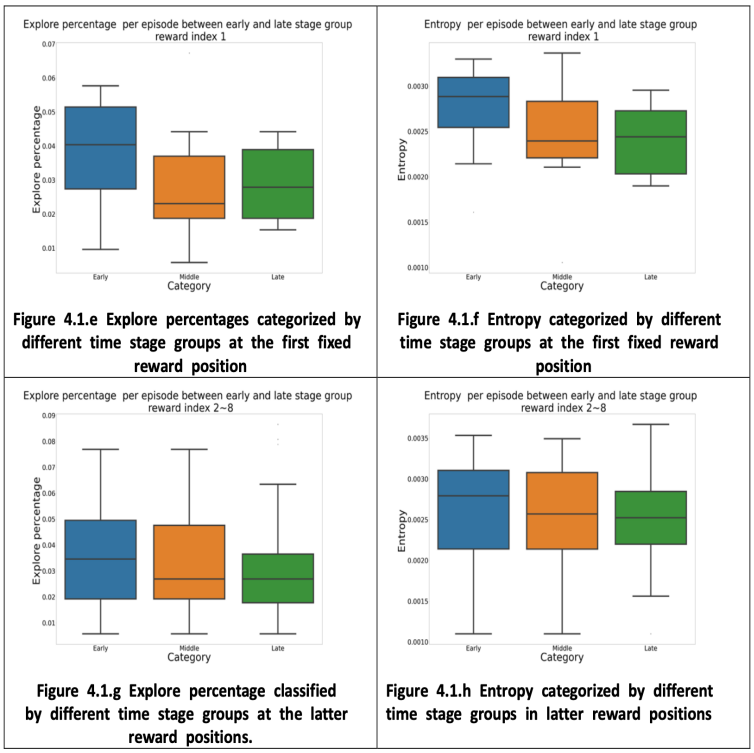

Figure 4.1.e Explore percentages categorized by different time stage groups at the first fixed reward position

Figure 4.1.f Entropy categorized by different time stage groups at the first fixed reward position

Figure 4.1.g Explore percentage classified by different time stage groups at the latter reward positions.

Figure 4.1.h Entropy categorized by different time stage groups in latter reward positions

Figure 4.2 Non-optimal behavior of rhesus monkey on latter reward positions

Similar behavioral metric changes were shown in explorative behavior measured by explore percentage, and entropy of action choices as shown in Figure 4.1.e ~ 4.1.h In the first fixed-reward environment subject showed optimizing behavior by decreasing the explore percentage of map and entropy shown in Figure 4.1.e, 4.1.f, but not in the latter reward trials shown in Figure 4.1.g, 4.1.h.

2. Reinforcement Learning Models Result

2.1 Agent Using the DQN Algorithm Alone Optimizes Well in the Simple Grid Maze but Not in the Complex Maze with Multiple Cross Sections, Unlike Natural Intelligence Agents.

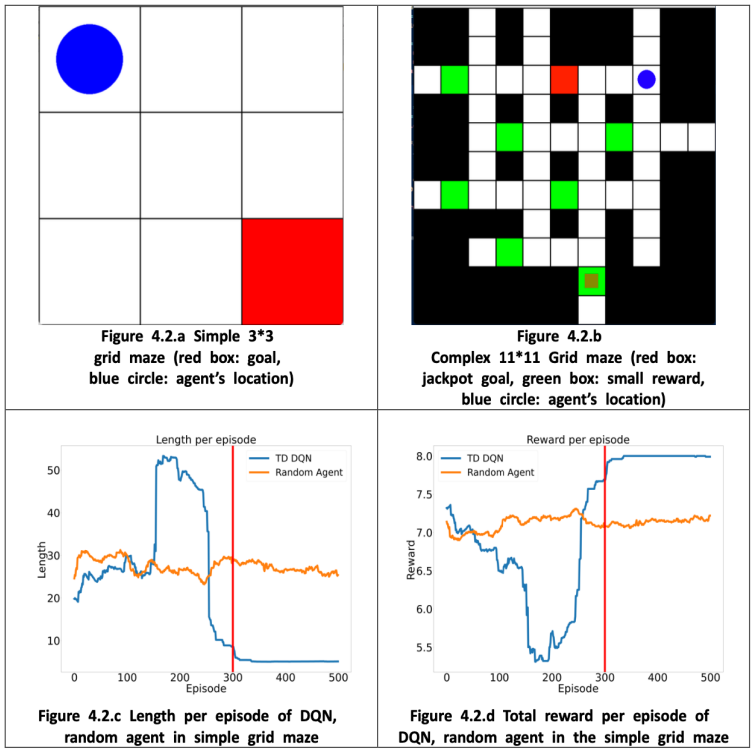

Figure 4.2.a Simple 3*3 grid maze (red box: goal, blue circle: agent's location)

Figure 4.2.b Complex 11*11 Grid maze (red box: jackpot goal, green box: small reward, blue circle: agent's location)

Figure 4.2.c Length per episode of DQN, random agent in simple grid maze

Figure 4.2.d Total reward per episode of DQN, random agent in the simple grid maze

Figure 4.2.1 DQN agent results in simple grid-maze

We compared the performance of the DQN agent to the random policy agent in both simple and complex grid mazes to show the optimality difference due to the complexity of the environment. In the simple gridmaze environment. DQN agent shows optimizing behavior of lowering episodic length, increasing total reward per episode and reward rate with respect to random policy agent, in the simple grid maze.

As seen in Figure 4.2.c, 4.2.d unlike the random agent which shows nearly constant behavior score (length, reward) across the entire episodes, the DQN agent shows fluctuating behavior score which optimizes and converges after 300 episodes. This optimizing behavior score showed more clearing in the reward rate score in Figure 4.2.e, 4.2.f (linear coefficient = 0.025, $p < 0.01$)

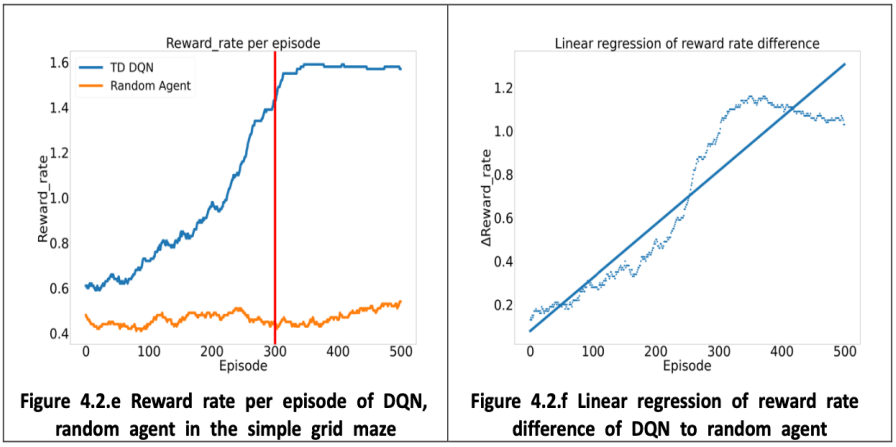

Figure 4.2.e Reward rate per episode of DQN, random agent in the simple grid maze

Figure 4.2.f Linear regression of reward rate difference of DQN to random agent

Figure 4.2.2 Reward rate of DQN agent in simple grid maze

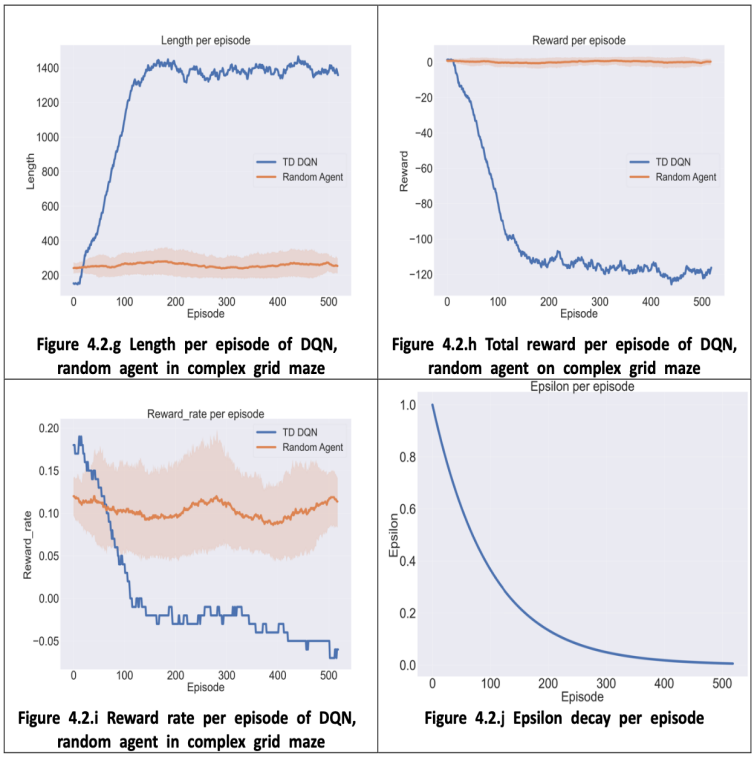

Figure 4.2.g Length per episode of DQN, random agent in complex grid maze

Figure 4.2.h Total reward per episode of DQN, random agent on complex grid maze

Figure 4.2.i Reward rate per episode of DQN, random agent in complex grid maze

Figure 4.2.j Epsilon decay per episode

Figure 4.2.3 DQN agent results in complex grid world

However, in the complex grid world environment shown in Figure 2-b, the DQN agent was not able to optimize within the given time window (519 episodes, episode length limit of 2000). This effect of the agent converging to non-optimal behavior is due to the lack of sample efficiency of the use of experience collected by the agent. Also, a relatively fast epsilon decay rate of ϵ to the DQN network optimization leads to non-optimal behavioral metrics in a complex grid maze environment.

2.3 Batch Sampling Increases Sample Efficiency for the DQN Agent in Complex Grid Maze with Fixed Reward Position

The limitation of sample efficiency in a complex grid world environment can be solved by using batch sampling. Batch sampling originates its idea from experience replay which is important to build an agent that can continuously learn with sample efficiency.(Rolnick et al. n.d.) We tested the increase in sample efficiency using the fixed goal position shown in Figure 4.3 (behavior metric of single-step DQN, batch sample DQN, random agent on the first fixed goal position for 519 episodes)

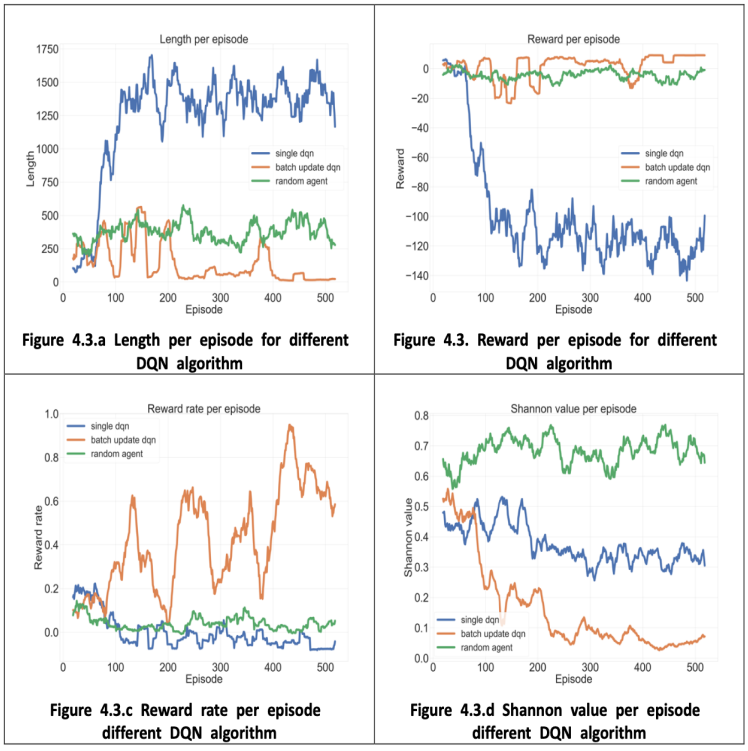

Figure 4.3.a Length per episode for different DQN algorithm

Figure 4.3. Reward per episode for different DQN algorithm

Figure 4.3.c Reward rate per episode different DQN algorithm

Figure 4.3.d Shannon value per episode different DQN algorithm

Figure 4.3.1 Behavioral results of batch sample, sinlge-step DQN and random agents. (first fixed reward position)

As shown in figure 4.3.a ~ 4.3.c DQN agent using batch sampling showed the most optimal behavior. This optimizing behavior is best shown in Figure 4.3.c as an increasing reward rate across every episode. Also, the batch sample DQN agent showed the most convergence of actions measured by the Shannon information value of actions in Figure 4.3.d. Although the DQN algorithm with single updates showed a convergence of action compared to the random agent shown in Figure 4.3.d, it could not converge to the optimal behavior shown in Figure 4.3.a ~ 4.3.c.

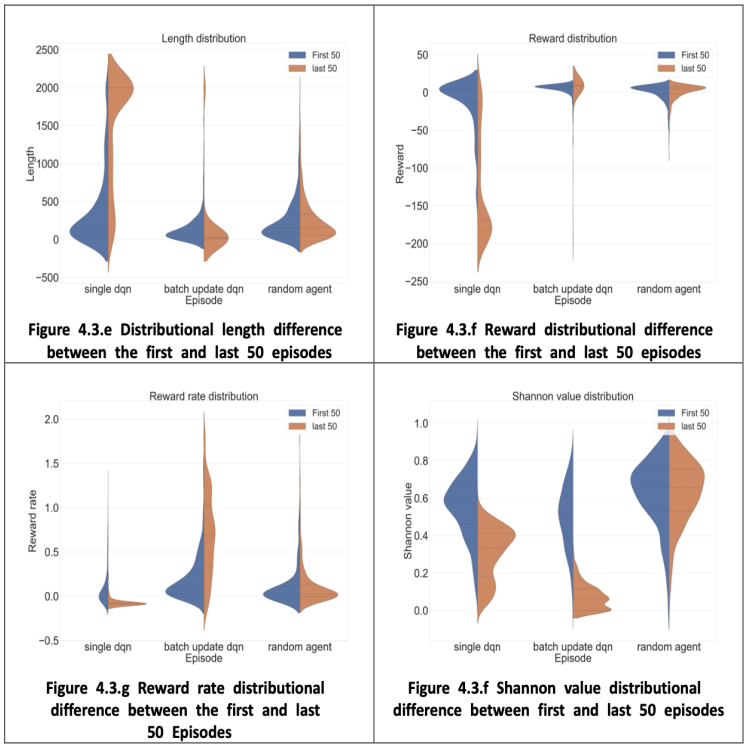

Figure 4.3.e Distributional length difference between the first and last 50 episodes

Figure 4.3.f Reward distributional difference between the first and last 50 episodes

Figure 4.3.g Reward rate distributional difference between the first and last 50 Episodes

Figure 4.3.f Shannon value distributional difference between first and last 50 episodes

Figure 4.3.2 Distribution of behavioral data on every fixed reward positions

Figure 4.3.e~4.3.f shows the violin plot of the behavior metric of the DQN single update, the DQN batch sample, and the random agent at all fixed reward locations. The first and last 50 episodes are selected to show distributional changes in the behavioral metric. Unlike a random agent that shows no changes in distribution for the first and last 50 episodes, the DQN agents showed changes in the behavioral metric. While the batch DQN agent showed an increase in the reward, the reward rate, and decreases in the length and Shannon action value, which indicates the agent was converging to optimal behavior, the single-step DQN algorithm failed to show such optimal behavior.

2.4 Batch Sampling DQN Agents are Slow to Adapt to the Changes in the Dynamic Reward Environment

We trained batch-sampling DQN agents in the changing reward environment shown in Figure 3.2. Unlike in the environments with fixed reward positions, batch-sampling DQN agents were not able to adapt to changing reward locations as shown by behavioral metric in Figure 4.4.

