DEPARTMENT OF ELECTRICAL AND

COMPUTER ENGINEERING

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Electrical and Computer Engineering

A Study of Habituation Failures and Rehabituation Process: A Reinforcement Learning Experiment

Umut Ekin Gezer

DEPARTMENT OF ELECTRICAL AND

COMPUTER ENGINEERING

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Electrical and Computer Engineering

A Study of Habituation Failures and Rehabituation Process: A Reinforcement Learning Experiment

Author:	Umut Ekin Gezer
Supervisor:	Prof. Dr.-Ing. habil. Alois C. Knoll
Advisors:	Berkay Demirel, M.Sc. (Universitat Pompeu Fabra)
	Josip Josifovski, M.Sc. (Technical University of Munich)
Submission Date:	October 19, 2022 in Garching

I confirm that this bachelor's thesis in electrical and computer engineering is my own work and I have documented all sources and material used.

Munich, Submission date Umut Ekin Gezer

Acknowledgments

First and foremost, I have to thank my research supervisors, M.Sc. Berkay Demirel and M.Sc.Josip Josifovski. Without their assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. I would also like to show gratitude Martí Sànchez-Fibla professor at University Pompeu Fabra for giving me the opportunity to do research on this topic. I would also like to thank my dear friend Çağdaş Akşit for his support during this thesis process. Last, but not least, my warm and heartfelt thanks go to my family for their tremendous support they had given to me.

Abstract

In recent years, there has been increasing inter-disciplinary research combining artificial intelligence and neuropsychology on the human-decision making mechanisms. The neuropsychological findings indicate two distinctive human decision-making mechanisms: habitual and goal-directed behaviour. The arbitration between these two decision-making mechanisms has become a research interest in this field. The environmental changes resulting in habituation failures are one of the key signals for the necessity of goal-directed behaviour domination in human decision-making. Afterwards, rehabituation for the changes is another key signal of the redomination of habitual behaviour. Thanks to biological and psychological similarities in the structure of the reinforcement learning framework in artificial intelligence, this arbitration mechanism can be understood through these key signals. In this work, we investigate the arbitration between two mechanisms by modelling the habituation, habitation failures and rehabituation in a 3D version of a Hotel Elevator Row(HER) environment. We use model-free Deep Q-learning(DQN) and model-free and model-based in-between approach Successor Representation (SR) to model these key components. We compare our results to an identically designed human behavioural experiment to detect the biological plausibility of modelling rehabiutation and habituation failures with these reinforcement learning frameworks. As a result of this work, we conclude that Successor Representation provides a plausible alarm mechanism for model-free habitual behaviour against habituation failure. Finally, we detect similarities in rehabituation performances between human behavioural data and Successor Representation.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

According to neuropsychological findings, it has been hypothesized that human decision-making has two distinctive behavioural mechanisms: habitual and goal-directed behaviour [BD98]. Habitual behaviour is an automatic decision-making mechanism that selects actions rapidly with low demand for cognitive effort. Goal-directed behaviour has a flexible action selection system sensitive to environmental changes and demands great attention. The human decision-making mechanisms can plausibly be modelled with the reinforcement learning framework similar to standard reinforcement learning used to explain some aspects of operant learning and its underlying neural activity [SL14].

Habitual behaviour can be modelled with model-free (MF) reinforcement learning since model-free reinforcement learning caches long-term estimated actions and has Temporal Differences(TD)-error-based value learning similar to the mid-brain part's dopaminergic activity. The model-based (MB) reinforcement learning can model goal-directed behaviour due to its sensitivity to environmental changes and similarities between the human brain's cognitive mapping structure and model-based state transition matrix [DND05]. Moreover, the idea of modelling both decision-making systems with an in-between approach, Successor Representation (SR), has been argued. Since Successor Representation (SR) is explicitly represented in the human brain and combines the MF and MB properties in its structure [SBG17; Rus+17].

However, habitual and goal-directed behaviour have different properties, therefore, suitability for different tasks. During the control of habitual behaviour mechanism, when abrupt changes in the environment occur, the goal-directed behaviour needs to take control to deal with habituation failures which result from the insensitivity of habitual behaviour against changes. Therefore the necessity of an arbitration mechanism between these two mechanisms has been argued in recent years [A; BD98].

1.2 Goals and Objectives

In this thesis, we aim to further clarify the arbitration between two main human decision-making of habitual and goal-directed behaviour by modelling habituation, habitation failures due to environmental changes and the rehabiutation process for these changes using reinforcement learning framework. In addition, we intend to investigate the proposed alarm mechanism of Successor Representation-monitoring model-free behaviour [San20] against these changes in a 3D environment. Going one step further, clarifying this arbitration mechanism could shed new light on novel research methods for diseases caused by disfunctions in the habituation mechanisms, such as obsessive-compulsive disorder and addictions.

To model these properties, we redesign the 3D version of the Hotel Elevator Row(HER) environment (the original grid version is introduced in work [San20]). The aim of designing the 3D Hotel Elevator Row(HER) environment is to create habituation failures through the changes in the starting point of a single learning agent. The learning agent is trained with model-free reinforcement learning deep Q-learning (DQN) and Successor Representation(SR) for a room-finding task in 3D Hotel Elevator Row(HER) environment.

1.3 Overview

- **Chapter 2** provides background information for the human-decision making mechanisms and reinforcement learning frameworks. Afterwards, the relation between the two concepts is clarified.
- **Chapter 3** provides the summary of the five works [LSO14; I+; Rus+17; San20; NS22] which investigate underlying arbitration structure between habitual and goal-directed behaviours.
- **Chapter 4** provides 3D Hotel Elevator Row(HER) environment's design structure and parameters. The last section of chapter 3 introduces deep Q-learning (DQN) and Successor Representation(SR) and their hyperparameters for training.
- **Chapter 5** details the learning agent experiment setup. We provide the section "Experiment Results" results of the experiment and interpret the results in the section "Discussion".
- **Chapter 6** briefly summarises the work and provides future research ideas in this field.

2 Background

2.1 Dual Mechanism in the Human Decision-Making

The human brain has an adaptive decision-making system. This system selects a series of possible candidate actions for maximizing long-term future benefits in a complex and dynamic environment [VS19; DND05; I+]. It has been hypothesized that the human brain evaluates information from the environment and then computes an action as a response. Based on the environmental consequences of generated action, human brain reevaluates the environmental suitability of the following action in the next step. This mechanism enhances individuals' survival skills against environmental changes [BD98].

This evaluation system gives negative or positive feedback for applied actions. In some cases, humans behave in a novel environment without prior knowledge, merely depending on this feedback mechanism, called a trial-error system [San20]. Due to a lack of knowledge and a prior experience with the environment, a human takes action carefully and sensitively but slowly. In the novel task environment, humans consciously select their actions. This mode of behaviour in the decision-making system refers to deliberative behaviour. On the other hand, after habituating to the environment, the decision-making system rapidly distinguishes suitable actions to maximize the individual's benefit. Therefore, this system reflects habitually, fast, and in some cases unintentionally but being less sensitive to environmental changes [I+]. Due to these different modes of behaviour during habituation in an environment, the existence of two decision-making systems has been argued. The two distinctive mechanisms are identified as deliberative (goal-directed) and reflexive(habitual) systems.[LSO14; I+; VS19; BD98; San20; Daw18; Kah11; Gef18].

Several evidence has been found for these two distinct decision-making mechanisms in recent years. Psychologist Kahneman identifies [Kah11], these two mechanisms as System 1 and System 2. System 1 refers to an automatic and rapid decision-making mechanism with less effort or without any existence of voluntary intention. However,

System 2 decision mechanism is responsible for effortful decisions with a greater attentional mechanism. Furthermore, neural findings support that two primary systems might control instrumental behaviour. As mentioned, one is the habitual behavioural mechanism, and the second is the goal-directed mechanism [BD98], similar to two opposite mechanisms that are cold versus hot, deliberative versus automatic [Daw18], or solvers vs learners [Gef18].

The habitual behaviour in the human decision-making system represents the rapid selection of action with less cognitive effort. This behaviour is achieved through habituation, long-term experience and sufficient knowledge of the environment, which is based on a trial and error mechanism without generating an internal environment model of the environment [LSO14]. Nevertheless, a habitual decision-making instrument is insensitive to environmental alterations, and rewards [VS19]. Briefly, this mechanism is straightforward and related to intuitive behaviour. However, this behaviour is generally suitable only for expected events[NS22].

On the other hand, goal-directed behaviour in the decision-making system flexibly selects the actions and is sensitive to the changes in the environment. Nevertheless, this behaviour is not as reflexive as habitual behaviour. The goal-directed mechanism is appropriate for generating more sensitive actions against unexpected changes in the environment [Rus+17], but this system is cognitively effortful and demands higher attention [NS22]. In general, this system models the internal model of the environment by constructing a cognitive map [SBG17; VS19].

The existence of the two main action selection systems in the human brain raises the question of how the human brain manages arbitration between habitual and goal-directed mechanisms and how the two systems interact [LSO14]. According to Daw [Daw18], the arbitration between habitual and goal-directed behaviours can be modelled with Plato's famous Chariot Allegory. There is a chariot with a pair of horses in which the first horse is noble and the second is beastly. The beastly horse is fast and better in its habituated environment and task but unable for domestication, therefore unreliable for adaptation to environmental changes. The noble horse can easily be domesticated, meaning that it can deal with the changes in the environment, despite being slower than the beastly horse. Now, one could ask, "How does a charioteer work with two different horses?" this refers to a controversial question in the human decision-making system which relies on the degree of one of these systems in a complex and dynamic environment.

According to the psychological findings about the arbitration between habitual and goal-directed behaviours [A; BD98], it has been hypothesized that habitual behaviour might be the default behaviour. The goal-directed system takes control when abrupt environmental changes happen. Furthermore, the computational modellings of the arbitration mechanism with reinforcement learning models suggest the evidence that habitual behaviour is the initial behaviour, and the alarm system exists between habitual and goal-directed behaviours based on increased demands on attention due to habituation failure. When habituation failure occurs, the alarm signal triggers to change from habitual behaviour to goal-directed behaviour [San20; LSO14].

Several scientific work have been published [VS19; SBG17; DND05; I+; NS22; San20; Rus+17; LSO14], which focus on arbitration between two distinct mechanisms in the human decision-making . The reinforcement learning framework is utilized in this research to explain this arbitration. There are two main reasons behind employing reinforcement learning models. The first reason is that reinforcement learning ensures a more biologically plausible structure than other machine learning models since it provides testable assumptions about human behaviour and corresponding neural implementation of structure learning [Ten+11]. Secondly, predictive and compact illustration of structures of the neural system can be achieved through the reinforcement learning principles.[I+].

The next section provides more detailed information about the reinforcement learning structure to relate two decision-making systems and their arbitration mechanism with reinforcement learning models. In addition, the biological plausibility of reinforcement learning models; Model-Free RL, Model-Based RL and the in-between approach: Successor Representation, are provided.

2.2 Reinforcement Learning

In this section, the basic concepts of reinforcement learning are summarized from the book **"Reinforcement Learning: An Introduction"** [SB] by Richard S. Sutton and Andrew G. Barto.

2.2.1 Main Concepts in Reinforcement Learning

In reinforcement learning, the learning agent receives a particular signal value from the environment called a reward at each time step. The goal of the learning agent is to maximize obtained cumulative reward in the long run, not receiving the immediate reward for a particular step. The agent in a reinforcement learning environment continuously interacts with the environment. The particular level of sensation of the learning agent regarding the environment influences its actions with feedback as a reward signal of the reached states after the agent has taken action. Behind the formalization of the RL, ideas from dynamic systems theory are utilized primarily as the optimal control of the partially known Markov decision process is predominantly employed.