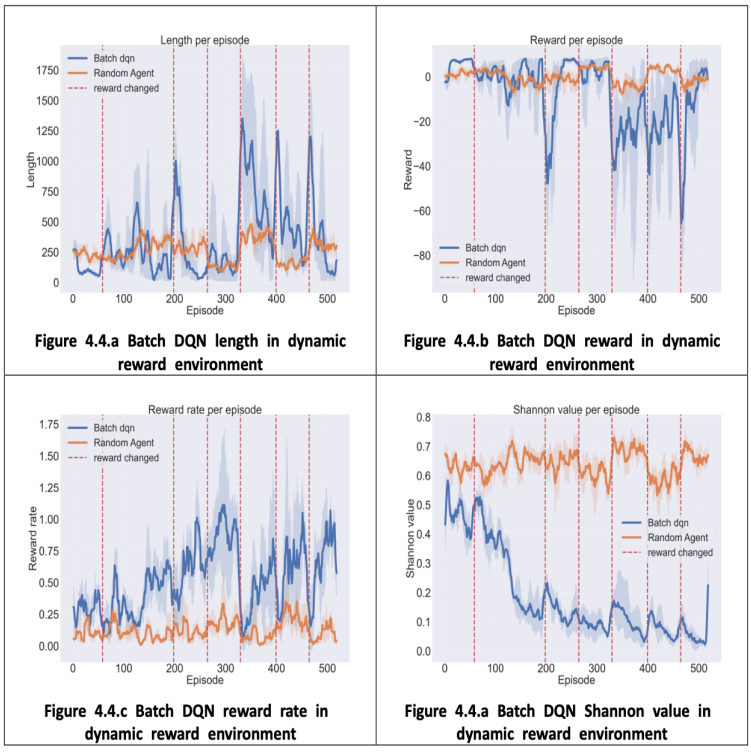

Figure 4.4.a Batch DQN length in dynamic reward environment

Figure 4.4.b Batch DQN reward in dynamic reward environment

Figure 4.4.c Batch DQN reward rate in dynamic reward environment

Figure 4.4.a Batch DQN Shannon value in dynamic reward environment

Figure 4.4.1 Behavioral metric of batch-sampling DQN agents in a dynamic reward environment

Although batch learning DQN agents have better reward rates and convergence of actions measured by Shannon value shown in Figure 4.4.c, 4.4.d, failed to quickly optimize lengths and maximize total reward and minimize length shown in Figure 4.4.a, 4.4.b.

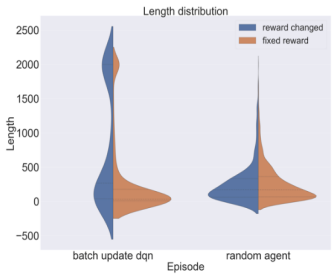

Figure 4.4.e Violin plot of length distribution categorized by fixed reward, and changed reward episodes

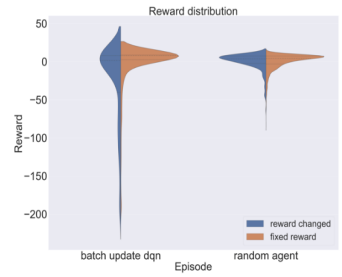

Figure 4.4.f Violin plot of reward distribution categorized by fixed reward, and changed reward episodes

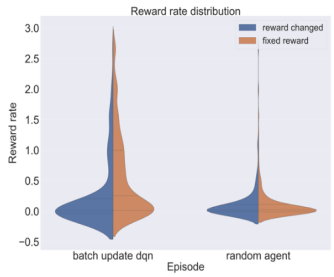

Figure 4.4.g Violin plot of reward rate distribution categorized by fixed reward and changed reward episodes

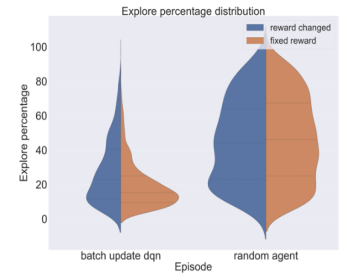

Figure 4.4.h Violin plot of visited percentage of map distribution categorized by fixed reward and changed reward episodes

Figure 4.4.2 Behavioral metric of fixed, changed reward episodes in a dynamic reward environment

The non-adaptive behavior of batch-sampling DQN agents can be seen in Figure 4.4. Every episode was split into two groups, reward changed (0 ~ 10 episodes within reward changed), and reward fixed(other episodes). Unlike reward-fixed episodes, reward-changed episodes showed an increase in episodic length and a decrease in episodic reward. This effect is due to the low exploring behavior of the batch-sampling DQN agent compared to the random agent shown in Figure 4.4.h

2.5 Model-based DQN Agent Quickly Adapts to the Changing Reward Environment than Batch Update DQN Agent

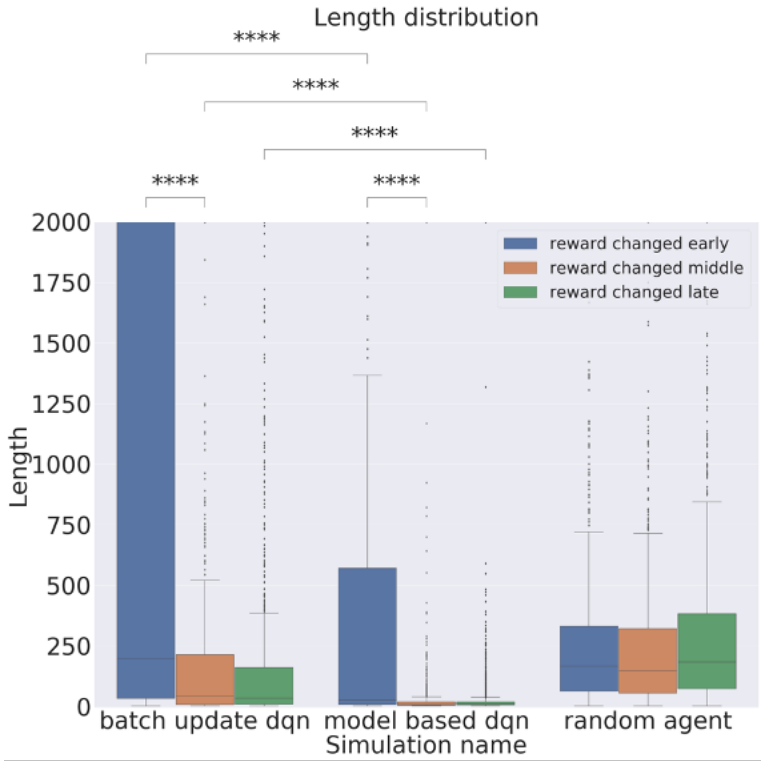

Figure 4.5.a Length distribution of different models on early, middle, and late stages after reward position change

Model-based agents were trained in the same dynamic reward environment. The behavior metric results were divided into three stages, early, middle, and late after reward position change. (early: 1~20 episodes after reward change, middle: 21~40 episodes after reward change, late: other; mean of episode number of reward positions is 64.875) As seen in Figure 4.5.a model-based DQN agents showed better behavioral scores than batch-update DQN agents. In the first 20 episodes after a reward position change, agents' episodic length was worse than random agents. However, model-based agents showed better results in reward, reward rate shown in Figure 4.5.b, 4.5.c

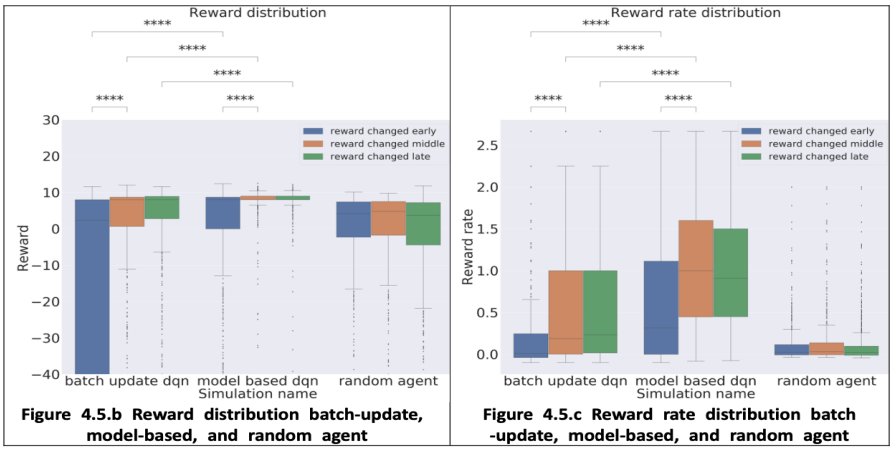

Figure 4.5.b Reward distribution batch-update, model-based, and random agent

Figure 4.5.c Reward rate distribution batch -update, model-based, and random agent

Figure 4.5.1 Reward, reward rate distribution of batch learning, model-based, and random agent in dynamical reward environment

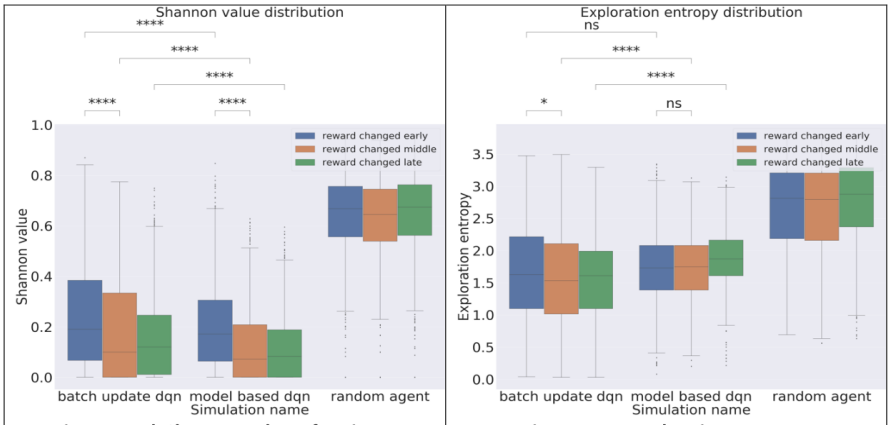

Figure 4.5.d Shannon value of actions

Figure 4.5.e Exploration entropy of visited states

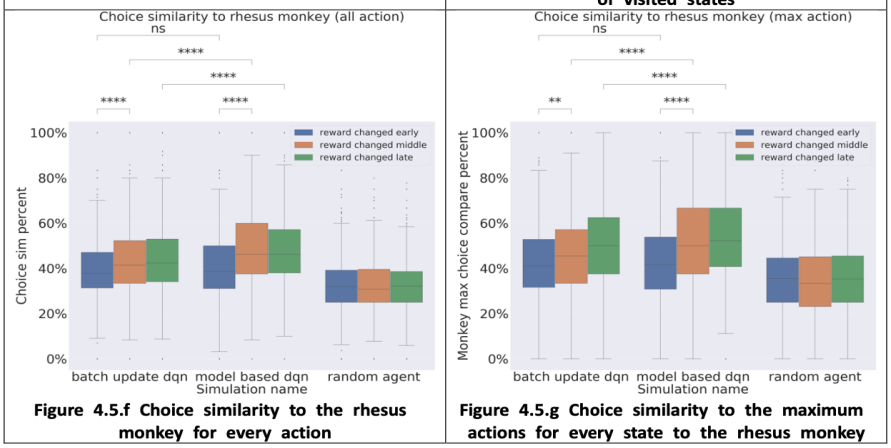

Figure 4.5.f Choice similarity to the rhesus monkey for every action

Figure 4.5.g Choice similarity to the maximum actions for every state to the rhesus monkey

Figure 4.5.2 Shannon value of actions, exploration entropy of visited states, and choice similarity to the rhesus monkey's behavior across multiple agents

This relatively fast adaptation of model-based DQN agents results in increases in choice similarity to the rhesus monkey as shown in Figure 4.5.c. (choice similarity in Figure 4.5.f compares using every path of the rhesus monkey. Maximum choice comparison in Figure 4.5.g sorts the path of the rhesus monkey in states and compares only the maximum action chosen by the rhesus monkey) This similarity increase is likely due to the easing of exploit policy of past rewards shown by increases of action Shannon value and exploration entropy in Figure 4.5.d, 4.5.e

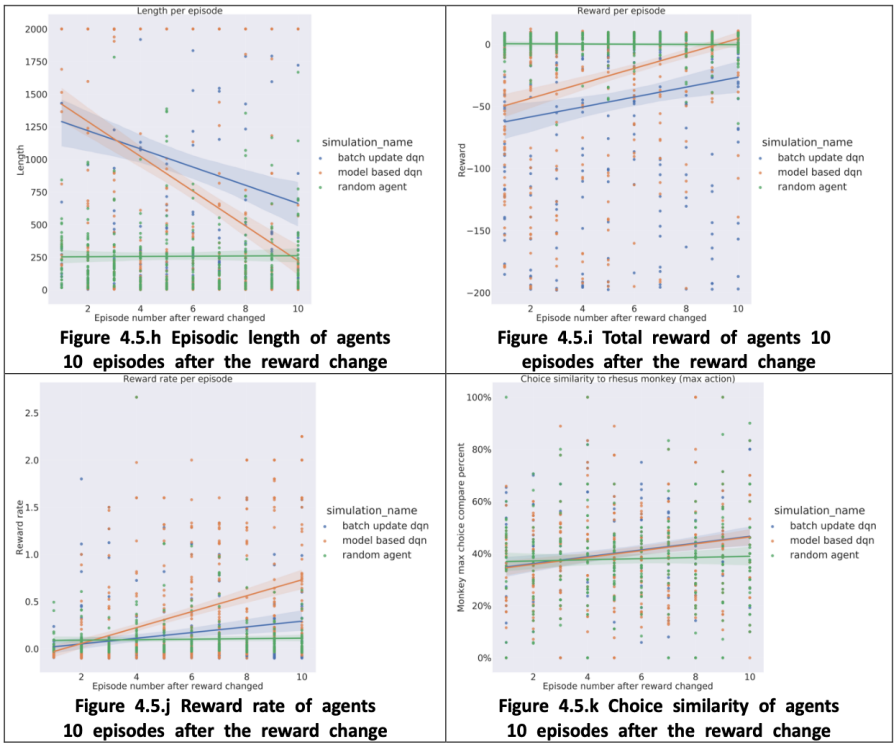

Figure 4.5.h Episodic length of agents 10 episodes after the reward change

Figure 4.5.i Total reward of agents 10 episodes after the reward change

Figure 4.5.j Reward rate of agents 10 episodes after the reward change

Figure 4.5.k Choice similarity of agents 10 episodes after the reward change

Figure 4.5.3 Behavior metrics and choice similarity to rhesus monkey of agents within 10 episodes of reward change

However, model-based agents showed similar results to batch-learning DQN agents within 5 episodes after reward position change as shown in Figure 4.5.h ~ 4.5.j. This causes behavior differences between model-based agent to the rhesus monkey which results in a similarity decrease to the random level shown in Figure 4.5.k.

2.6 Curiosity-based Agent Shows Most Adaptive Behavior in the Dynamic Reward Environment

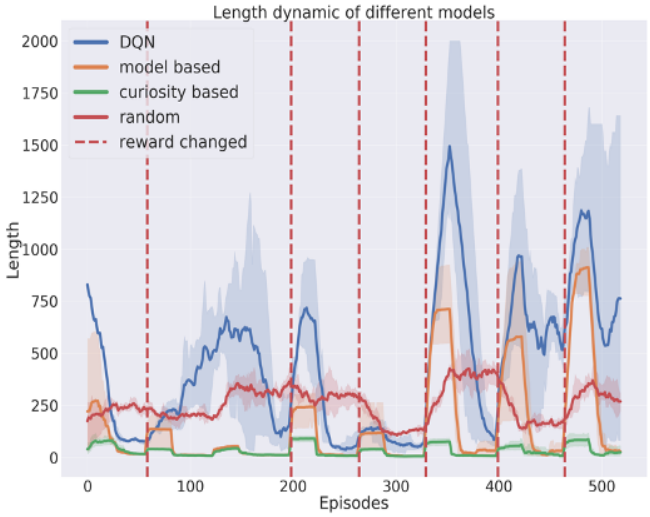

Figure 4.6.1 Length per episode results of different models

Unlike model-based agents which showed non-optimal behavior within short episodes after the reward changed as shown in previous results, curiosity-based agents showed better adaptive behavior after the reward change in the dynamic reward environment shown in Figure 4.6.1. (Figure 4.6.1 ~ 4.6.3 is the results of three agents per each model, windowed averaged of size 10) This adaptivity difference between model-based agents and curiosity-based agents is significant in the latter reward-changed episodes (300~). Reward and reward rate results showed similar differences as shown in Figure 4.6.2.