Exploration-Exploitation Trade-Off

In reinforcement learning, there are some challenges exist. One of the main dilemmas is **exploration and exploitation**. A reinforcement learning agent selects actions from its action domain to achieve maximum reward effectively. If the agent has already had information about some rewards in the environment, an agent might tend to concentrate on explored rewards. However, the goal of the learning agent is maximizing the cumulative reward; to obtain greater rewards, the learning agent needs to discover the states. The agent has to **exploit** based on its experience to maximize cumulative reward. At the same time, it has to **explore** to select better actions in the future to achieve its goal.

Correspondence with Psychology and Neuroscience

Reinforcement learning can be employed to explain several concepts in psychology and neuroscience since reinforcement learning has the closest learning structure that humans and other animals have. Biological learning systems inspire especially many main reinforcement learning algorithms. For instance, it has been hypothesized that the "reward prediction error hypothesis of dopamine neuron activity" can be modelled with Temporal Differences Learning in reinforcement learning. [SB]

Main Elements of Reinforcement Learning

Reward Signal (r): The reward signal is an information signal due to interaction between the environment and reinforcement learning agent. The environment sends a reward signal to the reinforcement learning agent at each time step. The reward signal informs the agent about how rewarding is the reached state. Through this feedback mechanism, the learning agent can detect the maximum reward-given state from the reward signal.

Value Function (V or Q): The value function is the expected total amount of reward starting from a state which accumulates over the future. A reward suggests the immediate desirability of the given states. However, the value function refers to the long-term desirability of the given states. The long-term desirability of states is detected by the available rewards that are collected in the following states to a given state.

Policy (π): Policy is described as a learning agent's way of behaving at a specified time. Policy maps the actions that are taken by the agent in the perceived states in an environment. Thus, policy is the core of the reinforcement learning agent.

Model: A model of the environment enables predictions for how the environment will behave. A model can predict subsequent states and rewards under a given current state and action. Thanks to having the model of an environment, **planning** can be done by considering possible future situations without visiting them. The subclass of reinforcement learning **model-based** methods exploits the model and make a plan to solve a reinforcement learning problem, whereas **model-free** methods learn from trial and error and do not use the model of an environment.

Markov Decision Process(MDP)

This subsection is summarized using the sources [SB; Bel57; San20]. Markov Decision Process (MDP) provides a useful framework for modelling reinforcement learning problems. The MDP model denotes possible reachable states in the environment and the influence of the actions on those states. MDP is formalized with the tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ denotes the **state space**, $\mathcal{A}$ denotes the **action space**, $P = P(s' \mid s, a)$ denotes the **transition probability function**, defines as $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, fulfills $\sum_{s'} P(s' \mid s, a) = 1$ and, describes path to the next state s' from the state s by taking action a. $R = R(s, a)$ denotes the **reward function** for each state and action, defines as S $\times \mathcal{A} \rightarrow \mathbb{R}$. γ is the **discount factor** for future rewards, which ranges between 0 and 1, $\gamma \in (0, 1)$.

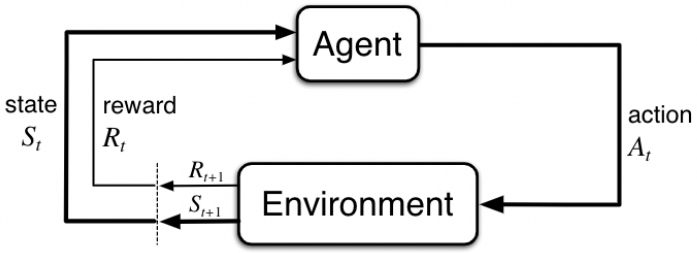

Figure 2.1: The agent-environment interaction in a Markov decision process [SB]

The MDP formalized reinforcement learning problem is a sequential decision-making problem in a discrete-time stochastic process. At the beginning of the process agent starts from an **initial state** $s_0 \in \mathcal{S}$,then agent in the environment takes an **action** $a_t \in \mathcal{A}$ based on its policy $\pi(a_t \mid s_t)$ at each time step t. On account of taken action by following policy at each time step, agent transits the **next state** $s_{t+1} \in \mathcal{S}$ determined by state transition probability function $P\left(s_{t+1} \mid s_t, a_t\right)$ and as a feedback , agent receives a **reward** $r_t \in R$ based on the reward function $P\left(r_{t+1} \mid s_t, a_t\right)$. Figure 2.1 demonstrates the explained mechanism of the Markov Decision Process.

As already mentioned in the subsection 2.2.1, a policy π is one of the main element in RL, that maps states to actions $\pi : \mathcal{S} \rightarrow \mathcal{A}$. Therefore, a policy specifies how the agent acts in the current state s_t. A policy depends on the agent's current state and does not depend on time. A policy can be represented in two form, either deterministic policy $\pi(s)$ or stochastic policy $\pi(a \mid s) = P\left[A_t = a \mid S_t = s\right]$.

A reward signal r represents one-step feedback given in both states. However, in **episodic environments**, interactions between the agent and environment repeatedly occur as the agent steps into the environment each time. Thus, all weighted rewards, each reward is multiplied by the discount factor $\gamma \in (0, 1)$, are aggregated and returns into the **discounted return** R_t, which is demonstrated in equation 2.1. In practice, discounting every future reward with a discount factor ensures the balance between an agent's long-term and short-term goals.

$$R_t = \sum_{k}^{\infty} \gamma^k r_{t+k} \tag{2.1}$$

Value Functions and Policies

Above in the subsection 2.2.1, the one of RL concept **value function** is already introduced. In general, most reinforcement learning algorithms involve estimating value function property. The value function has two different types, which are value function V and state-value function(q-value function) Q.

The value function $V_\pi(s)$ has the **input of states**, which estimates the quality of being in the given state in terms of the last subsection defined 2.1 expected return. However, the state-value function $Q_\pi(s, a)$ takes as **input state and action pairs** to estimate how good to take the given action in the given state in terms of its expected return. Briefly, both value functions are related to the value of expected future rewards. Inherently, the term expected reward of an agent corresponds to the actions and specifies a way of acting of an agent similar to the policy [SB].

The state-value function $V_\pi(s)$, denotes the expected return starting from given a state s by acting under a policy π in any time step t.

$$V_\pi(s) = \mathbb{E}_\pi \left[R_t \mid S_t = s \right] \tag{2.2}$$

Similarly, the action-value function(q-value function) $Q_\pi(s, a)$ represents the value of initializing a state s by taking action a by following a policy π.

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[R_t \mid S_t = s, A_t = a \right] \tag{2.3}$$

Bellman Equation

Formalizing both the discounted return given in equation 2.1 and the state value function equation 2.2, an equation is obtained, which is called the Bellman equation. The Bellman equation describes the correspondence between the value of a state, and its successor states [Bel57].

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s', r} P\left(s', r \mid s, a\right) \left(r + \gamma V_\pi\left(s'\right) \right) \tag{2.4}$$

Optimal Value Functions and Optimal Policies

Finding an **optimal policy** π^* defines through detecting a policy with the greatest cumulative reward over the long run. At the same time, optimal policy refers to finding

a proper solution for a reinforcement learning problem. The agent's goal is to detect the best action series, which maximizes reward in the long term [SB].

$$\pi^* = \arg\max_{\pi} V_\pi(s) = \arg\max_{\pi} Q_\pi(s,a) \tag{2.5}$$

Furthermore, since the value function describes partial order over the policies, optimal policies for an RL task share a similar state-value function and action-value function. The optimal policy owned value functions are defined as respectively **optimal state-value function** $V^*(s)$ and **optimal action-value function** $Q^*(s,a)$ [SB].

Bellman Optimality Equation

The Bellman equation that is given in 2.4 for optimal state-value function $V^*(s)$ is expressed in the equation 2.6.This equation is defined as Bellman Optimality Equation. It expresses the condition in which the value of a state by having an optimal policy is equivalent to that state's expected to return for the best action.

$$V^*(s) = \max_{\pi} V_\pi(s) = \max_{a} Q_{\pi^*}(s,a) = \max_{a} \sum_{s',r} P\left(s', r \mid s, a\right)\left(r + \gamma V^*\left(s'\right)\right) \tag{2.6}$$

Analogously, the Bellman optimality equation for the optimal state value function $Q^*(s,a)$ is expressed in the 2.7 below.

$$Q^*(s,a) = \max_{a} \sum_{s',r} P\left(s', r \mid s, a\right)\left(r + \gamma Q^*\left(s', a'\right)\right) \tag{2.7}$$

As a result, the optimal policy can be directly detected if the state-value and action-value functions are known. The policies which maximize the Bellman optimality equations are determined as optimal policies. Nevertheless, finding an optimal policy for the high-dimensional MDP problem could be challenging in terms of computational complexity [SB]. Therefore we introduce in below "Temporal-Difference Learning".

Temporal-Difference Learning

Due to the difficulty of solving high dimensional RL problems with the previous subsection introduced Bellman equation, iterative methods such as dynamic programming, Monte Carlo and temporal-difference(TD) learning could be employed [SB]. In this subsection, we focus on TD learning to explain the TD-Error property in Q-learning.

The TD learning method is an iterative, online and model-free method [SB]. The TD learning method is a bootstrap method which means updating an estimation based on previous estimations given in 2.8.

$$V_\pi(s) = V_\pi(s) + \alpha \left(r_{t+1} + \gamma V_\pi \left(s_{t+1}\right) - V\left(s_t\right)\right) \tag{2.8}$$

The parameter α is the learning rate which is a small positive value between 0 and 1. TD method updates the estimate of the value function based on TD-error $(r_{t+1} + \gamma V_\pi \left(s_{t+1}\right) - V\left(s_t\right))$, which implies the difference between reward in current time and predicted reward in a given state [SB].

2.2.2 Q-Learning Model-Free Reinforcement Learning

Q-learning is a model-free and off-policy reinforcement learning method. This method detects the optimal action-value function without depending on a policy. It learns by estimating the action-value function, which stores future discounted rewards. While updating the action-value function, Q-learning does not make a model of the environment. A Model-free Q-learning method detects an optimal policy indirectly through finding the highest future discounted action-values $\pi^*(s) = argmax_a Q(s,a)$ [Wat89].

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right] \tag{2.9}$$

The Q-learning algorithm updates the action value in each step through the equation 2.9. In this equation, $(R + \gamma \max_{a'} Q\left(S', a'\right) - Q(S,A))$ is defined as TD-Error in Q-learning. TD-Error in Q-learning specifies to the **reward expectancy signal**. As an example, after transition from state s_t to s_{t+1} by applying action a_t, the learning agent receives a reward signal r_t. If the $TD^Q_{error} = R + \gamma \max_{a'} Q\left(S', a'\right) - Q(S,A)$ is close to 0, it means that the expected reward is obtained. Additionally, a negative TD-Error is interpreted as an overestimation of the expected reward, whereas a positive TD-Error is evaluated as an underestimation.

ε-greedy Action Selection Method

Q-learning selects actions with the ε-greedy method. To compute an action, this algorithm exploits the estimated action-value function with a probability $1 - \varepsilon$; however, with probability ε, the agent selects a random action to discover higher rewards. This balance of random action and computing an action by exploiting the action-value function is called an **exploration-exploitation trade-off** which is mentioned in the subsection "Main Concepts in Reinforcement Learning" 2.2.1. Generally, ε is initialized with 1 and decays linearly or exponentially in each episode (ε decays to 0 or close to 0)—this decay in ε guarantees the balance between exploring new higher rewards and exploiting available rewards.

The Model-Free Q-Learning corresponds to habitual behaviour. Similarly, Q-learning does not create a model of an environment. Furthermore, Q-learning relies on caching values from the pre-computed table during updating action value. Due to this property, Q-Learning computes actions rapidly and automatically like habitual decision mechanisms [San20].

Model-Free Reinforcement Learning and Habitual Behavior

In the human brain, the phasic response of dopamine neurons in the dorsolateral striatal is responsible for action selection. This mid-brain part's dopaminergic activity corresponds to habitual behaviour according to based fMRI results in neurophysiological findings [DND05]. The action selection structure of mid-brain dopamine neurons demonstrates a similar structure to TD-error-based action selection in the MF reinforcement learning method. Both systems' decisions rely on prediction error [Rus+17].

Furthermore, the neurological investigations [LC06; LG08] verify that the action selection mechanism in the mid-brain part is based on caching long-term estimated actions by firing medium spiny neurons. The caching of fully computed long-run action values in this system show similarities with caching value functions V and Q in the MF reinforcement learning algorithm and interim storing steps of value computation. For this reason, habitual behaviour and MF mechanisms reflexively compute an action. Moreover, habitual behaviour has an action computation mechanism that demands less cognitive effort, like less computational complexity of MF learning[I+; Rus+17; Daw18].

The habitual behaviour mechanism learns directly from experience without creating an internal model of an environment. This system does not have any information about the relationship between various states. Similarly, MF-learning is not aware of changes in the environment [I+]. Thus, this system fails in revaluation tasks such as latent learning and rewards revaluations [Rus+17].

2.2.3 Model-Based Reinforcement Learning

Model-based(MB) reinforcement algorithms form the model of an environment to estimate environmental responses to maximize cumulative reward. The model of an environment computes an estimation for the following states and rewards by storing possible transition probabilities and reward functions in a matrix.

MF methods rely on **learning**. On the other hand, the main property of MB algorithms is **planning**. In the RL literature, "planning" refers to a computational operation that improves a policy through interaction with the modelled environment by taking a model as input. Both learning and planning methods are based on backing-up update computations to estimate value functions. However, learning employs **real experience** while planning uses **simulated experiences** built on an environment model [SB].

In the RL literature, there are two main planning methods: state-space planning and plan-space planning. In our work, the term planning refers to the **state-space planning**. In state-space planning methods, the model of an environment simulates an experience in the space of states by having possible transitions between the states and rewards. These computations are stored in the state transition matrix $T(s, s_t)$ and the reward matrix R. After that, the model updates state estimation values. Finally, updated value functions improve the policy accordingly. Briefly, the Model-based method iterates policy through improvements on the environmental model [SB].

Solving a small dimensional reinforcement learning problem by modelling and planning might be the most efficient approach. However, solving a high-dimensional problem through planning and modelling poses hard in tackling computational complexity. The main reason is that the MB method utilizes computationally highly storage demanded wide and deep tree search method while updating the environmental model in each step[SB].

Dyna-Q

The Dyna-Q is a model-based method which is built on Q-learning. This method contains four main stages: planning, acting, model learning, and direct RL. The random sample Q-learning method is used for planning. In the planning phase, an updated model improves policy. At the same time, a policy is updated through real experience based on an interaction between the agent and environment using one-step tabular Q-learning. This update is specified as direct RL. Similar to updates in policy after an interaction, each experience in the environment updates the model of the environment. Finally, the learning agent acts according to updated policy in the environment. The diagram 2.2 explains the whole structure of Dyna-Q model-based learning. Briefly, Dyna-Q has a hybrid structure, which uses both real experiences for learning and simulation experiences for planning. Using one-step tabular Q Learning to learn from experience allocates fewer computational resources, whereas planning through the model is computationally intensive.

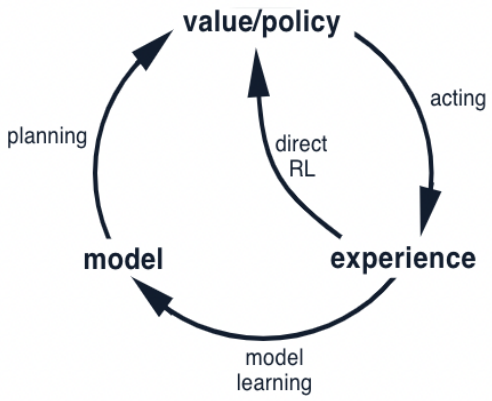

Figure 2.2: Dyna-Q [SB]

Model-Based Reinforcement Learning and Goal-Directed Behavior

The Model-Based reinforcement learning method stores a complete model of an environment. In each step, the model is updated through the improvements on the state transition matrix T and the reward matrix R. Because of this property, MB methods are sensitive to state changes and rewards placement in the environment. Formally

described, the MB method is sensitive to the state and reward revaluation tasks [I+]. Similarly, goal-directed behaviour in the human decision-making system stores an internal map of the environment. Thus, this behaviour is a convenient behaviour against instant changes [LSO14].

The prefrontal cortex system in the human brain is responsible for cognitive action planning. This mechanism chains short-term estimations regarding the immediate outcomes of each sequential action. Due to the branching structure in the prefrontal cortex(similar to tree-search), goal-directed behaviour can estimate the consequences of possible situations. This characteristic is costly, resembling the computationally expensive tree searches in the MB method. Similar to the computational complexity of MB methods, goal-directed behaviour is a cognitively effortful decision-making system [DND05].

Moreover, the cognitive map of the human brain uses offline mental simulations. Due to computed simulations through cognitive mapping, planning can be done before the action is taken [I+]. Similarly, the Dyna structure in the MB method simulates an experience on the environment model to improve state estimation values [Rus+17].

2.2.4 Successor Representation

We introduced the main characteristics of MF and MB methods in previous sections 2.2.2 and 2.2.3. The MF method updates the value function directly from the reward. It updates its policy accordingly, whereas the MB method learns through modelling an environment by storing the environmental parameters in state transition matrix T and reward function R. This section shows a hybrid method, the "Successor Representation Method", which is formalized by Peter Dayan [Day]. The successor representation (SR) includes both dynamics of MF and MB methods.

$$V(s_t) = R(s_t) \cdot M(s_t, s') \tag{2.10}$$

In successor representation(SR), value function V depends on future expected discounted states matrix M and immediate reward R. In the given equation 2.10, M is a relaxed version of the MB state transition matrix T. MB has a complete and one-step model for every environmental transition probability. However, the SR maps expected future occupancy of multi-step successor states. The term expected future occupancy of

the states refers to how likely the states s' from a given state s_t to be visited. Thus, SR is a "Semi Model Based Method".This relaxed feature of SR decreases the computational complexity of SR against high-dimensional RL problems, which is an advantage of SR against MB methods[Jul19].

$$M_{t+1}\left(s_t, s'\right) \leftarrow M_t\left(s_t, s'\right) + \alpha\left[\mathbb{I}\left(s_t = s'\right) + \gamma M_{t+1}\left(s_{t+1}, s'\right) - M_t\left(s_t, s'\right)\right] \tag{2.11}$$

On the other hand, the expected discounted future states matrix M learns through TD-learning, similar to Q-learning. In equation 2.11, the part $\mathbb{I}\left(s_t = s'\right) + \gamma M_{t+1}\left(s_{t+1}, S'\right) - M_t\left(s_t, s'\right)$ is the TD-Error component of SR. In SR, TD learning is based on the inconsistency between a given state and expected state occupancy. For instance, if TD-Error of SR has positive value, it indicates that matrix M underestimates the expected transition probability from state s_t to state s_{t+1}. This situation triggers the model to enhance the expected transition probability between two states s_t and s_{t+1}. Moreover, $TD - Error = 0$ indicates that the transition between two states is an expected transition [San20]. As a result, M is obtained by TD-learning, whereas TD-Error is the state prediction error(in Q-learning, TD-Error is reward prediction error) [VS19].

The SR combines the main advantages of MF and MB methods. SR depends on the policy and environmental model as future occupancy states matrix M updates through interaction with an environment. Unlike MB's complete state transition matrix T, the SR has the matrix M that contains only future expected state transitions thanks to its TD structure. For this reason, the SR can solve revaluation task problems in a shorter duration. In addition, since SR caches long-run estimation about the expected states, the habituation might be modelled with SR. Moreover, due to this structure, the computational complexity of SR might be decreased to the complexity level of MF methods (depends on task and task environment) [I+].

Successor Representation and Modelling Human Decision-Making Mechanisms

The previous sections give several biological plausible reasons that the model-free reinforcement learning framework can model habitual behaviour. In addition, the model-based learning framework provides an appropriate representation for goal-directed behaviour modelling. Due to its insensitive characteristic, the MF method might fail in a dynamic environment against rapid changes in reward and state. We specify this failure as a habituation failure. The MB method can flexibly deal with changes in the environment. However, MB behaviour switches to MF behaviour after

habituating to changes in the task. For this reason, arbitration is required between MF and MB systems.

Several ideas are put forth on dealing with the arbitration task with alterations in reward in a state in the task environment by using both models. The paper [LSO14] suggests the existence of an alarm mechanism for arbitration between two models. Therefore, this paper will be revisited in the following "Related Work" chapter. On the contrary, several studies in the field also supported that goal-directed and habitual behaviours should be modelled with one reinforcement learning algorithm, which is the Successor Representation method [SBG17; Rus+17; VS19].

The SR bypasses significant properties of MB and MF methods by having both significant modelling characteristics of habitual and goal-directed behaviour. SR can cache long-run predictions based on its experience and make multi-step estimations more cost-effectively. SR functionality in the human brain might agree on both deliberation and automaticity. Therefore it has been hypothesized that SR is the biological reinforcement learning algorithm[I+].

First, the SR is explicitly represented in the human brain. The place in the human brain SR is defined over, which possesses many characteristics of rodent hippocampal place cells. The rodent hippocampal place cells in humans are responsible for abstract stimulus structure [SBG14]. The SR represented place in the human brain covers fMRI pattern similarity with hippocampal and prefrontal areas[Sch+13]. In addition, SR is represented in the brain place, where plenty of memory effects are explained in the Temporal Context Model of memory [Ger+12]. Secondly, SR owns a TD-learning system that plausibly explains the TD learning theory of dopamine response, like MF learning. If the expected states in matrix M are visited, SR learns from reward function R. Because of this feature in SR, many dopamine-affected behavioural patterns of MB and MF might be explained with SR [Daw+11]. Thirdly, there has been empirical evidence suggesting the SR account part in the brain can solve reward revaluation tasks such as reward devaluation, latent learning, and sensory preconditioning similarly to the MB represented brain parts[I+].

As a result, the SR algorithm employs TD-learning stages similar to dopaminergic-striatal circuitry to cache long-term value function. At the same time, the SR uses hippocampal's model-based strategy for the reward revaluation task. For this reason, we hypothesize that habituation can be modelled with SR due to its TD-learning structure. Moreover, due to the sensitivity of SR against the changes in the environment, this algorithm can deal with habituation failure.

3 Related Work

3.1 Neural computations underlying arbitration between model-based and model-free learning

Several investigations and neurological evidence have shown that the prefrontal and anterior striatum parts in the human brain are responsible for a goal-directed system, and the posterior lateral striatum corresponds to habitual control [BD98]. Both areas in the human brain are responsible for action selection tasks. However, the systems are specific for different cases. Thus, human needs to make changes between the two systems for task suitability. Lee et al. [LSO14] investigate how control passes from one mechanism to the other. In addition, this work examines the interaction between two systems. This research[LSO14] hypothesizes an existence of an arbitration mechanism in the human brain that evaluates the dual system's performance. Moreover, this arbitrator has a level of control over the arbitration between the two systems.

The human experiment in this work [LSO14] consists of a two-choice Markov decision task to examine the hypothesized arbitrator theory. In this selection experiment, there are two tasks: specific and flexible. The specific task has properties of goal-directed behaviour, which generates state prediction error (SPE). This task is controlled by the model-based FORWARD learning and BACKWARD planning methods. Secondly, flexible tasks have identical properties to habitual behaviour, generating reward prediction error(RPE). This task is computationally supervised by the MF SARSA method. An arbitration system based on the Bayesian control model controls arbitration between MF and MB systems. This Bayesian control model estimates the system's reliability on MB's SPE signals and MF's RPE signals. In addition, during the selection task, these behavioural correspondence parts in the human brain were monitored by an fMRI device.