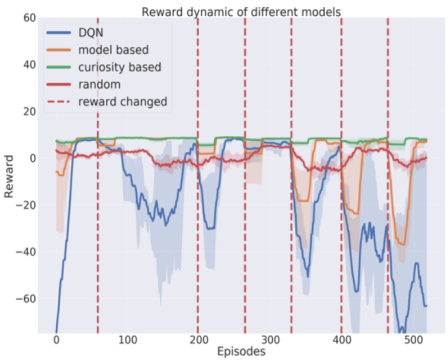

Figure 4.6.a Reward per episode results of different models

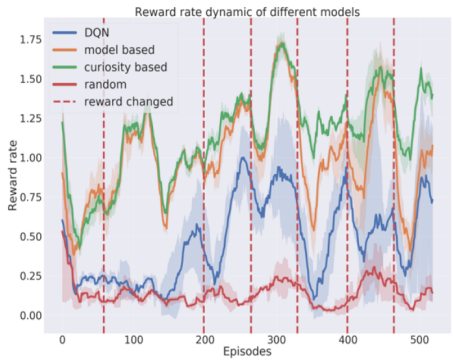

Figure 4.6.b Reward rate per episode results of different models

Figure 4.6.2 Reward, reward rate results in the dynamic reward environment

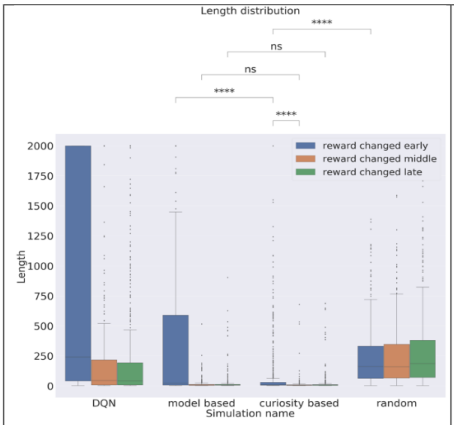

Figure 4.6.c Reward distribution of different rates in early, middle, and late stages after reward change

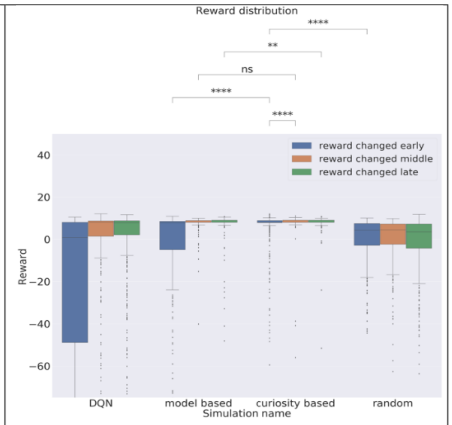

Figure 4.6.d Length distribution of different rates in early, middle, and late stages after reward change

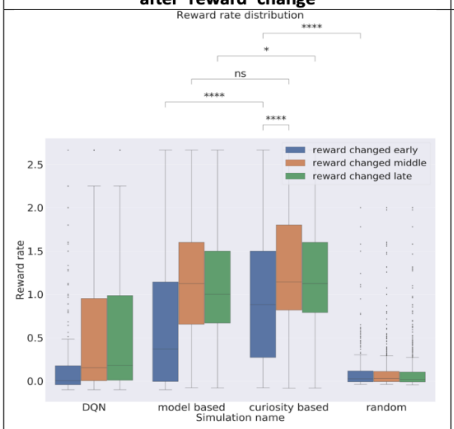

Figure 4.6.e Reward rate distribution of Different rates in early, middle, and late stages after reward change

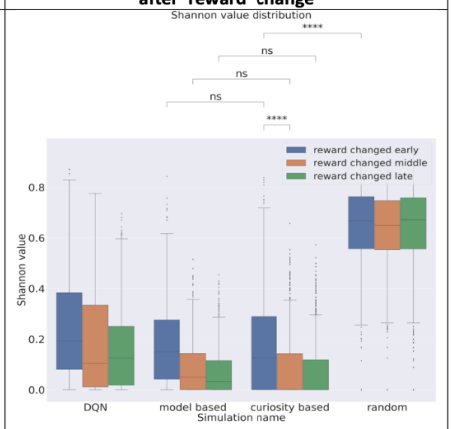

Figure 4.6.f Shannon values of actions distribution of different rates in early, middle, and late stages after reward change

Figure 4.6.3 Behavioral result distribution of multiple models in the dynamic reward environment grouped by early, middle, and late stages after the reward change (20 episodes each)

The effect of adaptivity increase of curiosity-based maximum entropy update is significantly shown in the early stages after reward change as shown in Figure 4.6.c ~ 4.6.e. (within 20 episodes after reward change) Also this effect is not due to overall random exploration increase (e.g. epsilon increase) which will show high Shannon value of actions, exploration percentage, and exploration entropy like a random agent. The curiosity maximum entropy update results in structured exploration of less-visited states, resulting in lower exploration percentage, exploration entropy of states, and Shannon value of actions shown in Figure 4.6.f ~ 4.6.h

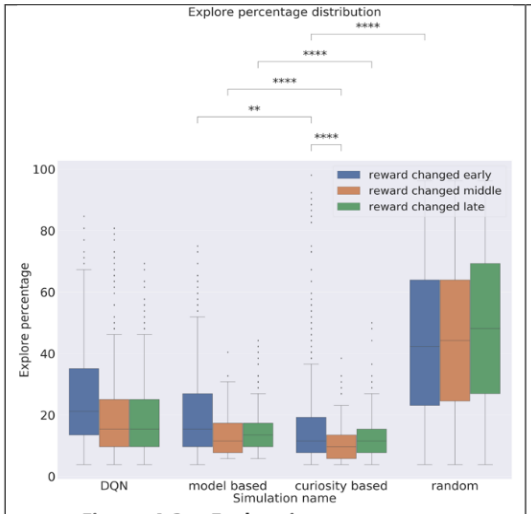

Figure 4.6.g Exploration percentage distribution of different rates in early, middle, and late stages after reward change

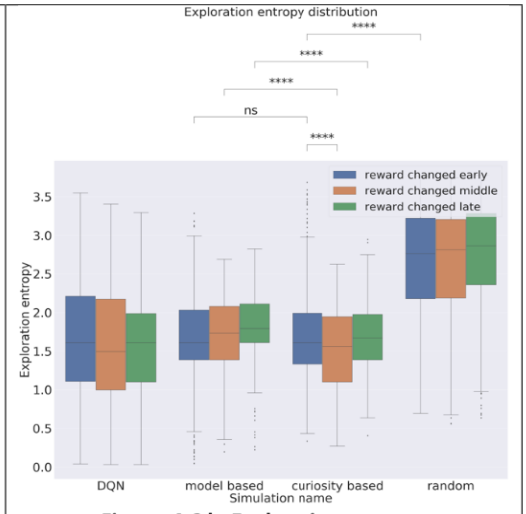

Figure 4.6.h Exploration entropy distribution of different rates in early, middle, and late stages after reward change

Figure 4.6.4 Explorative behavior distribution of multiple models in the dynamic reward environment grouped by early, middle, and late stages after the reward change (20 episodes each)

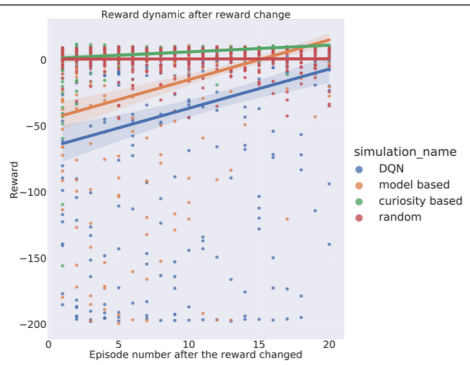

Figure 4.6.i Reward results within 10 episodes after reward position change

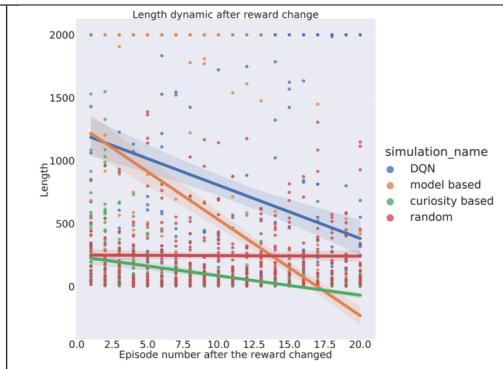

Figure 4.6.j Length results within 10 episodes after reward position change

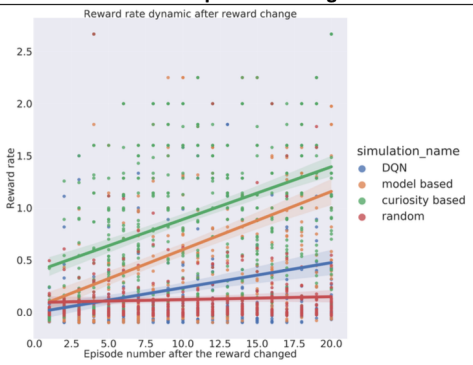

Figure 4.6.k Reward rate results within 10 episodes after reward position change

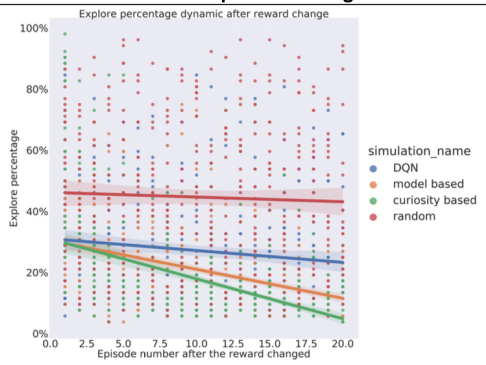

Figure 4.6.l Exploration percentage results within 10 episodes after the reward position change

Figure 4.6.5 Behavioral results of multiple models within 10 episodes after reward position change

Figure 4.6.i ~ 4.6.l shows the effects of curiosity maximum entropy update in more adaptive behavior right after the reward position change. As shown in Figure 4.6.l, the curiosity model showed increased exploration in the first episode after the reward changed and quickly adapted to the new reward position better than batch-updating DQN and model-based DQN agents.

Since curiosity maximum entropy update only happens in unknown reward conditions, the effect of this can be seen as the explorative behavior of the agent's online exploit policy affected by the agent's offline curiosity explore policy shown in Figure 4.6.6 ~ 4.6.7. (1 episode before the reward change and after the reward change) Exploit offline results of Figure 4.6.6 shows higher visit near the goal position because model-based simulation happens only after the goal is known. (after reaching the first goal at the end of the episode, some simulation happens) Similar results of model-based and batch-updating DQN agents can be found in supplementary results 2 and other reward locations of curiosity-based agents can be found in supplementary results 4.

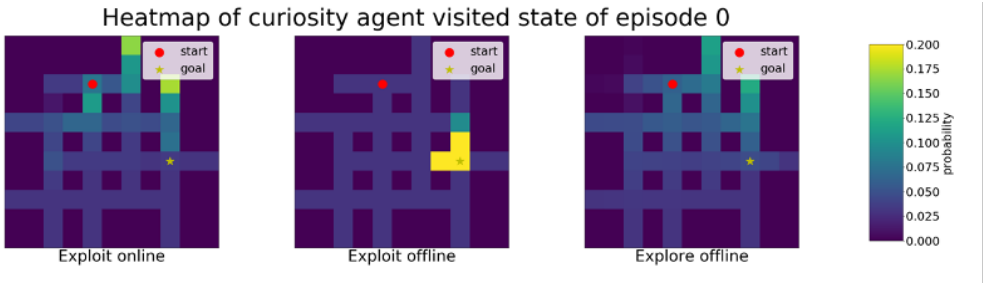

Figure 4.6.6 Heatmap of agent's exploit policy, explore policy on online(real) and offline (model simulated) environment

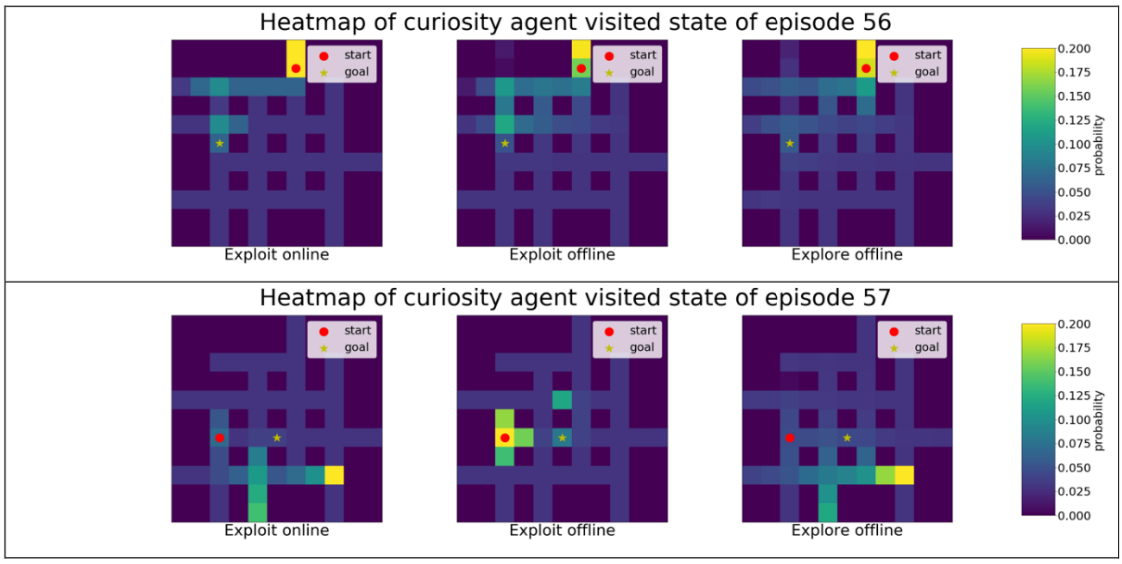

Figure 4.6.7 Heatmap change of agent's exploit policy, explore policy on online(real) and offline (model simulated) environment after reward change

2.7 Curiosity-based maximum entropy update agent showed the most behavior similarity to the rhesus monkey

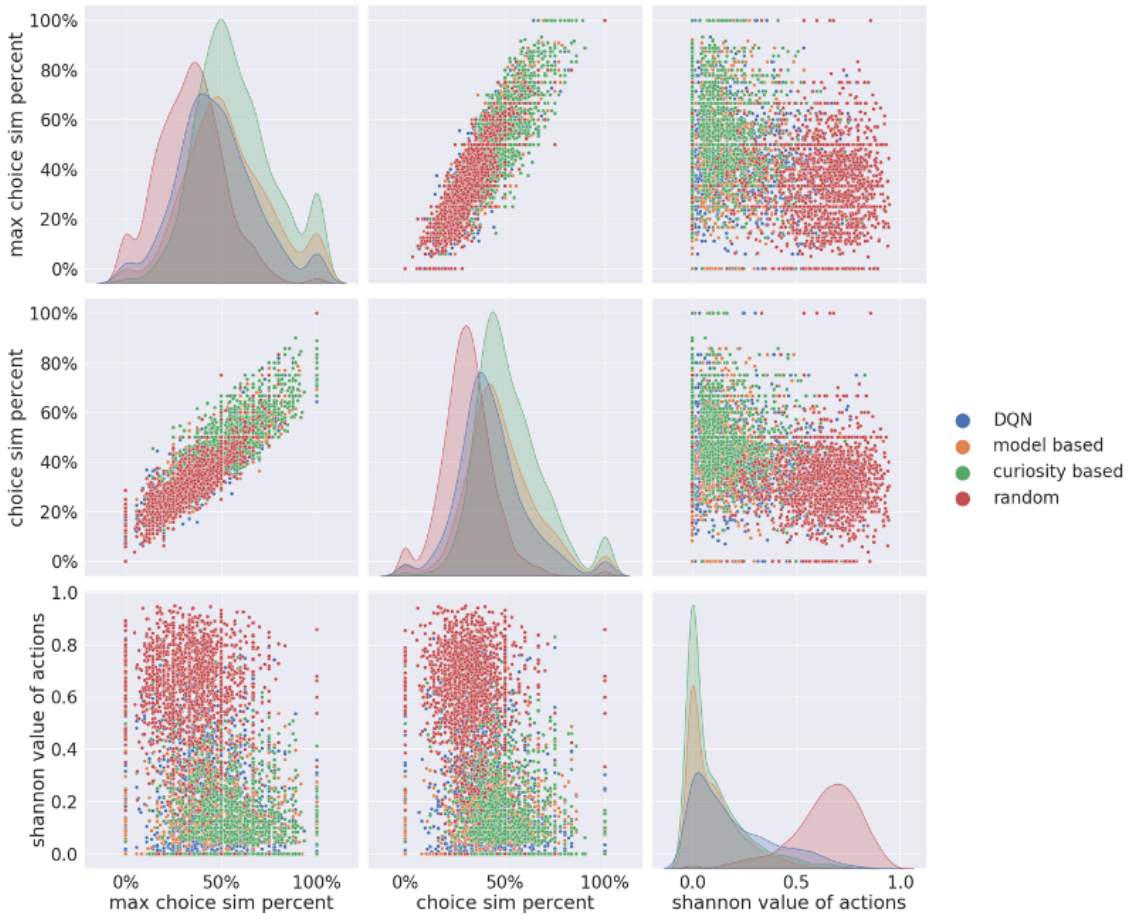

Figure 4.7.1 Pair plot of choice similarity and Shannon value of multiple agents

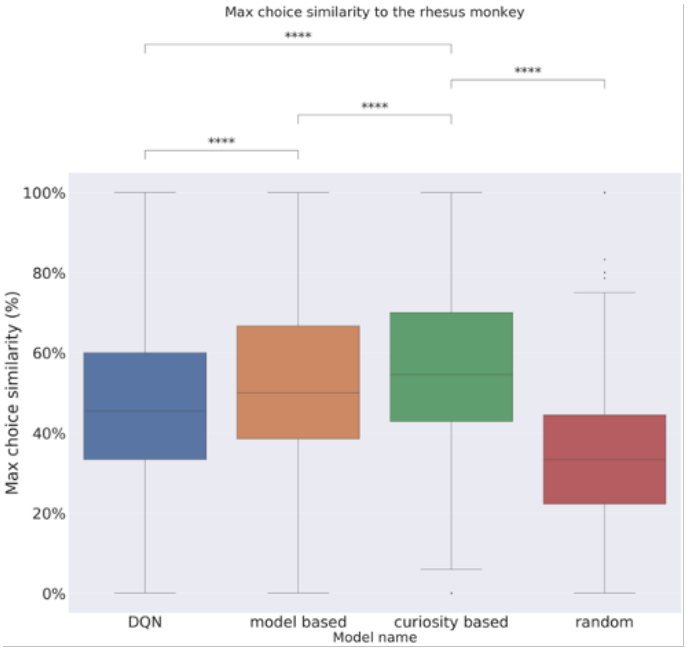

Figure 4.7.2 Max choice similarity distribution across different models

Curiosity-based maximum entropy update agent showed increased choice similarity to the rhesus monkey as shown in Figure 4.7.1 and 4.7.2. This effect of more similarity increases when we only compare maximum action of the rhesus monkey per states as shown in Figure 4.7.1 as increased slope of choice similarity and maximum choice similarity. Also curiosity-based agent showed low Shannon value distribution to the other agents, which indicated it's taking overall more planned action in both exploit and explore behavior.

2.8 Rhesus monkey showed most Representational Dissimilarity Matrix (RDM) similarity to the curiosity-based agent

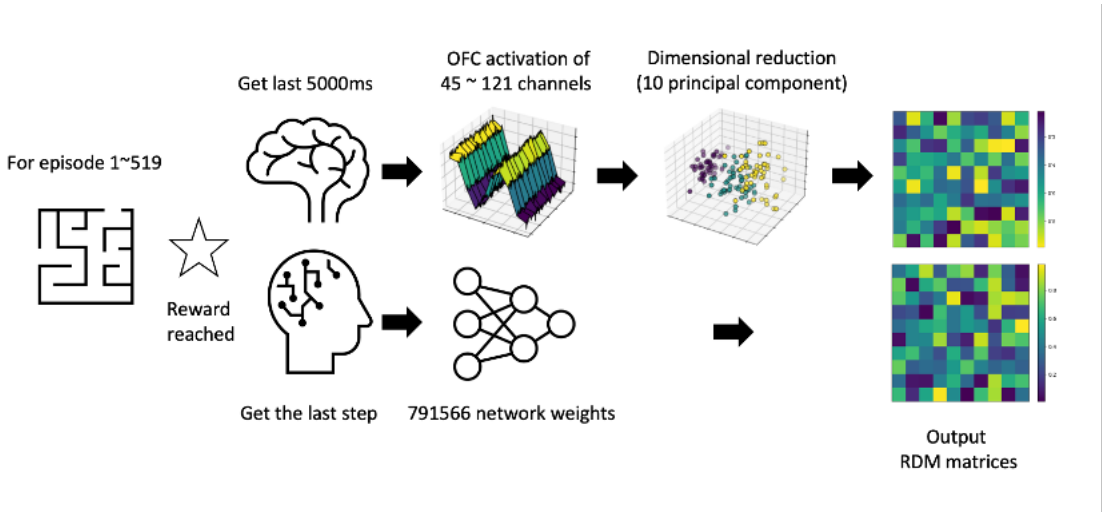

Figure 4.8.1 RDM matrices processing procedure

To compare neural similarity between the rhesus monkey and artificial agents, we calculated 519 * 519 episodes of Representational Dissimilarity Matrices (RDM). The OFC data of the rhesus monkey of the final 5000ms wasselected and pre-processed with dimensional reduction using PCA(Principal Component Analysis). After this, RSA (Representational Similarity Analysis) was calculated using cosine-similarity.

Similarly neural network weights of artificial agents, each totaling 791566(5 layers with 512 hidden dimensions, 2 input dimensions, and 4 output dimensions) were recorded at the last step of each episode and used to calculate RSA. Each model (batch update, model-based, and curiosity-based DQN) was trained 2 times for comparison.

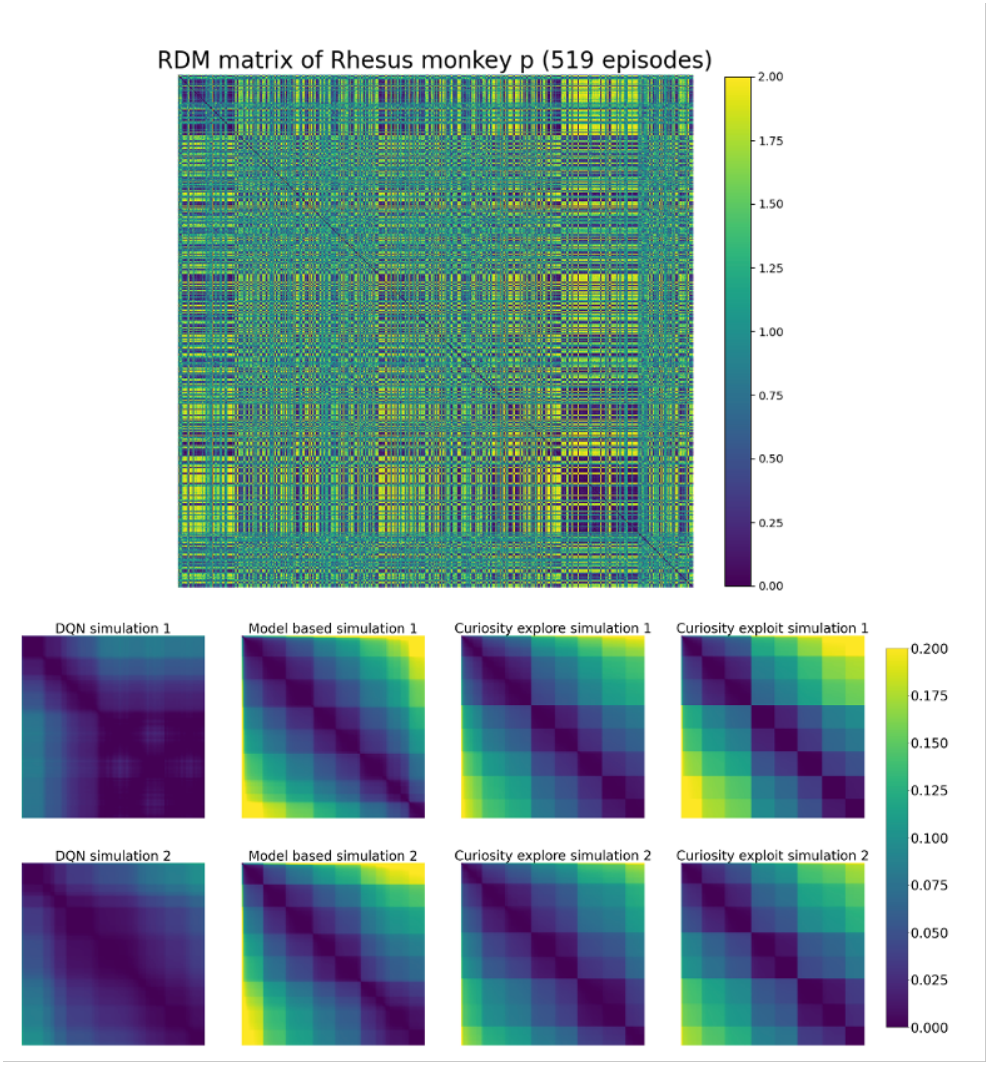

Figure 4.8.2 RDM matrices of rhesus monkey and multiple models (all episodes)

Batch-update DQN agent showed the most similarity across entire episodes as shown in Figure 4.8.2 meaning that there was no significant update of the neural network to dynamic changing reward conditions. Both model-based and curiosity-based agents showed significant differences between all 8 reward conditions as shown in Figure 4.8.2. Although artificial agents showed a low range of dissimilarity to the rhesus monkey, this effect is likely due to the high dimensionality of data (791566 dimensions) to dimensionality reduced OFC data. (10 dimensions)

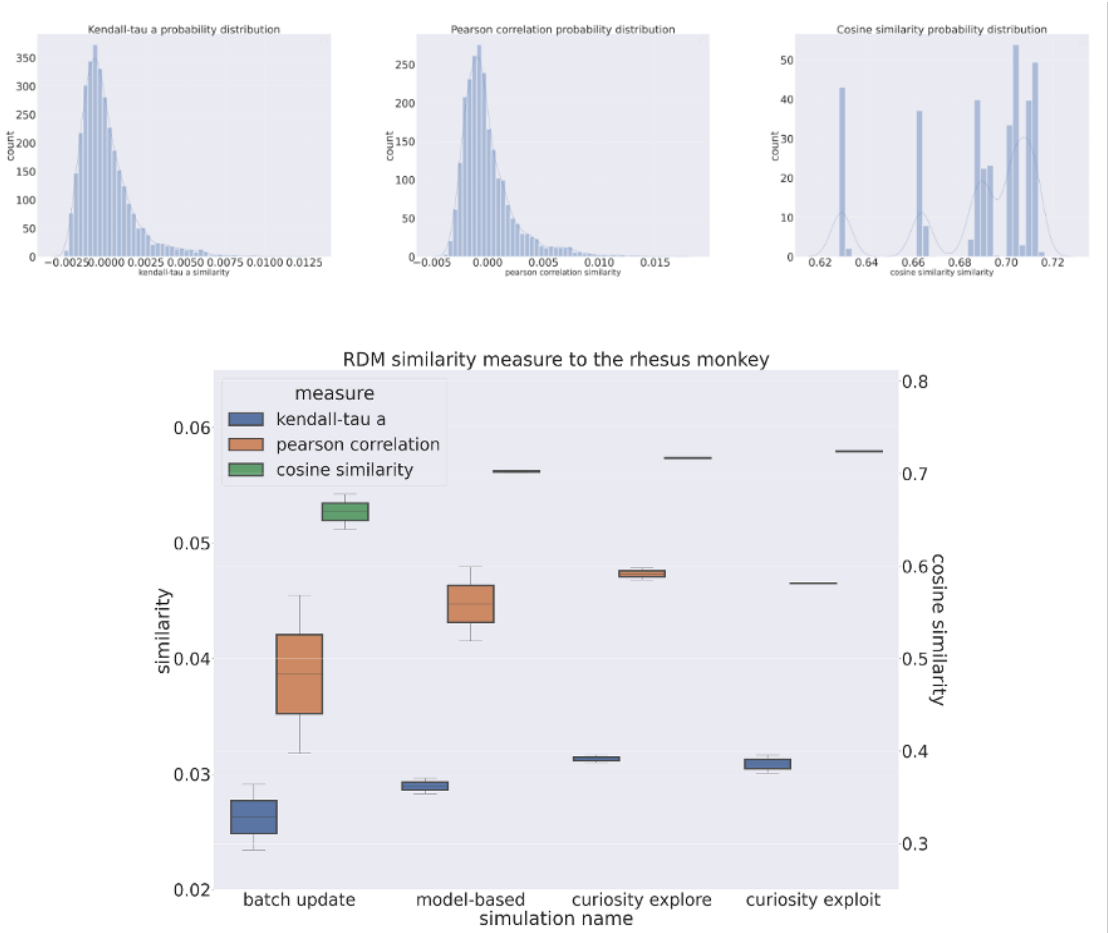

Figure 4.8.3 RDM similarity to the rhesus monkey using multiple measures (top figure shows the distribution of each measure between randomly shuffled RDMs) (all episodes)

Similarity of multiple reinforcement learning RDMs to the rhesus monkey OFC RDM was calculated using three different metrics Kendall-tau a, Pearson correlation, and cosine similarity. Each metric was calculated with 1000 shuffled RDMs of each model per simulation (2 simulations per model) to calculate the significance of these similarities. All the similarities above showed significant similarity to the random shuffled RDM matrices. ($p \approx 0$)

The overall similarity of curiosity-based models, both explore and exploit policy showed higher RDM similarity to model-based, batch-update DQN agent.

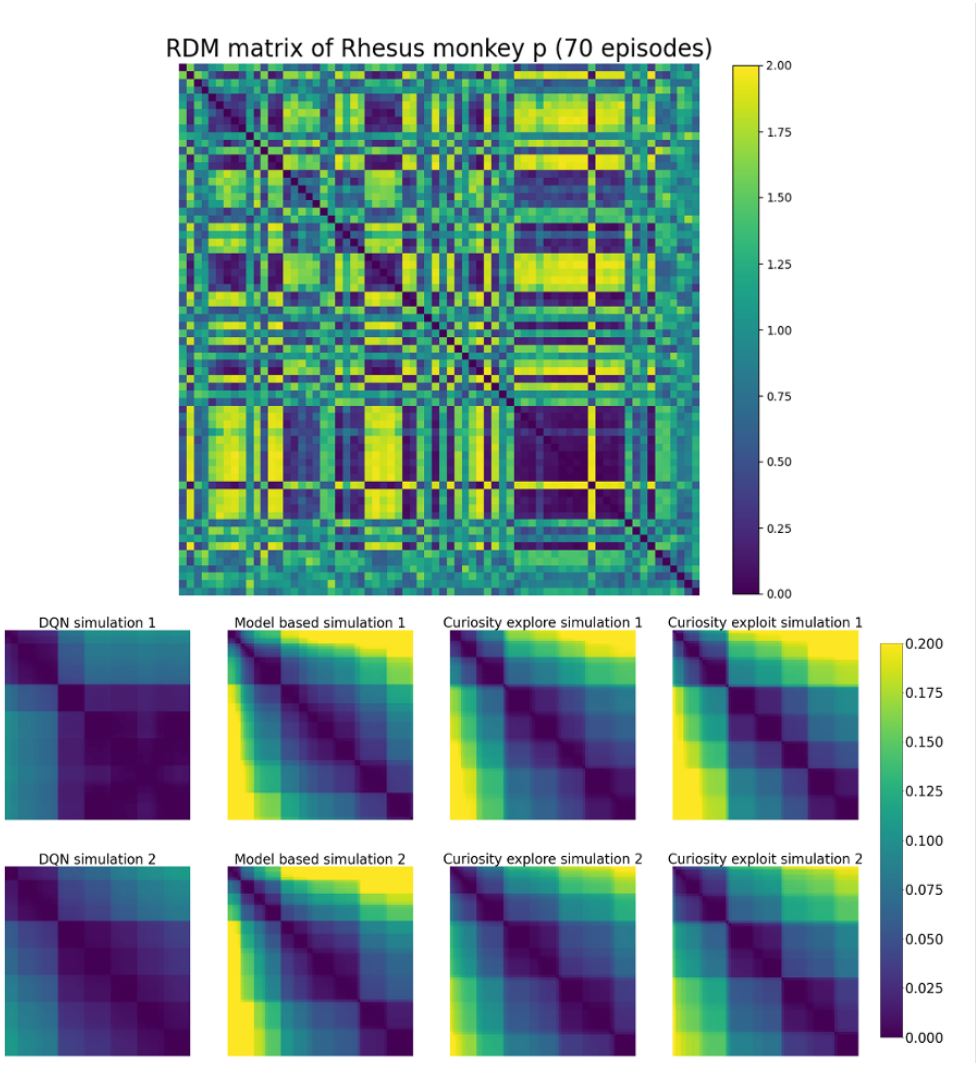

Figure 4.8.4 RDM matrices of rhesus monkey and multiple models (10 episodes within reward change) (70 episodes)

RDMs showed higher similarity within 10 episodes of rewards position change(early stages) as shown in Figures 4.8.4, and 4.8.5. Overall Kendall-tau-a and Pearson correlation were higher in the early stages compared to the all-episode case. Only DQN algorithm showed no significant different to random shuffled RDMs matrix as shown in Table 3.