As a result of the experiment, the MB system tends to gain control of the specific task due to the high level of uncertainty in this task. Moreover, the increased cognitive effort is an indicator of MB control. An increase in SPE signals stimulates the dorsolateral prefrontal cortex, whereas a rise in RPE signals induces the ventral striatum under

fMRI. Significantly, this work provides evidence of a possible arbitration mechanism in the human brain that determines switches between MB and MF learning systems. According to fMRI results, direct connectivity between the arbitrator and MF-related regions is found. However, a direct interaction between arbitrator regions and MB behaviour has not been found. According to findings, the MF might be the default control system in human behaviour since this system is cognitively economical unless it produces a high SPE signal. Therefore, MF might play an inhibiting role in the arbitration between the two systems.

3.2 The Successor Representation in human reinforcement learning

In [I+]; the MF, SR and hybrid SR-MB (which uses MB's one-step model instead of the classical multi-step SR approach) algorithms are compared in terms of sensitivity for reward revaluation, transition revaluation and policy revaluation tasks. In addition, human experiment data are collected from two different human experiments. The first experiment (passive learning task) measures human participants' reward and transition revaluation scores compared to employed algorithms. The MF displays insensitivity against reward revaluation, transition revaluation and policy revaluation tasks. The hybrid SR-MB and SR have analogous sensitivity to human results in reward revaluation tasks. However, SR has less sensitivity than the score of the human participants, where the hybrid SR-MB model displays similarities with human data. The second human experiment includes a multi-step instrumental task that can also investigate human participants' policy revaluation tasks besides reward and transition revaluation. The hybrid SR-MB is as sensitive as human participants' reward, transition, and policy revaluation results in this task.

As a result of the comparison between human participant data and different algorithms, the noteworthiness of SR in human behaviour has been empirically argued. The SR is the most economical algorithm against changes in reward structure. However, according to experiment results, SR displays less sensitivity to changes in transition and policy.

3.3 Predictive Representations can link model-based reinforcement learning to model-free mechanisms

In [Rus+17], the Tolman's rat maze experiment environment [Tol48] is adjusted in a 10x10 grid world environment with four cardinal directional actions. The environment is simulated to investigate the different variants of SR: The original Successor Representation (SR-TD), Off-policy experience resampling (SR-Dyna) and Dynamic Precomputation of Successor Representation (SR-MB) in latent learning task(changes in the value of the reward), detour task(state manipulation in the maze by adding a wall) and novel(policy) revaluation task (changes in starting position and changes in value and position in reward). The state value function is recorded for each task and variant of SR.

Based on the experimental results, SR-TD can solve the reward revaluation task where the change in the reward does not affect policy by changing the structure of the state trajectory. However, SR-TD cannot solve the transition revaluation task, as the future state occupancy matrix, M, contains long-run cumulative state occupancies instead of one-step transition distribution. Therefore, the M is insensitive to environmental manipulations by adding a blockade. However, similar to [I+]'s experiment changing M with one-step transition model T (TD-updates preserved) makes SR-MB sensitive against changes in the transition revaluation task. However, the SR-MB cannot solve the policy revaluation task. The third algorithm, SR-Dyna, contains a future expected occupancy matrix M similar to classical SR but updates the matrix offline. The SR-Dyna can solve both detour and policy revaluation tasks. However, the SR-Dyna is significantly slower compared to other SR models.

In this paper[Rus+17], it has been argued that dopamine could have a significant role in action selection by computing prediction error. After a certain threshold (unexpected situation: habituation failure in our work), prediction error may cause a change in future occupancy matrix M from having a multi-step predictive system to a single-step prediction.

Significantly even though this work[Rus+17] proves that SR-TD cannot solve detour tasks for the rats, it has been suggested in the works[GD13; Alv+08] and that the animals tend to choose the shortest part to reward in detour tasks different than humans. However, a human can solve detour tasks by relearning for the shortest path [SG15]. Moreover, Daw demonstrates in [Daw+11] that the SR-TD algorithm fits human participants' choice readjustment in changing spatial mazes. This idea supports our

work that classical SR might solve detour tasks comparably with human subjects in the work [NS22].

3.4 Habituation and Goal-Directed Arbitration Mechanisms and Failures Under Partial Observability

In [San20], a novel grid and partially observable Hotel Elevator Task (HER) environment are designed to explain the human behavioural conflict between habitual and goal-directed behaviour. This task is a grid illustration of a hotel corridor, and habituation failure occurs due to changes in the starting point of an agent. This work[San20] aims to explain how humans deal with an anomaly and rehabituate again in the changing environment. Since we use the 3D structure of this environment in our work, we will provide detailed information about HER environment in the "Methodology" chapter.

This work introduces a novel system to explain arbitration between habitual and goal-directed behaviour. Firstly, during habituation in the environment MF system is the prevailing system in decision-making. The SR system monitors the habituation in parallel with MF to detect mismatches. The MF mismatches are detected by SR-TD error signals, which estimate how the current situation will end up. When an anomaly occurs during habituation, SR-TD signals exceed a certain threshold and trigger the MB system. The SR-triggered MB alarm system applies retrospective inference on the MF system to increase the attentional mechanism of the agent through enhancements in observation space in the partial observable HER environment. Thanks to increased observational space, the agent can detect the source of the habituation failure. After parallel collaboration of MF and MB mechanisms, the radius of the observation space decreases parallel with MF-TD error as the agent gradually rehabituates the environment. The system triggers returning for the SR-supervised computationally less-cost MF habituation procedure.

As a result, this work is essential since, in our work, we model the habituation alarm system by employing MF and SR reinforcement learning algorithms in the 3D version of the HER environment. Moreover, the following work[NS22] aims to detect anomaly cases in human participants in the identically designed HER environment.

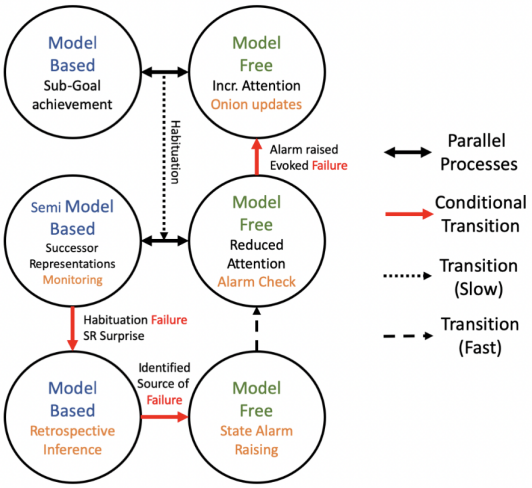

Figure 3.1: Model Free / Model Based arbitration mechanisms [San20].

3.5 Behavioral Correlates of Habituation and Deliberation: A virtual reality study with the Hotel Elevator Rows Task

Neto[NS22] extends the previous research [San20] from grid world reinforcement learning environment experiment to virtual reality human behavioural experiment in her work. Moreover, this research [NS22] provides comparable human behavioural data for our work. Thus, we will give detailed information on methodology, results and discussion about this work in this section. The human behavioural experiment aims to explain the underlying phenomenon of the arbitration between habitual and goal-directed behaviour in HER task (introduced in [San20])by conducting a Virtual Reality(VR) experiment for fifteen human participants.

Hotel Elevator Rows (HER) task

Briefly, the HER task is designed to detect habituation anomalies of human participants. In the HER environment, participants begin from the ground floor(hotel lobby corridor). Two symmetrically opposite elevator doors (left and right) lift the human participants to the room floor. In each trial, only one of these elevator doors opens. If the right elevator door opens, the trial is specified as **common trial**. Otherwise, if the left elevator

door opens, the trial is called **uncommon trial**. Furthermore, the room floor comprises twelve hotel room doors. The hotel rooms are four-sided and symmetrically located.

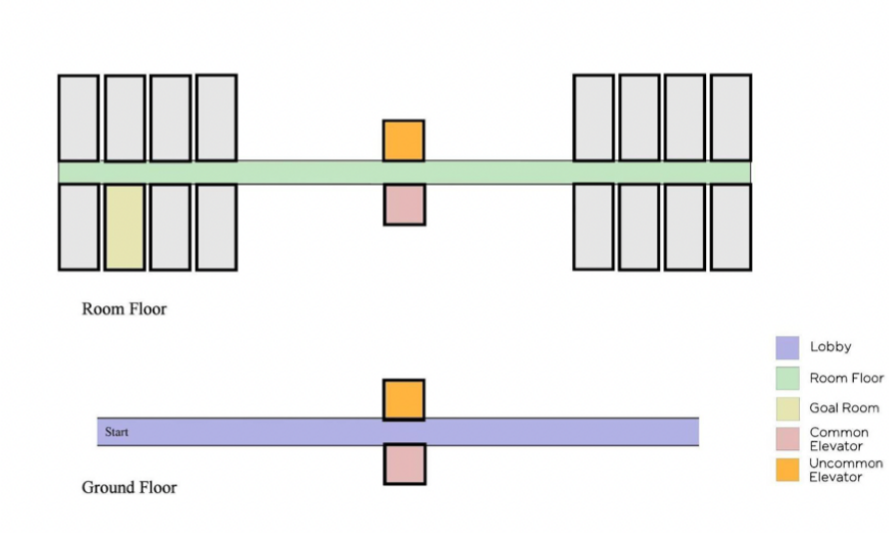

Figure 3.2: HER Behavioral Environment [NS22]

Experimental Setup

In the VR experiment, human participants can move forward, backwards and rotational with controllable speed. At the same time, they can open doors and trigger signs by clicking the controller. In each trial, human participants need to reach the correct room, which has the number 303. The correct room's location does never change. The experiment is composed of 3 main phases. At first, the participants have two trials to get used to the environment, which is called **exploration phase**. The exploration phase data is not taken into account for analysis. Secondly, the trials between 2 and 12 are specified as **habitation phase**, which ensures that human participants are being habituated for the HER task. The third phase is **testing phase**. In the exploration and habituation phase, there are only common trials. However, in the testing phase, the 13,18,21 and 28. trials are uncommon trials. In the given figure 3.3, the orange highlighted trials represent uncommon trials in the testing phase. The uncommon trials are used to generate an anomaly for human participants by opening the left elevator door on the first floor.

Figure 3.3: HER Behavioral Experiment Setup [NS22]

This setup in [NS22] is designed for human participants. Therefore, the computational measurement of the attentional mechanism is not possible like in Sanchez's work [San20]. For this reason, additional factors are employed to indicate the quantity of the attentional mechanism for the human participants. First, the room floor is designed with two different wallpaper colours. The wall in the correct corridor is reddish, whereas in the incorrect corridor blueish. The difference in the wallpapers aims to increase human participants' attentional mechanism about the corridor's correctness. The different colour design in the task is similar to SR-TD signals in Sanchez's work [San20], which trigger alarm mechanism-based SR-TD state prediction error. Secondly, during the trials, the participants have a virtual mobile phone. Random words from three categories (food, cities and animals) appear at random intervals for 2 seconds on the phone screen. At the end of the trial, the word count from a randomly selected category is asked. The participants can only see the phone screen if they look down. This setup aims to decrease the participants' attention to the task since, during habituation, they concentrate more on pop-ups from the virtual mobile phone. "Looking at the phone" is a binary logged value to measure this setup. If the frequency of "Looking at the phone" decreases, it is assumed that the attentional mechanism of human participants increases. Thirdly, a room sign checker is set up on the way to the elevator. The participants can learn the information about the room by clicking on the room sign checker. Triggering the checker means the destruction of the habituation process.