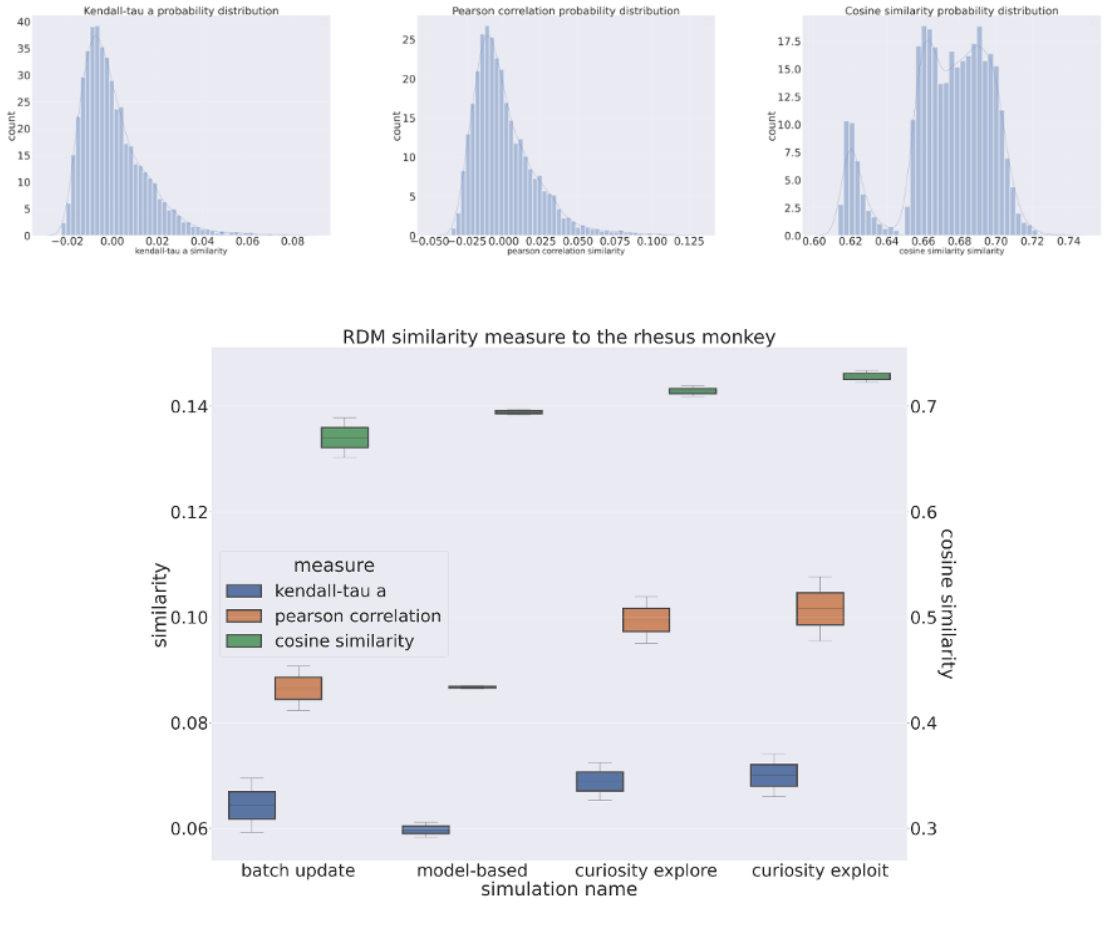

Figure 4.8.5 RDM similarity to the rhesus monkey using multiple measures (top figure shows distribution of each measure between randomly shuffled RDMs)(all episodes)

Table 3 P-value and similarity of models to random symmetrical shuffled RDM matrices

Simulation name	Measure method	p value	similarity
Batch update DQN	Cosine similarity	0.24725	0.134076
	Kendall-tau a	**0.00225**	**0.064377**
	Pearson correlation	**0.00475**	**0.086553**
Model based DQN	Cosine similarity	**0.00775**	**0.138910**
	Kendall-tau a	**0.00300**	**0.059758**
	Pearson correlation	**0.00600**	**0.086780**
Curiosity Based DQN Explore policy	Cosine similarity	**0.00450**	**0.142872**
	Kendall-tau a	**0.00175**	**0.068911**
	Pearson correlation	**0.00300**	**0.099502**
Curiosity Based DQN Exploit policy	Cosine similarity	**0.00400**	**0.145671**
	Kendall-tau a	**0.00275**	**0.070044**
	Pearson correlation	**0.00325**	**0.101622**

2.9 OFC Representational Dissimilarity Matrix (RDM) Shows Temporal Clustering

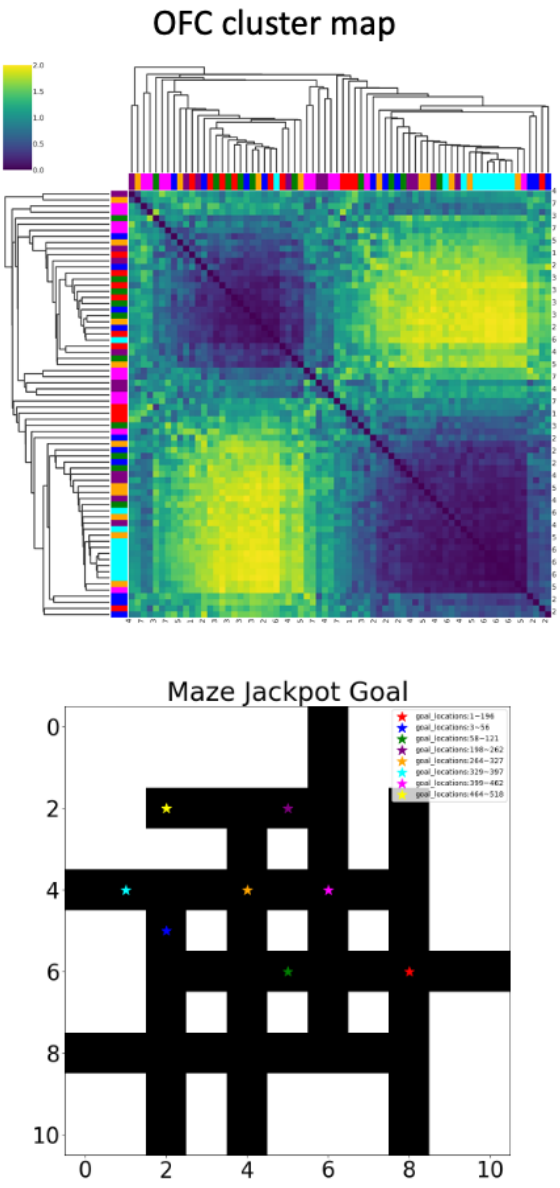

Figure 4.9.1 Hierarchal clustered OFC RDM

(70 episodes)

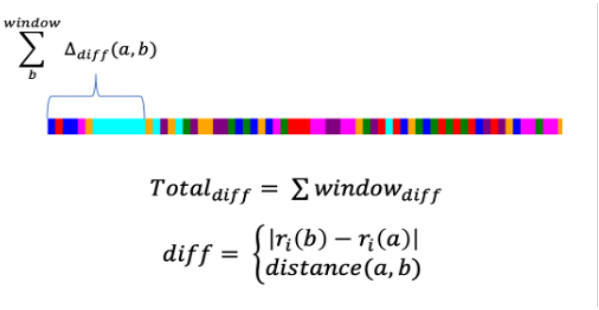

Figure 4.9.2 Difference measure calculation method (r_i indicate reward index)

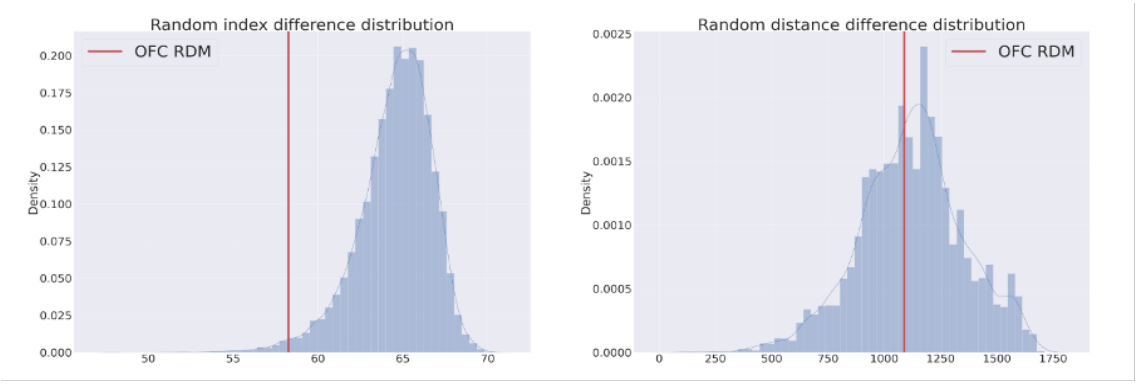

Figure 4.9.3 Distribution of difference measure of random shuffled reward number

A hierarchical clustered map of OFC RDM is shown in Figure 4.9.1. The colors represent the reward location of the selected episode. Only 10 episodes after reward change were selected, 70 episodes total. The sorted order of hierarchal clustered reward numbers was used to test if the clustering was due to temporal or positional differences.

The OFC RDM showed temporal(index) clustering shown in Figure 4.9.3 (difference measure = 58.26, p-value = 0.0126), and no Euclidian distance clustering. (difference measure = 1089.56, p-value = 0.4191)

2.10 Final Layers of Exploit, Explore Network Shows Most Similarity to the Rhesus Monkey's OFC

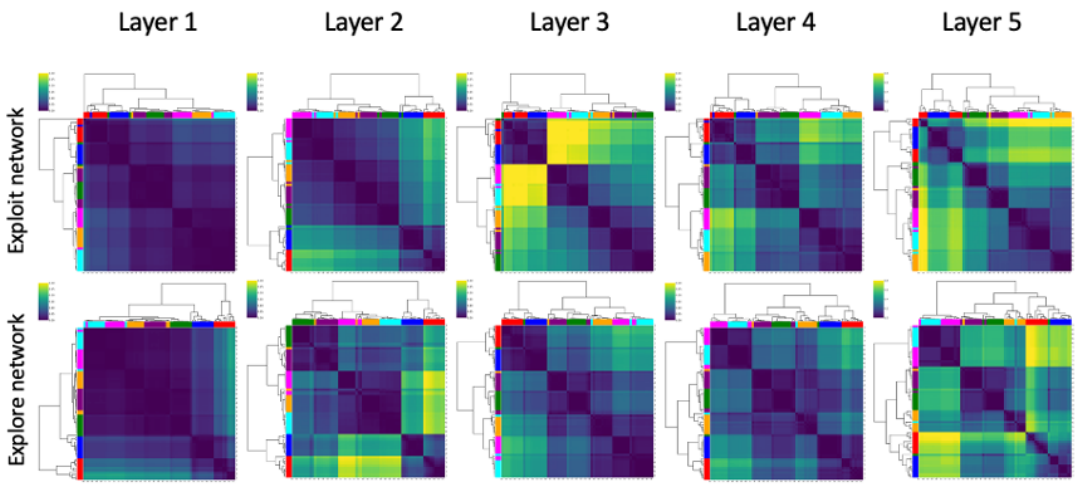

Figure 4.10.1 RDMs of curiosity-based agent by layer

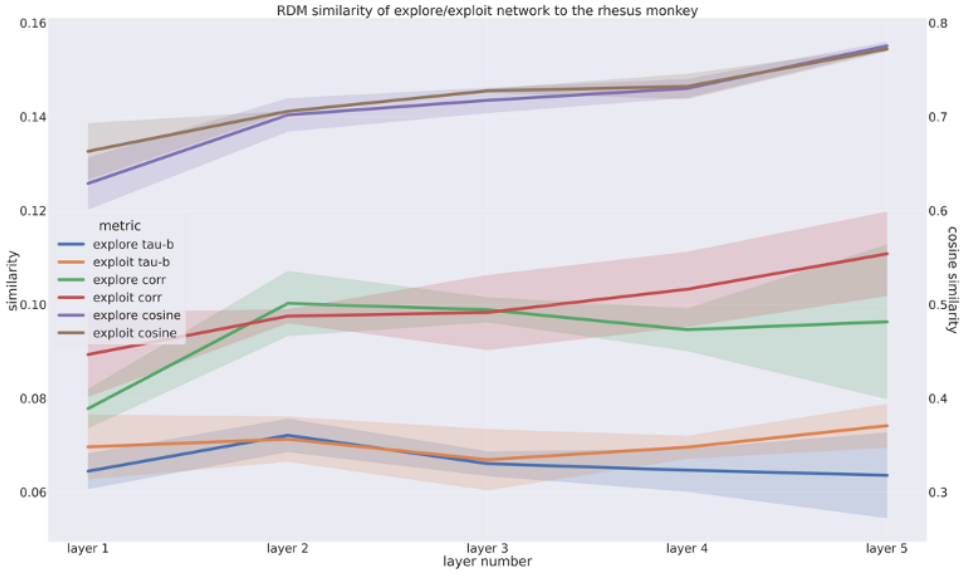

Figure 4.10.2 RDM similarity of each layer to the rhesus monkey's OFC

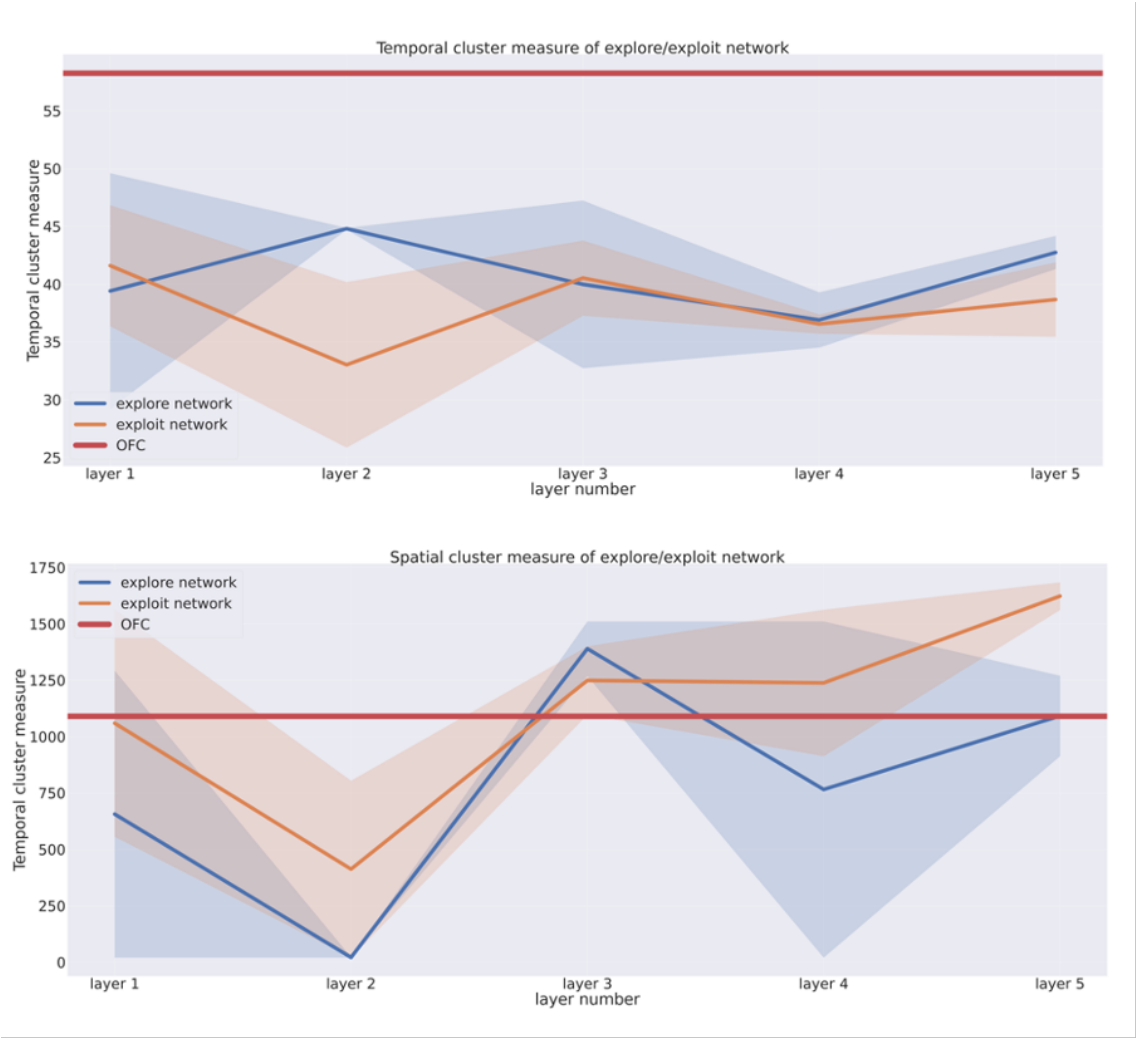

Figure 4.10.3 Temporal and spatial cluster measures of explore and exploit network layers

The hierarchical clustering of each layer of the explore and exploit network was performed as shown in Figure 4.10.1. layer 5 showed the highest dissimilarity between trials. (color scale is 0~0.8 unlike layer 1~4 which is 0 ~ 0.2)

The higher layer RDMs show increased similarity to OFC RDM as shown in Figure 4.10.2. Also like OFC RDM, all of the layers showed temporal clustering as shown in Figures 4.10.1 and 4.10.3. But the lower level layer showed spatial shown in Figure 4.10.3 which suggests OFC is related to higher level explore and exploit network which is important for adaptation in a dynamic environment.

제 5장 Discussion and Conclusion

Discussion

The OFC is known to function as a signaling reward identity and it's changing dynamic. (Howard and Kahnt 2021; Stalnaker et al. 2018) This function introduces flexibility in the network and helps the natural agents to show adaptive behavior.(Schoenbaum et al. 2009)

In this research, we showed that our novel curiosity-based DQN model showed the most optimal adaptive behavior in a dynamic reward environment. Also, this model showed the most similarity to the rhesus monkey's behavior and neurological similarity calculated using RDM.

This research also shows the possibility of creating an artificial OFC to increase adaptivity in a dynamic reward environment using model-based, curiosity-based reward functions and networks. Also, we showed that OFC showed the most similarity to the final layers of the explore and exploit network. This suggests that OFC might be related to two distinct reward pathways to help adaptation in a dynamic environment.

A recent paper about OFC-subcortical pathways shows two different adaptive behaviors of macaque monkeys using experience-based or infer-based updating. (Oyama et al. 2023) In this paper the authors suggest that experience-based value updating is related to the OFC-rmCD(rostromedial caudate nucleus) and infer-based updating is related to the OFC-Mdm (mediodorsal thalamus) pathways.

Our results show that both explore and exploit networks of curiosity-based agents have similarities to the OFC. Since learning of explorative behavior to reward in the known map is like infer-based learning and the exploit behavior to known reward and environment is based on experience-based simulation, we suggest that this finding shows a possibility of OFC network being composed of two networks(OFC-rmCD: exploit network, OFC-Mdm: explore network) with non-linear(max-entropy) interaction.

However, since the curiosity-based model used in this paper only uses the curiosity explore network to update the exploit network in unknown reward situations, there are still limitations in creating continually curious-driven models.

Also, since this research is only done on the dynamic of location changes to the

reward, building models that can adapt to dynamic transfer function (e.g. changing map by opening and closing of doors) and reward size should be further researched.

Conclusion

In summary, we have shown the adaptivity increase of our novel curiosity-based reinforcement learning agent which shows similarity in both behavioral and neural network dynamics. Thus, this research gives insight into creating natural intelligence alike an experience-driven deep learning model that can adapt to the dynamic environment.

제 6장 Appendix

1. Model Design

1.1 Model-Free DQN(Deep-Q-Network) with Experience Replay

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize exploration probability ε = 1

For episode = 1, episodes do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$

Store transition (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in D

Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in D

Perform a gradient descent step on θ with respect to the network parameters q

End while

If done

Anneal exploration variable ε

End if

End For

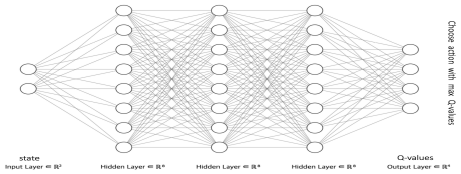

Figure A.1 Neural network of DQN

1.2 Model-Free DQN(Deep-Q-Network) without Experience Replay

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize exploration probability ε = 1

For episode = 1, episodes do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$

Calulate transition (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$)

Perform a gradient descent step on θ with respect to the network parameters q

End while

If done

Anneal exploration variable ε

End if

End For

2. Model-Based DQN

Initialize replay memory D to capacity N

Initialize model memory Dm to capacity N

Initialize reward location memory in model MR as FIFO deque to capacity N

Initialize small reward memory in model MR-small

Initialize action-value function Q with random weights q

Initialize exploration probability ε = 1

Initialize model as n*n array as map in model M_map

For episode = 1, episodes do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$

Store transition $(s_t, a_t, r_t, s_{t+1}, done_{t+1})$ in D

Store transition (s_t, a_t, r_t, s_{t+1}) in MMap and MR-small

If done and r > 0 do

Store s_{t+1} in MR

End if

Sample random minibatch of transitions $(s_t, a_t, r_t, s_{t+1}, done_{t+1})$ in D

Perform a gradient descent step on θ with respect to the network parameters

Perform Model based simulation based on Eq(a)

End while

If done

Anneal exploration variable ε

End if

End For

1.3 Explore DQN agent

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize exploration probability ε = 1

For episode = 1, episodes do

While Not(done or truncated)

With probability e select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$

From eq(6) calculate rt with exploration reward function

Store transition (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in D

Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in D

Perform a gradient descent step on θ with respect to the network parameters q

End while

If done

Anneal exploration variable ε

End if

End For

1.4 Curiosity-based RL

Initialize replay memory D to capacity N

Initialize model memory Dm-exploit to capacity N

Initialize model memory Dm-explore to capacity N

Initialize model as n*n array as map in model M_Map

Initialize reward location memory in model MR as FIFO deque to capacity N

Initialize small reward memory in model MR-small

Initialize action-value(environment reward) function with random weights

Initialize action-value(internal reward) function with random weights

Initialize exploration probability ε = 1

For episode = 1, episodes do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe reward rt, st+1, donet+1, truncatedt+1

Store transition $(s_t, a_t, r_t, s_{t+1}, done_{t+1})$ in D

Store transition (s_t, a_t, r_t, s_{t+1}) in M_Map and MR-small

If done and r > 0 do

Store s_{t+1} in MR

Else

update MR

End if

Sample random minibatch of transitions $(s_t, a_t, r_t, s_{t+1}, done_{t+1})$ in D

Set based on eq(1)

Perform a gradient descent step on θ with respect to the network parameters

Perform model-based simulation, curiosity based simulation on based on Eq(b), Eq(c)

do a cross entropy update of based on Eq(d)

End while

If done

Anneal exploration variable e

End if

End For

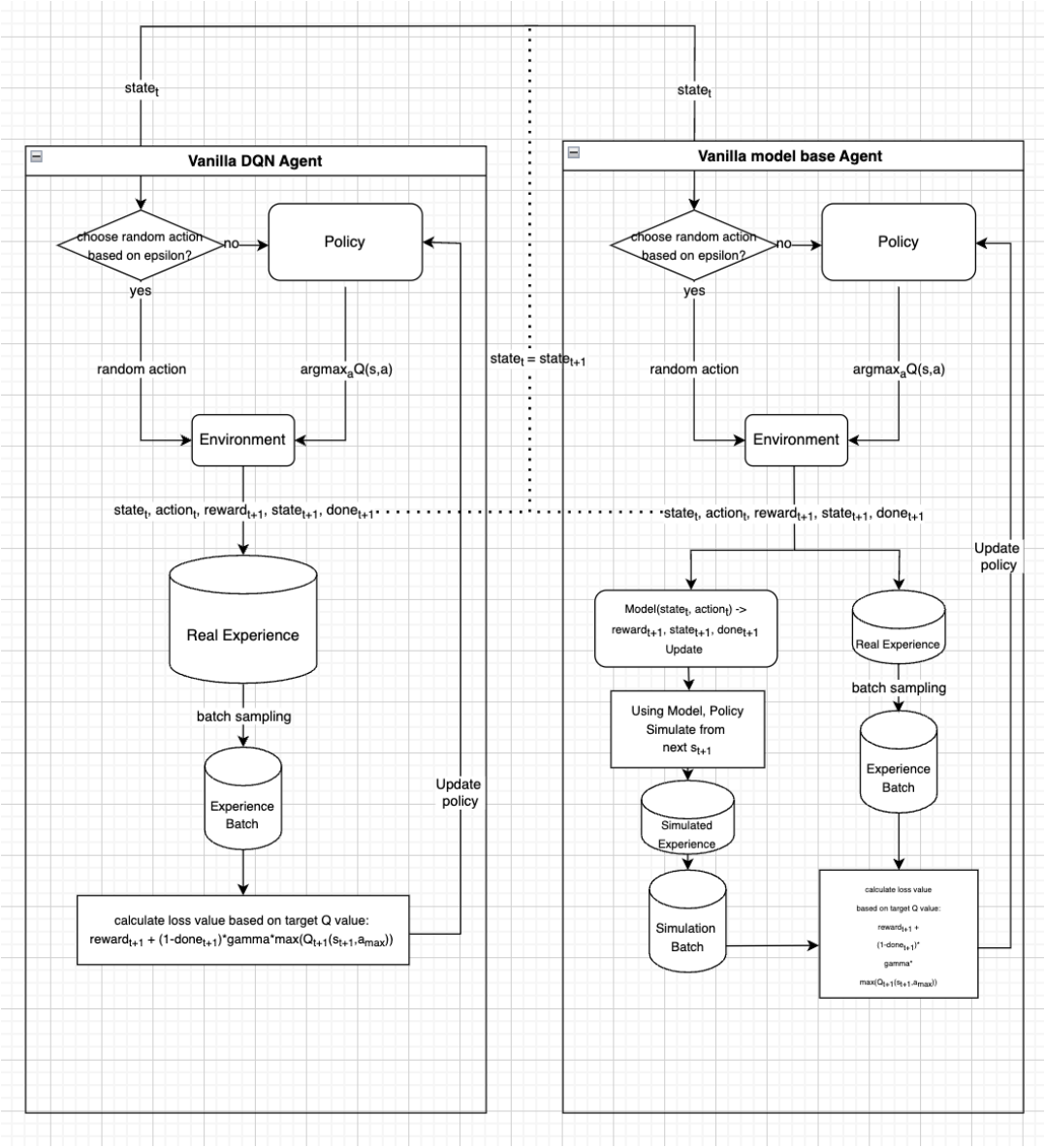

Figure A-2 Schematic of Model-Free and Model-Based DQN Agent

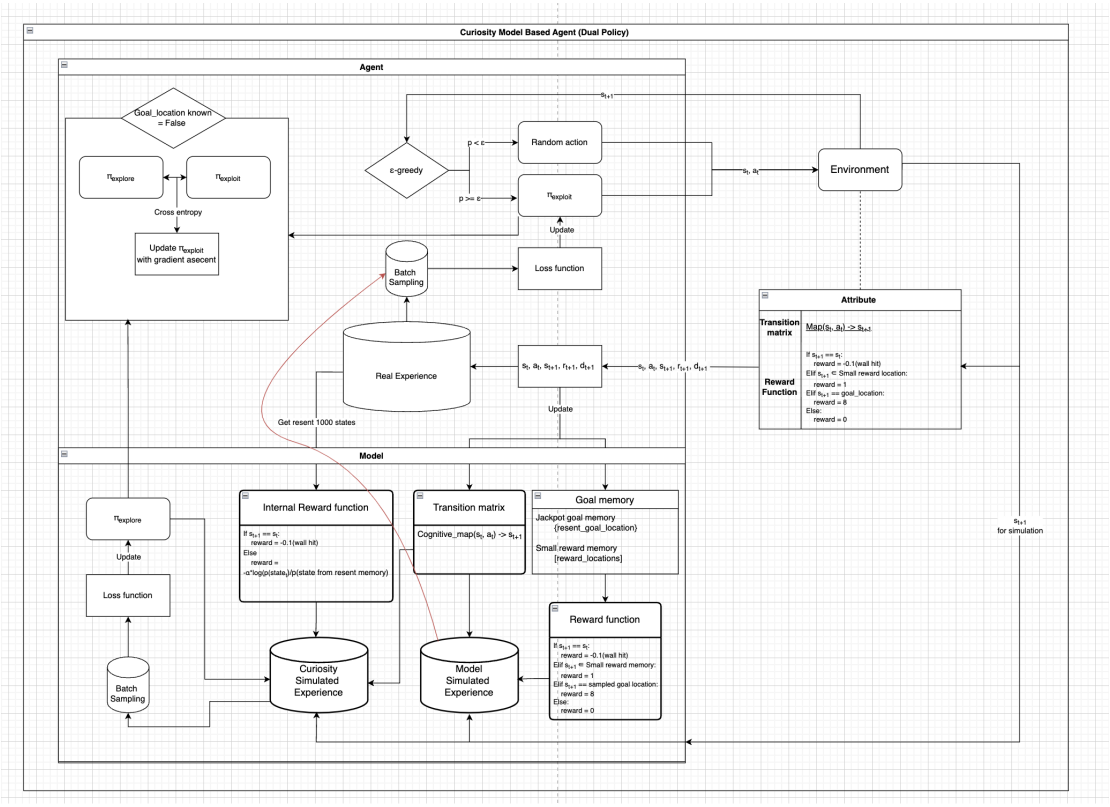

Figure A-3 Schematic of Dual Policy Curiosity-based Agent

2. Metric

2.1 Metric for Behavioral Data Analysis

1. Length

Length is calculated by total count of action that agent took for each episode. Maximum length limit was 2000 for the environment.