Furthermore, some indicators might inform whether participants behave in the habituation or deliberation mode. At first, the time duration is an indicator. A decrease in time duration for trials indicates that the participant habituates the task. Secondly, increased head orientation variability refers to hesitation during the trial. The using head-orientation variability as an indicator of deliberation idea is based on Tolman's Vicarious Trial, and Error(VTE) hypothesis on rats in the maze experiment [Tol48]. Tolman hypothesizes that increasing head-orientation variability means creating a cognitive map for the environment, which is one of the main properties of goal-directed behaviour.

4 Methodology

In this chapter, we introduce the design information for the 3D HER environment and the learning agent, including two sections: "Environment" and "Agent". The first section provides information about the environment's design and parameters. Moreover, we aim to explain why we use the Unity ML-Agent tool to design our experiment. The second section provides information about the agent parameters and agent environment interaction details. Finally, we introduce the algorithms we employ to train the learning agent to investigate habitual behaviour and detect habituation failures.

4.1 Environment

The Hotel Elevator Rows(HER) environment is introduced in [San20]. The main idea behind the environment's design is observing MF learning control habitual behaviour with Successor Representation(SR) algorithm supervision. In addition, Sanchez [San20] proposes an alarm mechanism oversight by the MB reinforcement learning algorithm against habituation failure. The alarm mechanism increases the MF system's observation space to detect the habituation failure's source. However, this MB oversight part of this alarm mechanism is not within the scope of our work. Our focus is Successor Representation(SR) algorithm supervision on habitual behaviour.

Another work we compare with our results is the human behavioural experiment conducted by Neto to observe the types of habituation failures [NS22]. In HER environment, participants begin a trial on the ground floor, designed as a hotel lobby corridor. Afterwards, participants must take an elevator to reach the room floor. However, in each trial, only one elevator between common and uncommon is available. Human participants are not aware availability of which elevator in which trial. In addition, as mentioned in the "Related Work" section 3.5, the virtual mobile phone task distracts human attentional mechanisms from forgetting which elevator is taken for lifting. Since the human participants experience the exploration and habitation phase with only common elevator availability, habituation failure is expected in the first and the other uncommon trials. If the participants reach the correct room 303, the trial finishes. If they reach the wrong room in an uncommon trial, the behaviour indicates a habituation failure due to following the habituated path from the common elevator

door. However, triggering the wrong room sends pop-up information to the VR screen instead of terminating the trial. Finally, Neto investigates the different habituation failure types in her experiment.

In our experiment, we employ the putative habituation algorithm MF reinforcement learning and its supposed habituation supervision algorithm SR. Our concentration is testing habituation failures due to MF learning and the suitability of SR for the habituation monitoring system. Secondly, this experiment provides a computational baseline to identify the learning agent's habituation failures and compare them with human experiment data[NS22]. Therefore our environment objects and parameters are designed to be comparable with human behavioural experiment methodology. In our experiment, the most similar error type to human behavioural experiments is trajectory error due to habituation failure. Measuring habituation failure through time duration is not possible in the ML-Agent experiment since the time duration of our experiment depends on the CPU device. In addition, we cannot expect from learning agent to make Vicarious Trial Error(VTE) resulting from an increase in head-orientation variability.

To design this intended experiment environment, we utilize Unity Machine Learning Agents Toolkit [Jul+18].

4.1.1 Unity Machine Learning Agents Toolkit (ML-Agents)

In designing the 3D HER environment, we utilize the Unity Machine Learning Agents Toolkit (ML-Agents) framework in version 2021.3.6f1 [Jul+18]. ML-Agents Toolkit provides an open-source project that offers to design a 3D environment for intelligent training agents.

Unity ML-Agents Toolkit allows users to train learning agents with machine learning algorithms: unsupervised, supervised, and reinforcement learning. As already stated in previous chapters, we employ reinforcement learning algorithms to investigate Sanchez's proposed [San20] alarm mechanism and create a computational baseline to compare with Neto's human behavioural experiment [NS22].

Reinforcement learning employs a sequential decision-making process and learns through interaction between a learning agent and an environment. In the Unity ML-Agent toolkit, a learning agent perceives the environment through its sensors. The agent evaluates the obtained information from the learning environment and sets an action. Thanks to the learning agent's mapping observation of the action, the agent improves a policy to maximize cumulative reward. Thus, an agent needs to experience many trials

and update the policy iteratively. Unity offers the environmental and learning agent parameter design possibility and flexibility for this purpose. In addition, the Unity tool provides training and inference phases. In the training phase, the learning agent learns the optimal policy. On the other hand, in the inference phase, the agent perceives the environment and acts according to the agent's current policy.

Moreover, since the deep-learning framework of reinforcement learning can learn in a 3D dimensional environment faster than standard reinforcement learning algorithms, we employ deep-learning forms of Q-learning and Successor Representation in our work. We introduce the employed deep reinforcement learning algorithms in section 4.2.2. The Unity toolkit provides an appropriate simulator and engine for deep learning algorithms. For this reason, we opt to design our 3D HER environment using the Unity tool.

4.1.2 Environment Design

The 3D Unity HER environment contains objects and a single agent. In the environment, the agent interacts with objects. Due to this interaction, the environment sends reward signals to the agent. The obtained reward signals in the Unity environment inform the agent about the interaction outcomes between applied action and environmental objects.

The environment includes 20 objects. The environment stands on the "ground" object and is enclosed by the "wall"s object. The environment contains two elevators and 16 rooms, as figure 4.1 displays. The design of the elevators and rooms is entirely symmetrical, like the environment in the human behavioural experiment. The significant difference is that the human behavioural experiment contains two floors: the ground floor and the room floor. The reason for this difference is that in the human experiments, the virtual phone setup causes a distraction for the human participants to forget which elevator they have used. This setup aims to decrease attentional demand for the participants and observe habituation failures. However, in the learning agent experiment, it is sufficient not to provide the agent with a byte of information from which type of elevator it starts. Thus, having only one floor in our environment provides an identical structure to the human experiment. Secondly, the investigation might deviate in an unnecessary direction due to time constraints and increased tasks given to the agent.

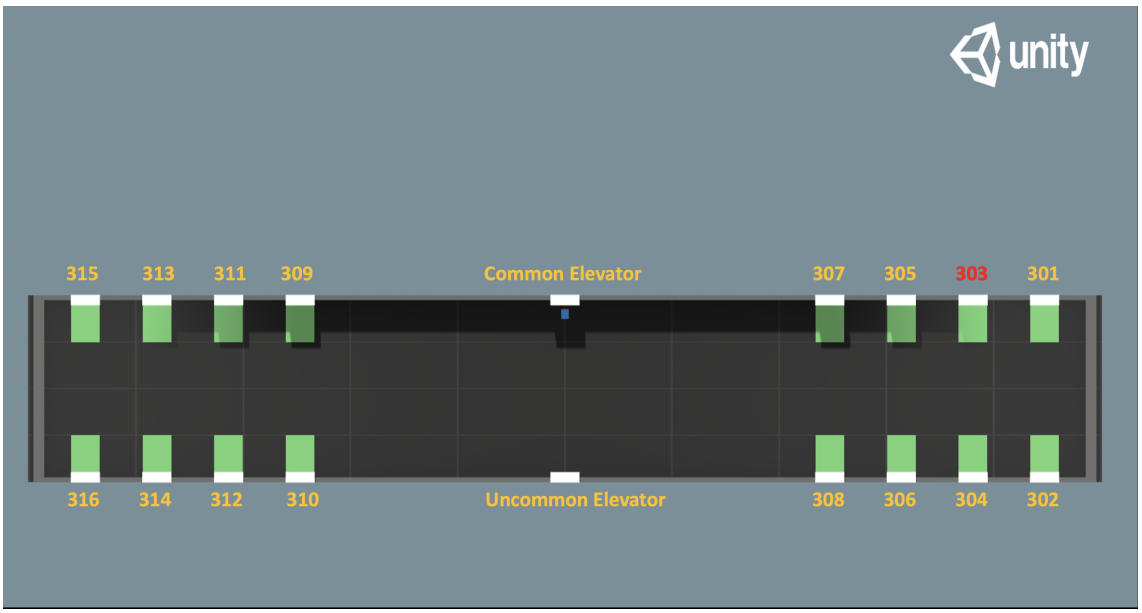

Figure 4.1: The 3D Unity HER Environment Design

Goal and Rewards Design

As already stated, the reinforcement learning agent learns through rewards. Similar to the human behavioural experiment [NS22], the goal of the agent is reaching the room with number 303 (see the red highlighted door in the figure 4.1). If the agent can find the room with number 303, the episode terminates, and the agent receives a +10 reward. However, reaching the wrong rooms does not send any positive or negative reward signal. In addition, reaching the wrong room does not terminate the episode; likewise, in the human behavioural experiment, triggering the wrong room does not finish the trial. The wrong rooms are employed to manipulate the agent to enforce the habituation failure.

Another reason which terminates an episode is the step count limitation for the learning agent. The maximum step count of the agent in the environment is limited to 500 steps. Therefore similar to reaching the correct room goal, if the step count reaches 500 in an episode, the episode terminates. Moreover, the agent is penalized in each step with a reward of -0.002. We adjusted the step cost as $1/maxstepcount$ for each step. In the human behavioural experiment, one of the goals of the human participant is to reach the correct room as fast as possible. Therefore each step of the learning agent is penalized in the same way to force the learning agent to finish the

task quickly. Secondly, limiting the step count causes it to converge for reinforcement learning training in a finite amount of time.

4.2 Agent

In the Unity environment, an agent class is used to create a learning agent and set the agent's parameters. These agent class implementations determine the agent's observations and how these observations influence the agent's policy. In the Unity ML-Agent framework, these unique features of an agent are defined with the subclass of **Behaviour Parameters**. This parameter specifies the agent's decision and which action to take after an observation. Every agent in the ML-Agent framework must have a suitable Behavior Parameter. Since we employ a single agent, we name only one behaviour parameter configuration in our work. In addition, behaviour parameters allow external inference with CPU and export the Unity-built model. We will introduce this property in 4.2.1.

Observation Space

The learning agent perceives the environment through the sensors. Then agents save the perceived values in an observation vector. The agent evaluates the environment through this vector. ML-Agents tool provides multiple possibilities to create an observation space for the learning agent. The first option is a visual observation based on the human-based seeing environment as an image. The second option is ray-cast observation through the vector properties interacting with objects in the environment. Third is grid observations which combine 2D spatial representation with visual observations. In our work, we opt to design an agent with ray-cast vectors since this method does not require huge observation space like visual observation. In addition, detection through the ray-cast vector sensor provides an optimal setup with less computational time and low dimensional observation space.

3D Ray Perception Vector

The ray-cast observation sends several rays into the environment and produces an observation vector by hitting the rays into the objects. In our agent design, we employ 3D ray perception sensor components. These sensor components are attached to the GameObjects. In the table below 4.1, we provide our parameters and parameter definitions. However, in our experiment, since we train our agent through the python

API communication, we send whole observation space $ObservationSpace \in \mathcal{R}^{168}$ in the deep neural network. Therefore "Stacked Ray casts" parameter is irrelevant information for our experiment. We leave the information in the table for extra information.