2. Reward

Reward was calculated by cumulative sum of rewards given episode. Reward function of environment follows.

$$r(s_t, a_t) = \begin{cases} 8 \ (\text{if } s_{t+1} = goal) \\ 1 \left(\text{if } s_{t+1} \in rewards_{small} \ , \ s_{t+1} \notin \bigcup_{i=1}^{t} s_i \right) \\ -0.1 \ (\text{if } s_{t+1} = s_t) \\ 0 \ (else) \end{cases}$$

$$Reward_i = \sum_{t=1}^{T} r_i(s_t, \ a_t) \ \text{for episode } i$$

4. Shannon Information Entropy

Shannon information value of agents, and monkey's behavioral data was calculated using which was calculated using counts of (s, a) pairs translated to the 2d maze environment.

$$p_i(s,a) = \frac{Count(s,a \mid i)}{l_i}$$

$$ShannonInfo = \sum_{(s,a)}^{(s,a) \in Path} -p_i(s,a) * \log(p_i(s,a))$$

3. Additional Equations and Model Simulations

3.1 Additional Equations

$$Eq(1) \quad r_t = -\alpha * \log(p(s_{t+1}\ ; memory) + \lambda) + \beta$$

(small number is needed for non-zero log input and for agent to take less steps)

(1) information reward of curiosity simulation

3.2 Additional Model Simulations

Eq(a) Model Based Simulation

For simulation_episode = 1, simulation_max do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a\ ; \theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$

Execute action at and observe using Model

Store transition (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in Dm

Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in D

Perform a gradient descent step on with respect to the network

End While

end For

Eq(b) Curiosity Based Exploit Simulation

Initialize simulation visited state memory Dsim to capacity N

For simulation_episode = 1, simulation max do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$

Execute action at and observe using Model

Store transition (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in Dm-exploit

Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}, $done_{t+1}$) in

Perform a gradient descent step on θ with respect to the network

End While

end For

Eq(c) Curiosity Based Simulation

Initialize simulation visited state memory Dsim to capacity N

For simulation_episode = 1, simulation max do

While Not(done or truncated)

With probability ε select a random action a_t

otherwise select $a_t = \arg max_a(Q(s_t, a;\theta))$

Execute action a_t and observe r_t, s_{t+1}, $done_{t+1}$ using reward function of eq(1)

Execute action at and observe using Model

Store transition (st, at, rt, st+1, donet+1) in Dm-explore

Store st+1 to visited state memory Dsim

Sample random minibatch of transitions (st, at, rt, st+1, donet+1) in Dm-explore

Perform a gradient descent step on θ with respect to the network parameters

End While

end For

Eq(d) cross entropy update of Explore Exploit policies

1. Using softmax function calculate p values of actions in both explore, exploit polices

2. Perform a gradient decent step on with respect to the network parameter of exploit policy which maximizes cross entropy outcome

$$\text{Loss} = \Sigma\, p\_explore(s,a) * \log(p\_exploit(s,a))$$

4. Supplementary Results

1. Model-based Learner Showed Adaptive Behavior in Changing Reward Environment with Long-term Model Simulation

We tested model-based agents with different simulation numbers, as shown in Table A.1. Simulation number is the number of simulations that the agent does that start from the current state in every step of the episode. Max simulation is the number of maximum steps that an agent can take for each simulation. (Simulation is stopped when the goal is reached.) For model comparison, the number of simulation numbers * max simulation was fixed to 60.

Table A.1 Simulation numbers and simulation maximum steps of different models

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Simulation Number	2	5	10	15	20	30
Max Simulation	30	12	6	4	3	2

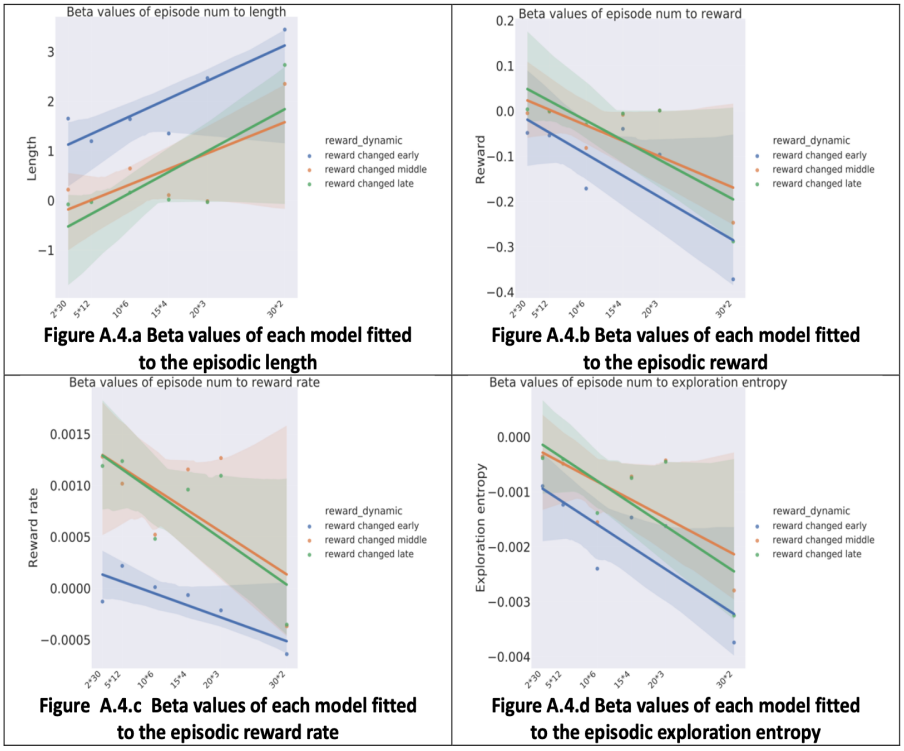

Figure A.4.a Beta values of each model fitted to the episodic length

Figure A.4.b Beta values of each model fitted to the episodic reward

Figure A.4.c Beta values of each model fitted to the episodic reward rate

Figure A.4.d Beta values of each model fitted to the episodic exploration entropy

Figure A.4.1 Beta values of each model fitted to different behavioral results.

The results of each model were fitted to linear regression grouped by early, middle, and late stages after reward change. (20 episodes each) As shown in Figure A.4.a ~ A.4.d models with less simulation number and longer-term simulations showed better results in all three stage groups. The long-term simulation models showed a lower increase in length in the early stages after reward change and less decrease in episodic reward and reward rate. Also across every group, the early stages after the reward showed significantly worse behavior scores.

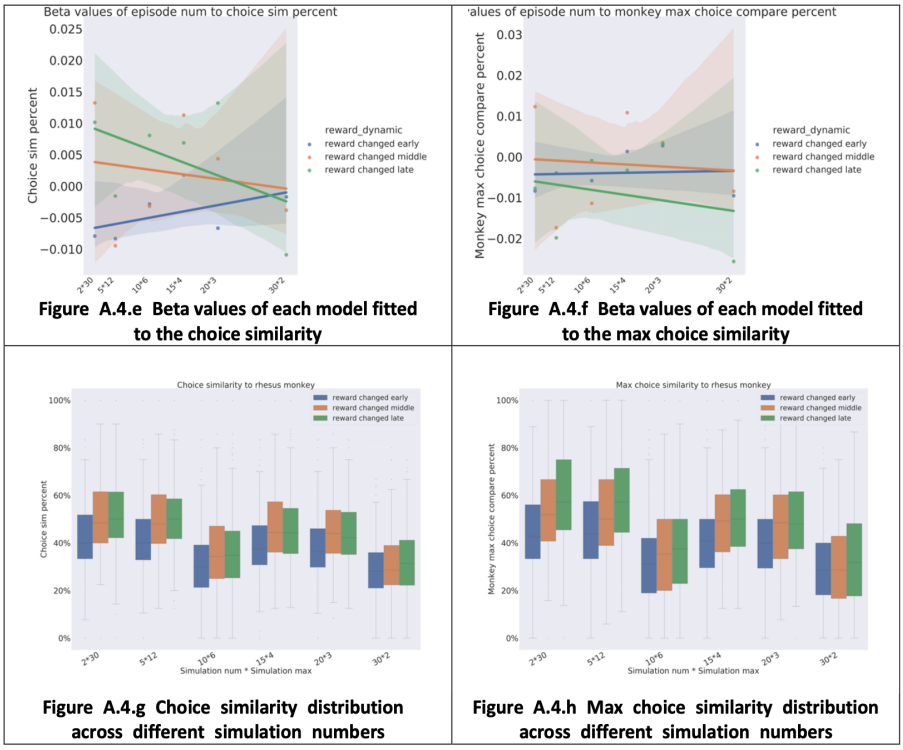

Figure A.4.e Beta values of each model fitted to the choice similarity

Figure A.4.f Beta values of each model fitted to the max choice similarity

Figure A.4.g Choice similarity distribution across different simulation numbers

Figure A.4.h Max choice similarity distribution across different simulation numbers

Figure A.4.2 Choice similarity between model-based agents and rhesus monkey

This sub-optimality of models in the shorter-term simulation maximum number group also results in different behavior to the rhesus monkey as shown in Figure A.4.e ~ A.4.h. Although beta values of the choice similarity group in the early stages after the reward change in the longer-simulation group showed more decreaseshown in Figure A.4.e, this effect is due to higher overall choice similarity shown in Figure A.4.g.

With these results, we have chosen the model with 5 simulation numbers and 12 maximum steps for comparison to the other DQN models after section 4.5.

2. Model-based Agents Showed Increased Adaptivity to Changing Rewards Due to Increased Visits and Simulation of Near-reward States

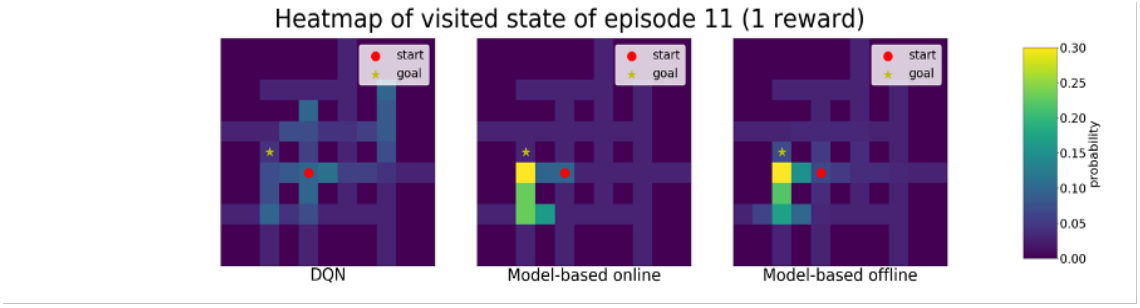

Figure A.4.3 Heatmaps of visited state across models 10 episodes after 1st reward

Figure A.4.3 shows heatmaps of visited states of batch-update DQN agent and model-based DQN agent. (5*12 simulation) Model-based online shows visited states of actual environment and model-based offline shows visited states in model simulation. As we can see in Figure A.4.3, the model-based agent showed an increase in visits closer to the goal location both in the online experience and offline simulation compared to the batch-update DQN agent. This effect is also shown in other reward positions in the Figures below.(10 episodes after reward changed)

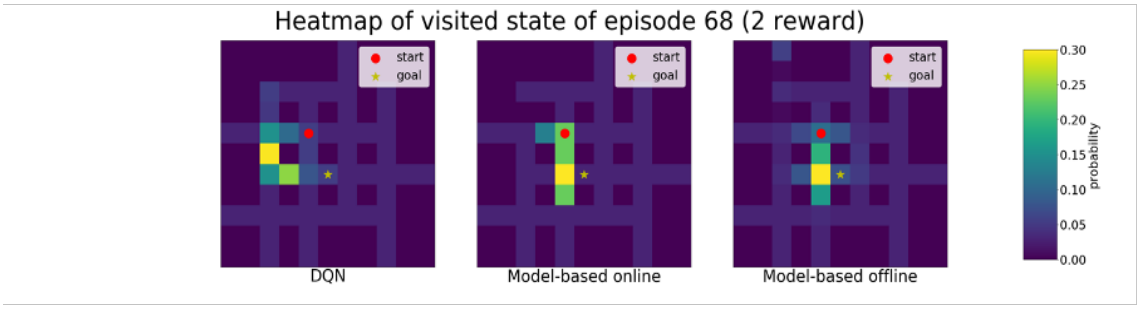

Figure A.4.4 Heatmaps of visited state across models 10 episodes after 2nd reward

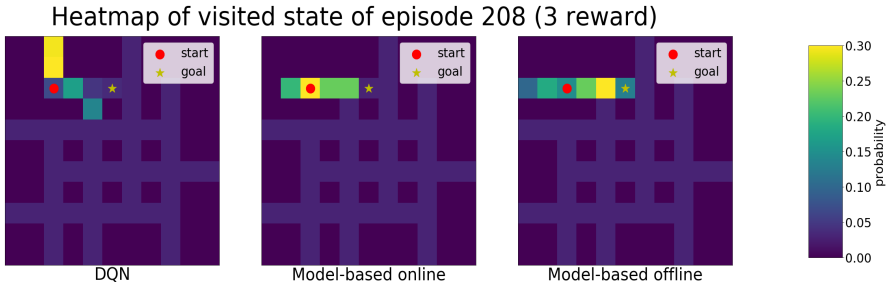

Figure A.4.5 Heatmaps of visited state across models 10 episodes after 3rd reward

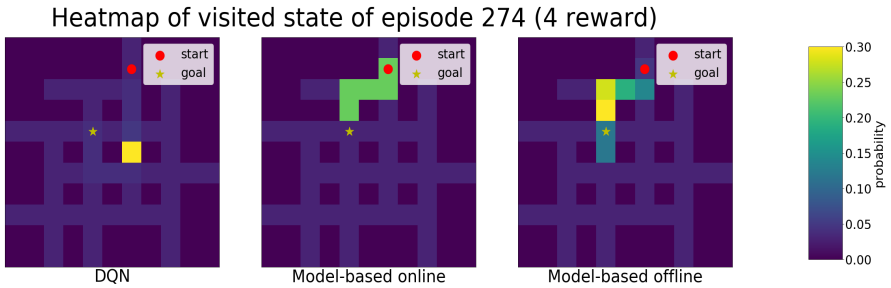

Figure A.4.6 Heatmaps of visited state across models 10 episodes after 4th reward

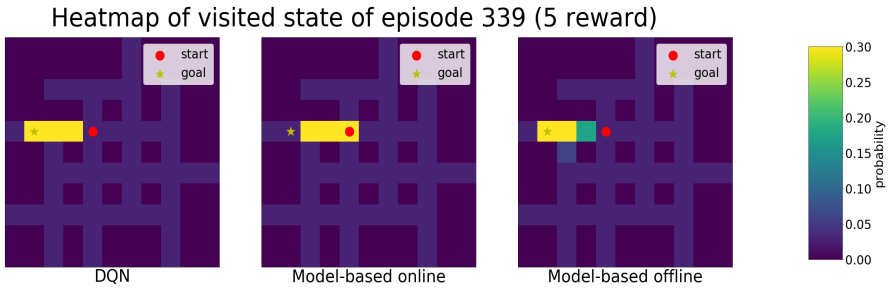

Figure A.4.7 Heatmaps of visited state across models 10 episodes after 5th reward

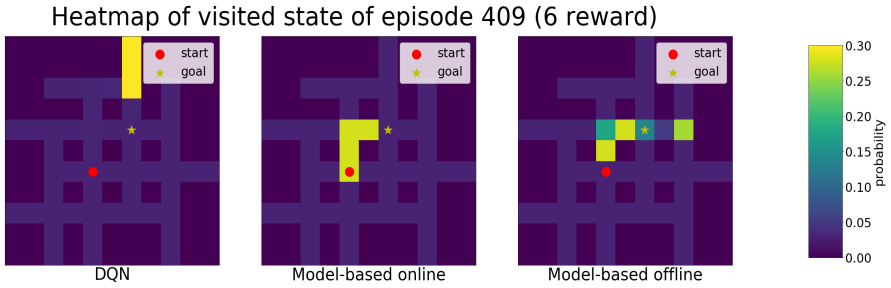

Figure A.4.8 Heatmaps of visited state across models 10 episodes after 6th reward

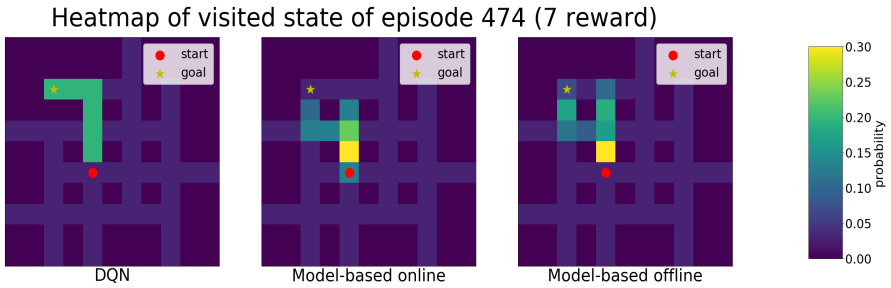

Figure A.4.9 Heatmaps of visited state across models 10 episodes after 7th reward

3. Internal Reward Function for the Explorative Behavior Showed no Significant Difference Between Different Log and Sigmoid Functions

We tested multiple internal reward functions shown in Figure A.4.10. We measured the exploration percentage and entropy of DQN agents using these reward functions. The results in Table A.2, and Table A.3 showed there is no significant difference in explorative behaviors using different internal functions.