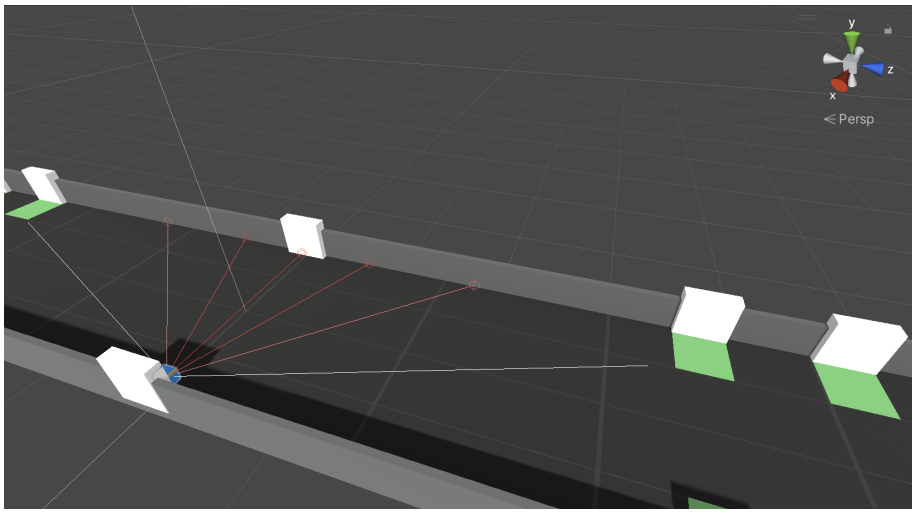

Figure 4.2: The 3D Unity HER Environment Ray Perception Sensor Illustration

Ray Perception Sensor 3D		
Component Name	Number	Definition
Detectable Tags	6	Number of distinguished GameObjects
Rays Per Direction	3	Number of rays to the left and right of center
Max Ray Degrees	70	Cone size for rays
Sphere Cast Radius	0.5	Radius of sphere to cast
Ray Length	35	Length of the rays to cast
Ray Layer Mask	"Mixed"	Controls which layers the rays can hit
Stacked Ray casts	3	Number of ray-casts stacked in neural network
Start Vertical Offset	0	Ray start is offset up or down by this amount
End Vertical Offset	0	Ray start is offset up or down by this amount

Table 4.1: 3D Ray Perception Sensor Configuration

As a result, our learning agent's $ObservationSpace \in \mathcal{R}^{168}$ is formed by 3D Ray Perception sensors. The entire state of the 3D HER environment is not fully visible through the agent's sensor. Therefore the experiment environment is **partially observable environment** for the agent perception.

GameObjects	Tagging Class	Behavioral Experiment Trajectory Error Types
Ground	-	-
Walls	wall	-
CommonElevator	elevatordoor	-
UncommonElevator	elevatordoor	-
Room301	fourthdoor	Correct Corridor, Correct Side(wrong room)
Room302	fourthdoor	Correct Corridor, Incorrect Side(wrong room)
Room303	**thirddoor**	**goal room**
Room304	thirddoor	Correct Corridor, Incorrect Side(wrong room)
Room305	seconddoor	Correct Corridor, Correct Side(wrong room)
Room306	seconddoor	Correct Corridor, Incorrect Side(wrong room)
Room307	firstdoor	Correct Corridor, Correct Side(wrong room)
Room308	firstdoor	Correct Corridor, Incorrect Side(wrong room)
Room309	firstdoor	Incorrect Corridor, Correct Side(wrong room)
Room310	firstdoor	Incorrect Corridor, Incorrect Side(wrong room)
Room311	seconddoor	Incorrect Corridor, Correct Side(wrong room)
Room312	seconddoor	Incorrect Corridor, Incorrect Side(wrong room)
Room313	thirddoor	Incorrect Corridor, Correct Side(wrong room)
Room314	thirddoor	Incorrect Corridor, Incorrect Side(wrong room)
Room315	fourthdoor	Incorrect Corridor, Correct Side(wrong room)
Room316	fourthdoor	Incorrect Corridor, Incorrect Side(wrong room)

Table 4.2: Game Objects and their tagging classes

GameObjects Tagging

One of the basic design patterns in our experiment is **detectable tags**. Each GameObjects belongs to one tagging class in the Unity ML-Agents framework. The ray perception sensor identifies the game objects according to their tagging classes.The 3D HER environment consists of 20 GameObjects: Ground, Room301, Room302, Room303, Room304, Room305, Room306, Room307, Room308, Room309, Room310, Room311, Room312, Room313, Room314, Room315, Room316, Walls, CommonElevator, UncommonElevator. In our experiment, we create 6 Detectable Tag classes: elevator doors, wall, first door, second door, third door and fourth door. The door categorization is decided according to the closeness of the elevator. Since in the human experiment [NS22], trajectory errors are categorized as "Correct Corridor, Incorrect Side", "Correct Corridor Correct Side","Incorrect Corridor, Incorrect Side" and "Incorrect Corridor Correct Side" 4.3. The correct room is not identified in the trajectory error group "Correct Corridor Correct

Side". If the agent detects the correct room in the third door class, the agent tends to reach the doors in the third door class. Therefore, based on our assumption, the identified trajectory errors in the human experiment are still observable in this environment. Moreover, classing whole doors as a "door" class causes increases the difficulty of learning. On the other hand, tagging the doors uniquely for their number causes an increase in observation space due to the increase indoor tags from 4 to 16. The increase in observation space poses a deceleration in training with SR due to SR's discounted future states matrix M.

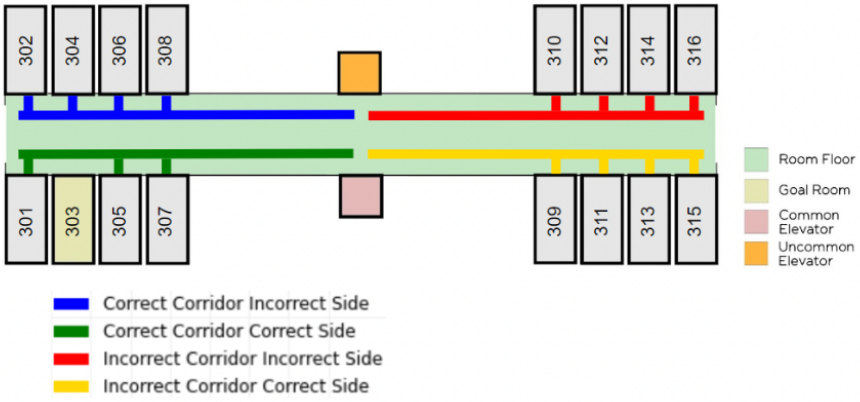

Figure 4.3: Categorization of trajectory errors in human experiment [NS22]

Action Space

In a Unity ML-Agent environment, the action of a learning agent is determined by a learning algorithm according to the expected reward. The learning algorithm selects an action from agents **actions space**. Formally, action space includes all possible action domains of the learning agent. The learning agent in the HER environment contains one discrete branch of actions with five discrete actions. The learning agent has discrete actions: move forward, move backwards, rotate left, rotate right and not move. The action space is designed suitably for human behavioural experiment [NS22] and Sanchez grid HER environment [San20].

$$actionspace \in \{forward, backward, rotateleft, rotateright, notmove\} \tag{4.1}$$

Our experiment aims to train the learning agent with the MF reinforcement learning algorithm and the Successor Representation algorithm. However, these algorithms are not provided in the Unity ML-Agents Toolkit package. Therefore an inference of external algorithms is required. Python Low-Level API includes communication between the environment and externally implemented reinforcement learning algorithms. Because of this reason, we use Python Low-Level API to train the learning agent with these reinforcement learning algorithms.

4.2.1 Python Low-Level API

The ML-Agents package includes a low-level API component `mlagent_envs` that allows the interaction between a Unity environment and an externally implemented reinforcement learning algorithm via a Python interface. Using the low-level API component, the user can access the observation vector and obtain a reward signal for each learning agent in each time step. In addition, users can reset the environment and send action signals to the environment. The other benefit of this feature is that the AI developer can train different Unity environments simultaneously using different CPU ports. In addition, the machine learning developers can accelerate the simulation time scale by using the Custom Side Channels. We simulate the training 100 times faster than the normal timescale by using Custom Side Channels. Because of these provided features, we opt the train our learning agent by utilizing this framework.

4.2.2 Deep Reinforcement Learning Algorithms

Moderate reinforcement learning methods, such as Q-learning, use tabular methods. The tabular form saves the state and action values on a table to find an optimal policy. The table of an algorithm resembles the brain of a reinforcement learning algorithm [Mog19]. For instance, Q-learning uses a table that takes the state values as a row and action values as a table column. The tabular method might quickly provide an optimal solution for reinforcement learning problems, including small observation and action space. However, deep reinforcement learning methods offer preferable solutions to complex environments with massive data. Deep reinforcement learning combines reinforcement learning and a neural network system. Deep reinforcement learning utilizes the **function approximation** method. The function approximation eliminates the requirement of saving all state and value pairs. Because of this reason, training with deep reinforcement learning in 3D environments is preferable [Wil].

We employ the deep learning form of Q-learning and Successor Representation in our work. In the experiment, we use the version of `python` 3.6.2, `PyTorch` 1.4.0, `unityagent` 0.4.0 `and` `NumPy` 1.19.2 libraries. Central Processing Unit(CPU) is employed in the Jupyter Notebook editor for the experiment. Each DQN and SR training experiment run in 4000 episodes. The maximum step is limited to 500 for each episode. Epsilon decays exponentially from 1.0 to 0.01 with a decay rate of 0.995. The epsilon reaches the minimum value of 0.01 in the first 1000th episode. The epsilon decay parameter is tuned intuitively for the suitability of the task based on our observations before the experiment; the agent finds the correct room 303 in the first 1000 episodes. After 1000 episode agent needs to act with its optimal policy to preserve its habituation.

Deep Q-learning (DQN)

In our experiment, we employ the Deep Q-Learning, introduced by (DQN)[Mni+13]. We adapt the implementation of Udacity GitHub Page [Uda] in our experiment. In the below equation 4.2, the loss function of the Deep Q-learning Network is presented.

$$L_i\left(\theta_i\right) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q\left(s', a'; \theta_i^-\right) - Q(s, a; \theta)\right)^2\right] \tag{4.2}$$

In each step, DQN maps the input vector of 168 states into the five possible discrete actions to find the optimal policy for reaching a correct room with less step count. The neural network architecture of DQN contains two hidden, fully connected layers. Both layers include 64 neurons in the DQN structure. The network contains two Rectified Linear Unit (ReLU) activation functions between two layers and between the second layer and output. The ReLU activation function outputs values in range from zero to infinity, as given in the ReLU function : $ReLU(z) = max(0, z)$. The neural network structure of DQN minimizes the difference between the target value and the expected value by minimizing the objective function. We preserve the values of the original implementation of DQN [Uda]. The DQN network has the $1e5$ buffer size. The discount factor γ is 0.99. The network's learning rate α is $5e-4$. The τ parameter is tuned as $1e-3$ for a soft update of the target parameters. The network updates the parameters of the target network every four steps.

DQN Hyperparameters	
Parameter	Value
Replay buffer size	1e5
Batch size	64
Discount factor (γ)	0.99
Learning rate (α)	5e-4
τ	1e-3
Update frequency	4
FC1 units	64
1.Activation function	ReLU
FC2 units	64
2.Activation function	ReLU
Total number of episodes	4000
Max steps per episode	500
ϵ start	1.0
ϵ end	0.01
ϵ exponential decay rate	0.995

Table 4.3: DQN Hyperparameters

Deep Successor Representation(SR)

In the Unity ML-Agent experiment, we utilize the Successor Representation (SR), introduced in the paper "Count Based Exploration with Successor Representation" [MBB18]. We design the implementation from the project [Bon]. The presented neural network structure combines the loss functions of Q-Network (Policy model) 4.3, SR-network 4.4 and the Prediction network 4.5. TD symbol denotes the Q-network in the paper [MBB18] since the TD loss function uses Q-learning's TD update mechanism.

$$\mathcal{L}_{\mathrm{TD}} = \mathbb{E}\left[\left((1-\tau)\delta(s,a) + \tau\delta_{\mathrm{MC}}(s,a)\right)^2\right] \tag{4.3}$$

$$\mathcal{L}_{\mathrm{SR}} = \mathbb{E}_{\pi,p}\left[\left(\phi\left(S_t;\theta^-\right) + \gamma\psi\left(S_{t+1};\theta^-\right) - \psi\left(S_t;\theta\right)\right)^2\right] \tag{4.4}$$

$$\mathcal{L}_{Pred} = \left(\hat{S}_{t+1} - S_{t+1}\right)^2 \tag{4.5}$$

The overall loss function 4.6 is minimized in the network structure by having the weight values $w_{TD} = 1$, $w_{SR} = 1000$ and $w_{PRED} = 1e-3$.

$$\mathcal{L} = w_{\text{TD}}\mathcal{L}_{\text{TD}} + w_{\text{SR}}\mathcal{L}_{\text{SR}} + w_{Pred}\mathcal{L}_{Pred} \tag{4.6}$$

SR Model Hyper-parameters	
Parameter	Value
Replay buffer size	3e5
Batch size	64
Discount factor (γ)	0.99
Learning rate (α)	2.5e-4
Update frequency	4
FC1 units(Q-network)	128
1.Activation function(Q-network)	ReLU
FC2 units(Q-network)	64
2.Activation function(Q-network)	ReLU
FC1 units(PredNetwork)	64
1.Activation function(PredNetwork)	ReLU
FC2 units(PredNetwork)	128
2.Activation function((PredNetwork)	ReLU
FC1 units(SRNetwork)	64
1.Activation function(SRNetwork)	ReLU
β(reward exploration bonus)	0
w_{TD}	1
w_{SR}	1000
w_{PRED}	1e-3
Total number of episodes	4000
Max steps per episode	500
ϵ start	1.0
ϵ end	0.01
ϵ exponential decay rate	0.995

Table 4.4: SR Model Hyperparameters

Like DQN, the SR takes the vector of states with size 168 and maps the output of five discrete action values. The Q-network in this structure contains a fully connected first hidden layer with a size of 128 and a second hidden layer with a size of 64. The SR network contains a single fully connected hidden layer with 64 neurons. The Prediction

network includes a fully connected first layer with a size of 64 and a second layer with 128 neurons. The SR model includes two ReLU activation functions for Q-network and Prediction network; one ReLU activation function for SR-Network. In the same way with DQN, we preserve the original hyper-parameters of implementation [Bon]. The SR has the 3e5 replay buffer size. The discount factor γ is 0.99, the learning rate α is $2e-4$, τ is $5e-3$. The model updates the network every four steps. In addition, Machado introduced a reward bonus $\beta = 0.025$ to incentive representation; however, we ignore the reward exploration bonus by fixing the $\beta = 0$ for the comparability with DQN.

5 Experimental Evaluation

In our experiment, we aim to demonstrate the habituation failures and rehabituation performances of DQN and Successor Representation (SR) algorithms in the 3D HER environment. The learning agent is trained and tested with the identical task in the human behavioural experiment [NS22]. This setup provides a possible comparison between the simulation results and human behavioural data. Another purpose of this work is to demonstrate the sensitivity of the Successor Representation algorithm against the reward and transition revaluation tasks. Finally, the proposed parallel MF and Successor Representation habituation monitoring system [San20] is investigated in the 3D environment.

In the first section, the experimental setup is introduced. Then, we will provide the experiment's results in the second section. Finally, in the last section, the results will be discussed.

5.1 Experimental Setup

In the 3D HER reinforcement learning experiment, the learning agent has two possible initial positions. The agent can either start in front of the common or uncommon elevator. During single training or single test session, the starting position does not change. However, in each sequential session, the starting point of the learning agent is changed to the opposite side.

The goal of the learning agent is to find the correct room, 303. If the agent finds the correct room, the agent receives a +10 reward, and the episode terminates. On the other hand, if the learning agent cannot reach the correct room in 500 steps, the episode terminates. Afterwards, the agent restarts the episode from the starting point. The 3D HER task experiment includes five training and ten testing session. In addition, each training and testing session is conducted with five different random seeds to ensure the results are statistically trustable. In the "Experimental Results" section, we provide the average result of five different seeds.

Each training session includes 4000 episodes. We determine to train the agent in 4000

episodes per training session in order to force the agent to habituate to the task and the environment. For the same reason, we adjust the epsilon decay to reach the minimum value in the 1000th episode. Each testing session consists of 10 episodes. During each testing session, the learning agent always acts with the same policy since the learning does not occur in the testing experiment. We suppose that even less episode is sufficient for testing; however, to eliminate the chance factor by performing paired t-test, we keep the amount of episode at 10 for each testing session.

This section introduces the experiment setup in two subsections: training and testing. In the training experiment, we aim to demonstrate the performance of DQN and SR algorithms against the habituation anomaly due to changing the starting point of the agent for each sequential training session. In this way, the training session results allow us to investigate the sensitivity of these algorithm performances against reward and transition revaluation tasks. Furthermore, thanks to this training setup, the proposed SR monitored habitual behaviour alarm mechanism [San20] can be investigated in 3D Hotel Elevator Row(HER) environment.

In the testing part, we aim to compare the rehabituation performances of DQN and SR algorithms. Secondly, we intend to compare trajectory habituation failures during the testing experiment with the human behavioural experiment [NS22].

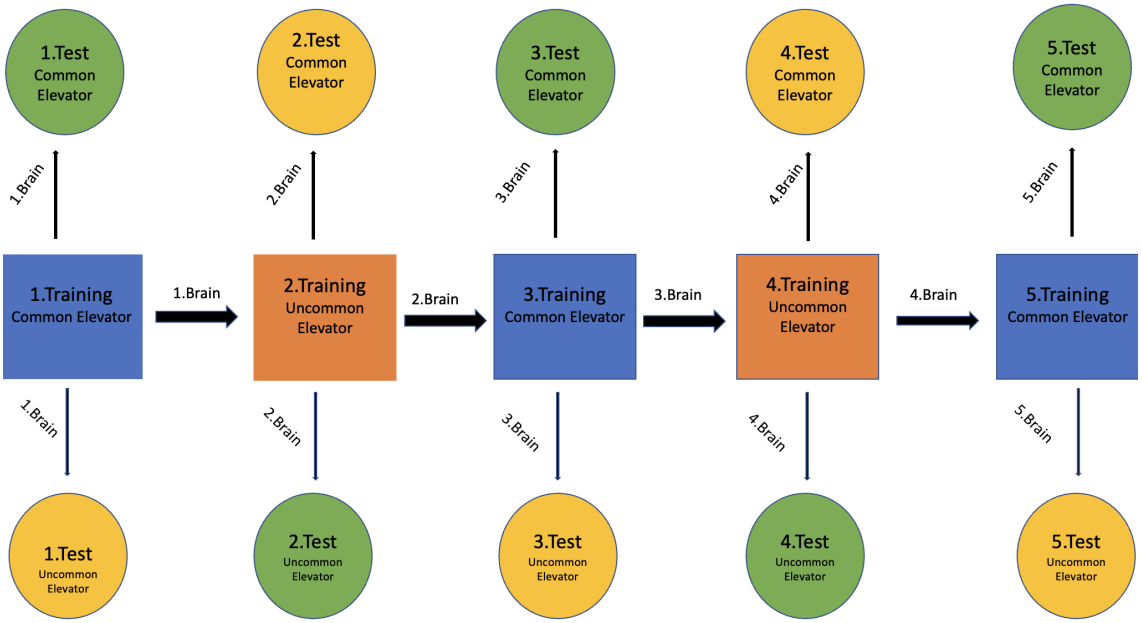

Figure 5.1: The 3D HER Reinforcement Learning Experiment Design

5.1.1 Training Setup

In the 3D HER experiment, the learning agent has two different starting points. In the common elevator environment, the agent starts in front of the common elevator at the beginning of each episode, in the uncommon elevator environment reversely. Except for the starting point, both types of environments are identical. In the training part of the experiment, we aim to evaluate DQN and SR algorithms' habitation performance for this change. The learning agent trains in 4000 episodes with five different seeds in each training session. The mean final score of each episode and the mean final step count values are saved for the result.

During an episode, if the agent cannot find the correct room(303), the agent does not receive any positive reward. The final score of the unsuccessful episode is automatically -1, as the step cost per episode is -0.002, and the maximum step count is 500 for each episode. Finding the correct room gives the reward +10. Finding the correct room terminates the episode. Therefore, the only possibility to reduce the final step count from 500 is reaching the correct room.

For the convenient interpretation of these results, we calculate the average of the last 100 episode scores and the final step count. In the 1000th episode, the epsilon value reaches the minimum value of 0.01, which implies the agent does not explore the environment anymore after this episode. The agent behaves with its last obtained policy after this episode. This behaviour shows similarity with the human behavioural pattern of **habituation**. At the end of the 4000th episode, training finishes. The agent's brain, which denotes the neural network weights of the agent in the last episode, is saved in the *.pth* file.

As introduced in figure 5.1, the agent's brain is loaded for the next training session. However, the agent starts in front of the opposite elevator door in the following training session. The agent behaves like the previous task at the beginning of each training session. Thus, the agent faces **habituation failure**. Afterwards, due to the increased TD-Error, the agent needs to improve a new policy for the opposite starting point. Since the epsilon value starts with 1.0 at the beginning of each training session, the agent can explore the environment and improves a new policy to habituate for the opposite starting position.

The setup of five sequential training sessions is presented in figure 5.1. The learning agent has the neural weights of the previous session at the beginning of each training except the first training. The first training denotes habituation since the agent improves

a policy for the room-finding task the first time. However, the second, third, fourth and fifth training sections represent rehabituation due to changes in the starting point.

By conducting the training experiment, we intend to evaluate the rehabituation performances of the MF and Successor Representation reinforcement learning algorithms. Secondly, we aim to evaluate the performance of the SR algorithm for reward revaluation and basic transition revaluation tasks. Finally, we investigate the plausibility of the Successor Representation monitored MF habitual behaviour alarm system [San20] against habituation anomaly in the 3D HER environment.

5.1.2 Testing Setup

In the training experiment, the agent improves a policy for the room-finding task by learning through the DQN and SR algorithms. As we already mentioned in the previous section 5.1.1, the term agent's brain denotes the last obtained policy saved on neural weights of deep learning algorithms after each training session. At the beginning of each testing session, the agent's brains obtained from training with each seed are launched into the identical seed-numbered learning agent. Therefore every seed is investigated separately to eliminate results found by chance. The agent in each seed is trained ten times for the same purpose. In subsection 5.2.2, the standard deviation between different seeds' mean scores is visualized with the standard deviation error bars.

In the testing part of the experiment, the agent is tested in common and uncommon elevator environments. If the agent is tested for the previously trained type of the environment, we specify the testing as **test on the trained elevator**. Otherwise, the testing type is **test on the not-trained elevator**. In experiment design figure 5.1, the green circles represent the test on the trained elevator, whereas the yellow circles denote the test on the not-trained elevator.

During each testing experiment, the learning through the reinforcement algorithms is inactive. Therefore the agent acts according to the last improved policy of the previous training session. We save the final mean score values of the "test on trained elevator" testing sessions. This session has saved mean score values, allowing us to compare the rehabituation performance of each testing session of the DQN and SR algorithms. In addition, each algorithm's results are compared with the human experiment's mean trial duration in terms of rehabituation trends.

In the "test on not-trained elevator" experiment, we aim to investigate habituation failures of the learning agent through the identified trajectory errors in the human behavioural experiment [NS22]. Therefore a string of information about the triggered room number by the agent (only single room information per episode is considered, for instance; if the agent triggers room 306 six times in one episode, we count it only once) is logged into the ”*.csv*” file. Collecting triggered room number data allows us to compare our data with trajectory error occurrences in the human behavioural experiment [NS22].

5.2 Experiment Results

The experiment is conducted with five different seeds and two different learning algorithms: DQN and Successor Representation. One training section comprises 4000 episodes, whereas one test section consists of 10 episodes. We introduce the experiment results in two separate subsections: the "training results" and the "testing results".