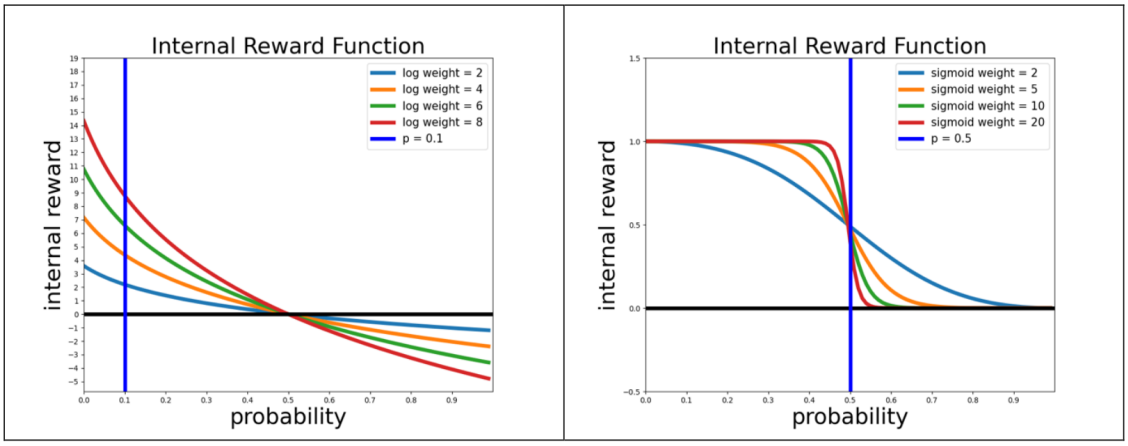

Figure A.4.10 Log and sigmoid internal reward functions with different weights

Table A.2 Exploration percentage mean and variance across agents with different internal reward functions

Exploration Percentage (%)	Log internal functions				Sigmoid internal functions			
Weight	2	4	6	8	2	5	10	20
Mean	89.21	90.31	90.37	90.36	90.31	90.35	90.33	90.33
Variance	30.35	.66	.32	.13	.52	.13	.43	.34

Table A.3 Exploration entropy mean and variance across agents with different internal reward functions

Exploration Entropy	Log internal functions				Sigmoid internal functions			
Weight	2	4	6	8	2	5	10	20
Mean	3.54	3.61	3.61	3.61	3.61	3.61	3.61	3.61
Variance	0.069	.005	.004	.005	.005	.005	.005	.005

4. Heat Map of the Curiosity-based Agent in All Reward Location Changes (reward 2 ~7)

The Figures below shows heatmaps of the curiosity-based agent of explore-exploit policies in both online(real) and offline(simulated) environment. (one episode before reward changes and after are shown for comparison)

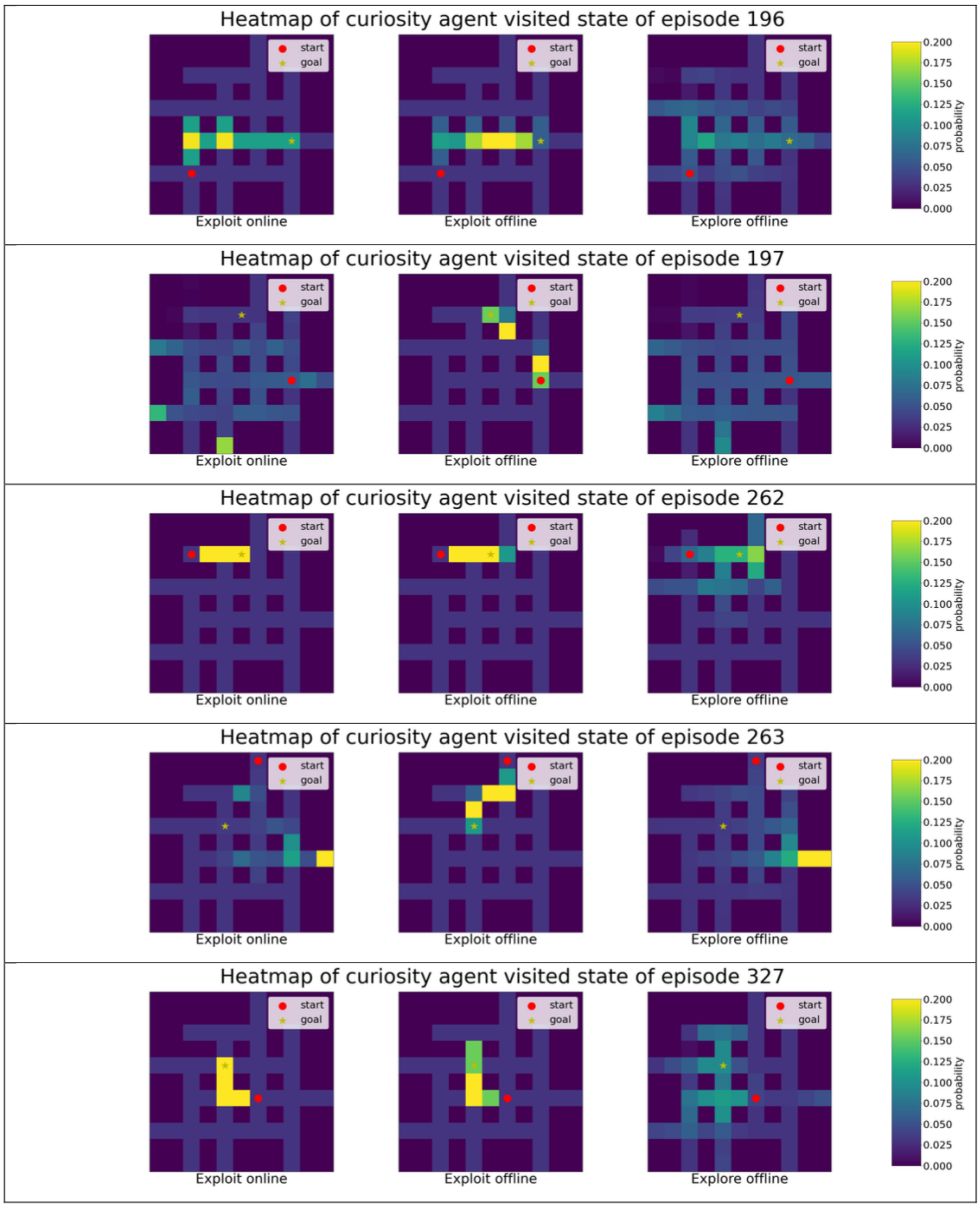

Figure A.4.11 Heatmap of curiosity-based agent on reward number 2 ~ 4

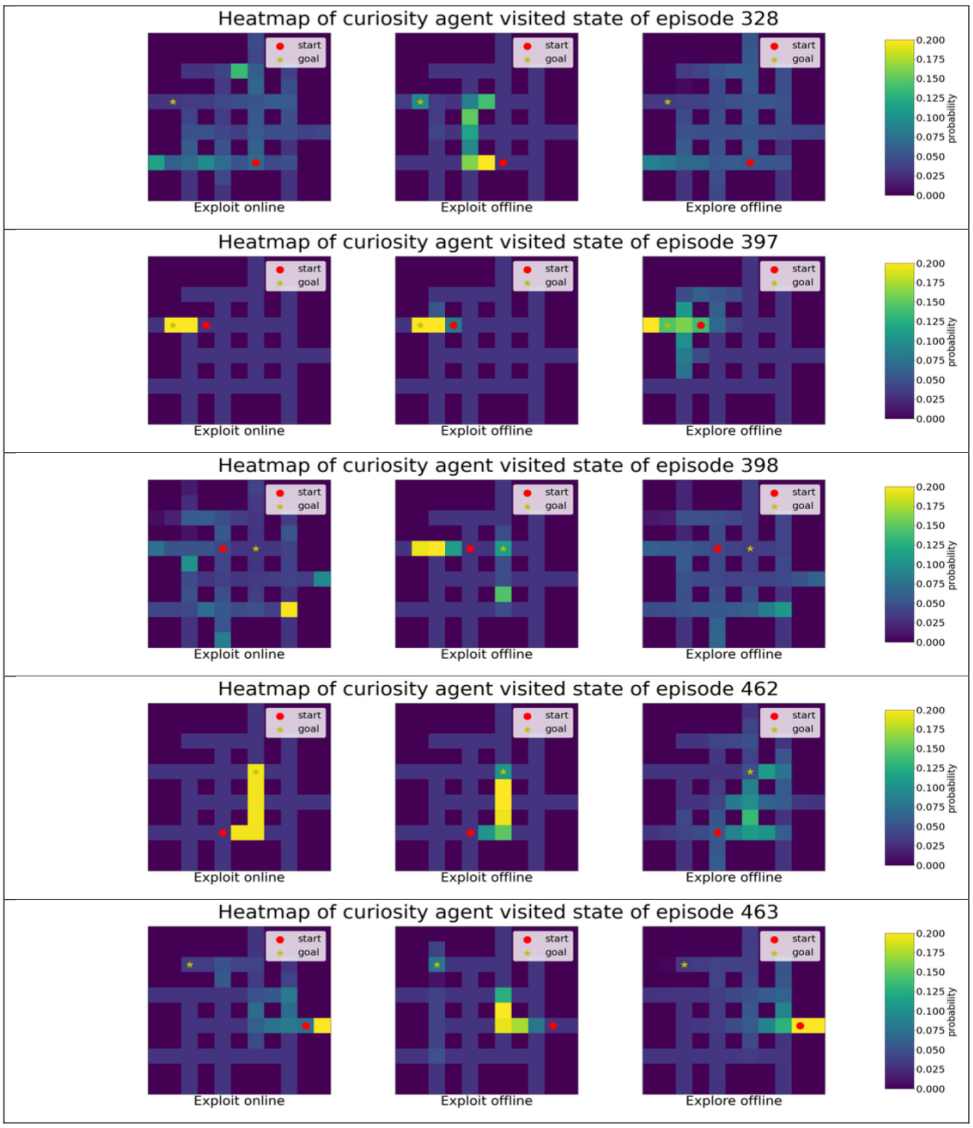

Figure A.4.12 Heatmap of curiosity-based agent on reward number 5 ~ 7

References

Groman, S. M., Lee, D., & Taylor, J. R. (2021). Unlocking the reinforcement-learning circuits of the orbitofrontal cortex. Behavioral Neuroscience, 135(2), 120–128.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., & Michalewski, H. (2019). Model-Based Reinforcement Learning for Atari. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1903.00374

Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning. In arXiv [stat.ML]. arXiv. https://github.com/Ardavans/DSR

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., & Finn, C. (2018). Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1803.11347

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., & Song, D. (2018). Assessing Generalization in Deep Reinforcement Learning. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1810.12282

Padakandla, S. (2020). A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/2005.10619

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven Exploration by Self-supervised Prediction. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1705.05363

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419), 1140–1144.

Yu, Y. (2018, July). Towards sample efficient reinforcement learning. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. Twenty-Seventh International Joint Conference on Artificial Intelligence {IJCAI-18}, Stockholm, Sweden. https://doi.org/10.24963/ijcai.2018/820

Addicott, M. A., Pearson, J. M., Sweitzer, M. M., Barack, D. L., & Platt, M. L. (2017). A Primer on Foraging and the Explore/Exploit Trade-Off for Psychiatry Research. Neuropsychopharmacology: Official Publication of the American College of Neuropsychopharmacology, 42(10), 1931–1939.

Behrens, T. E. J., Woolrich, M. W., Walton, M. E., & Rushworth, M. F. S. (2007). Learning the value of information in an uncertain world. Nature Neuroscience, 10(9), 1214–1221.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019). Reinforcement Learning, Fast and Slow. Trends in Cognitive Sciences, 23(5), 408–422.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., & Michalewski, H. (2019). Model-Based Reinforcement Learning for Atari. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1903.00374

Knox, W. B., Otto, A. R., Stone, P., & Love, B. C. (2011). The nature of belief-directed exploratory choice in human decision-making. Frontiers in Psychology, 2, 398.

Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning. In arXiv [stat.ML]. arXiv. https://github.com/Ardavans/DSR

Ladosz, P., Weng, L., Kim, M., & Oh, H. (2022). Exploration in deep reinforcement learning: A survey. An International Journal on Information Fusion, 85, 1–22.

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch Reinforcement Learning. In M. Wiering & M. van Otterlo (Eds.),

Reinforcement Learning: State-of-the-Art (pp. 45–73). Springer Berlin Heidelberg.

Li, M., Zhao, X., Lee, J. H., Weber, C., & Wermter, S. (2023). Internally Rewarded Reinforcement Learning. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/2302.00270

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1312.5602

Padakandla, S., Prabuchandran, K. J., & Bhatnagar, S. (2019). Reinforcement Learning in Non-Stationary Environments. In arXiv [cs.LG]. arXiv. https://doi.org/10.1007/s10489-018-1296-x

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning, second edition: An Introduction. MIT Press.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3), 279–292.

Addicott, M. A., Pearson, J. M., Sweitzer, M. M., Barack, D. L., & Platt, M. L. (2017). A Primer on Foraging and the Explore/Exploit Trade-Off for Psychiatry Research. Neuropsychopharmacology: Official Publication of the American College of Neuropsychopharmacology, 42(10), 1931–1939.

Behrens, T. E. J., Woolrich, M. W., Walton, M. E., & Rushworth, M. F. S. (2007). Learning the value of information in an uncertain world. Nature Neuroscience, 10(9), 1214–1221.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019). Reinforcement Learning, Fast and Slow. Trends in Cognitive Sciences, 23(5), 408–422.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., & Michalewski, H. (2019). Model-Based Reinforcement Learning for Atari. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1903.00374

Knox, W. B., Otto, A. R., Stone, P., & Love, B. C. (2011). The nature of belief-directed exploratory choice in human decision-making. Frontiers in Psychology, 2, 398.

Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning. In arXiv [stat.ML]. arXiv. https://github.com/Ardavans/DSR

Ladosz, P., Weng, L., Kim, M., & Oh, H. (2022). Exploration in deep reinforcement learning: A survey. An International Journal on Information Fusion, 85, 1–22.

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch Reinforcement Learning. In M. Wiering & M. van Otterlo (Eds.), Reinforcement Learning: State-of-the-Art (pp. 45–73). Springer Berlin Heidelberg.

Li, M., Zhao, X., Lee, J. H., Weber, C., & Wermter, S. (2023). Internally Rewarded Reinforcement Learning. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/2302.00270

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. In arXiv [cs.LG]. arXiv. http://arxiv.org/abs/1312.5602

Padakandla, S., Prabuchandran, K. J., & Bhatnagar, S. (2019). Reinforcement Learning in Non-Stationary Environments. In arXiv [cs.LG]. arXiv. https://doi.org/10.1007/s10489-018-1296-x

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning, second edition: An Introduction. MIT Press.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3), 279–292.

Rolnick, David, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. n.d. "Experience Replay for Continual Learning." Accessed November 13, 2023. https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Bellman, Richard. 1957. "A Markovian Decision Process." Journal of Mathematics and Mechanics 6 (5): 679–84.

Pathak, Deepak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. "Curiosity-Driven Exploration by Self-Supervised Prediction." ArXiv [Cs.LG]. arXiv. http://arxiv.org/abs/1705.05363.

Howard, James D., and Thorsten Kahnt. 2021. "To Be Specific: The Role of Orbitofrontal Cortex in Signaling Reward Identity." Behavioral Neuroscience 135 (2): 210–17.

Oyama, Kei, Kei Majima, Yuji Nagai, Yukiko Hori, Toshiyuki Hirabayashi, Mark A. G. Eldridge, Koki Mimura, et al. 2023. "Distinct Roles of Monkey OFC-Subcortical Pathways in Adaptive Behavior." BioRxiv. https://doi.org/10.1101/2023.11.17.567492.

Schoenbaum, Geoffrey, Matthew R. Roesch, Thomas A. Stalnaker, and Yuji K. Takahashi. 2009. "A New Perspective on the Role of the Orbitofrontal Cortex in Adaptive Behaviour." Nature Reviews. Neuroscience 10 (12): 885–92.

Stalnaker, Thomas A., Tzu-Lan Liu, Yuji K. Takahashi, and Geoffrey Schoenbaum. 2018. "Orbitofrontal Neurons Signal Reward Predictions, Not Reward Prediction Errors." Neurobiology of Learning and Memory 153 (Pt B): 137–43.