5.2.1 Training Results

The mean score lines in figures 5.2, 5.5 , 5.4, 5.7 and 5.8 represent the average mean scores of the five different seeds per episode. The mean scores in each seed are calculated as the average score of the 100 previous episodes. The shaded area in figures 5.2 and 5.5 represents the standard deviation between five different seeds training. Therefore, the shaded area denotes consistency between the mean score results of five random seeds training. Moreover figure 5.3 and 5.6 demonstrates average final step count per episode for five different random seeds trainings. Like mean score results, step count is presented by calculating the mean of the step counts in the last 100 episodes. Both results are dependent; due to the step cost in the environment. Decreasing the step count means reaching the correct room in a shorter path. Finding the optimal path indicates the habituation degree of the agent for the task. However, the primary goal of the agent is finding the correct room. In the training experiment, the mean score refers to the first indicator of habituation. The step count is also a significant indicator of habituation for the task.

In the 3D HER task environment, the maximum score is $+10$. Due to step cost -0.002, the maximum reached mean score in DQN training is approximately $+9.955$ and in SR training is $+9.9559$. The black horizontal solid line in figures 5.2 and 5.5 represent these maximum reachable scores. The difference in maximum mean scores between the DQN and SR results from the different minimum step count for algorithms. The blue vertical

solid line demonstrates the minimum step count in figures 5.3 and 5.6. DQN agent can solve the task at least with a 21 step count. However, the SR agent finds the shortest path by solving with a 19 step count. The detailed reason for this intriguing difference between both algorithms will be discussed in the following "Discussion" section.

In addition, the entire DQN and SR training experiment consists of 20.000 episodes. Each 4000 episode belongs to one training session. Each color in figures 5.2, 5.5, 5.2 , 5.5, 5.4 and 5.7 represents one training session. These trainings are run sequentially. We present termination of a training with a dashed black line in figures 5.2, 5.5, 5.3, 5.6 and 5.8. However, in each transition between the different training sessions, the starting point of the agent is changed. For instance, in the first training, the agent starts from the common elevator side for each episode until the first training terminates. Then the agent's brain is loaded for the second training, in which the agent starts from the uncommon elevator side. At the beginning of each training, ϵ has the maximum value of 1.0, and ϵ starts to decay exponentially. In episodes 1000, 5000, 9000, 1300 and 17000; the ϵ value falls to minimum value 0.001. After these episodes, the learning agent behaves with its last improved policy in the 3D HER environment.

DQN Training Results

Figure 5.2 outlines that the DQN algorithm converges to the maximum best score of $+9.955$ roughly in 2000th episode. The algorithm's convergence signifies that the agent learns the best policy after this episode which is also supported by a dramatic decrease in the step count in figure 5.3. In addition, the standard deviation between the five seeds decreases from this time step. In the second training, the mean score peaks approximately after episode 7250. Similarly, the step count hits a low amount after this point, and the standard deviation between random seeds decreases significantly. The third training in the common elevator environment converges to the highest reward and the lowest steps after the roughly in episode 11.000. The low decrease in the mean score at episode 11250 is derived from the increase in standard deviations, which means the mean score of one of the training seeds deviates from the best score. However, the DQN algorithm in the third training converges for the best policy at the end. Both figures 5.2 and 5.3 demonstrate that the agent cannot find the best policy for the fourth and fifth training.

The interbedded representation of all five training is demonstrated in figure 5.4. The dashed black vertical line indicates where epsilon reaches the lowest value of -0.01. In addition, the training sessions with the agent's identical starting points are presented

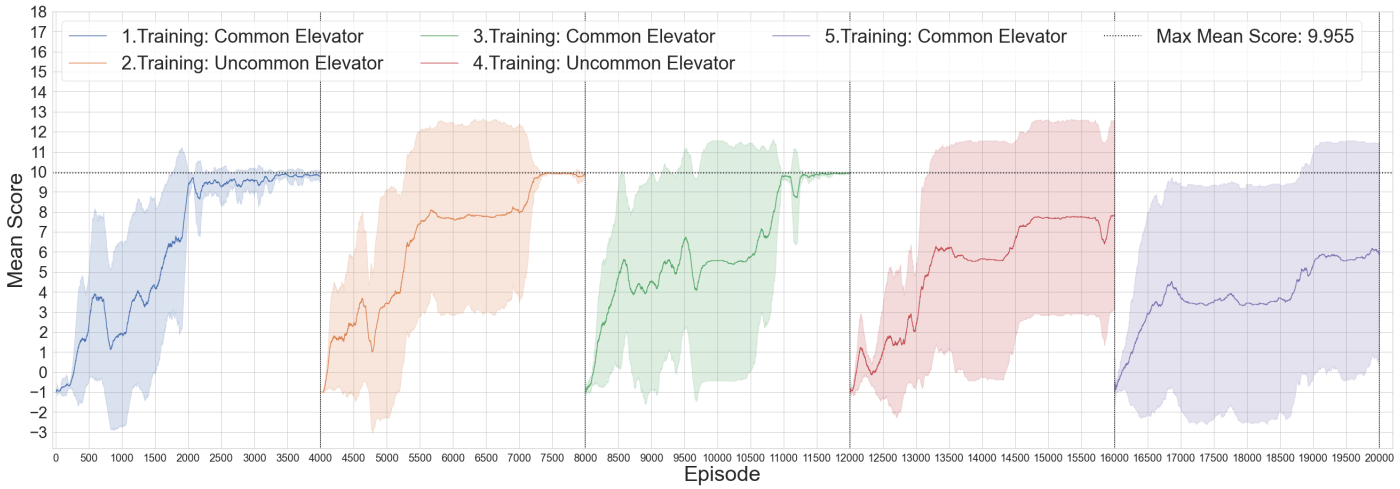

Figure 5.2: Mean scores of each DQN training per episode, including standard deviation of five different random seeds as shaded area

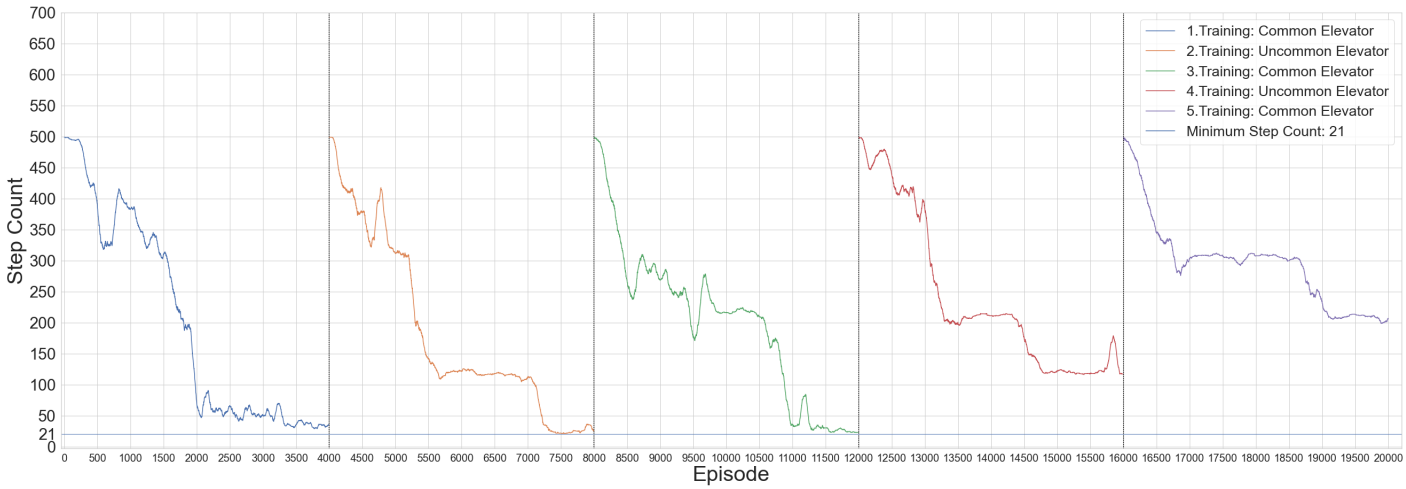

Figure 5.3: Mean step count results for each DQN training per episode

with identical line styles(solid or dashed). Figure 5.4 compares the different training sections trained with the same algorithm. In reality, the trainings are run sequentially.

The comparison between trainings demonstrates that the performance of the DQN algorithm for the HER task decreases for each sequent training session. However, the agent can find the optimal policy in the first three training sections. The differences in the performance rankings in the first 3000 episode can be confusing. However, the

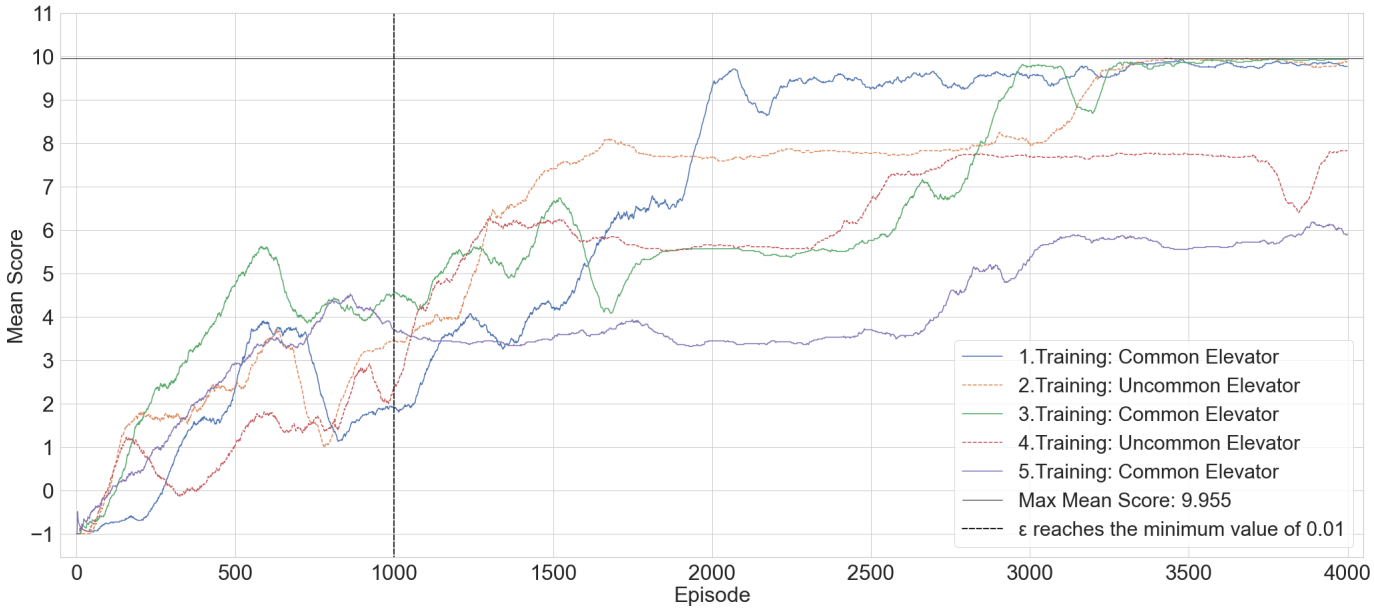

Figure 5.4: DQN mean scores, all training in a single plot

most important goal of the reinforcement learning agent is improving the best policy to solve the task. In other words, the convergence score is the most significant criterion for habituation. Figure 5.4 demonstrates that the first three training converges for the best score. However, the fourth training mean scores converge to a limit of 8, whereas the fifth training converges to a mean score of 6.

Successor Representation Training Results

Figure 5.5 and 5.6 demonstrate the results of the five sequent Successor Representation(SR) trainings. In the first training, the mean score converges to a maximum mean score of 9.959 after the 1000th episode. Correspondingly, the step count converges to the minimum step count value of 19. In addition, the mean score line in 5.5 indicates that the agent in the second training improves the optimal policy for the room-finding task approximately after the 6000th episode, which is also supported by the convergence to the minimum step count after the same point 5.6. In the third training, reaching the maximum point occurs in the 10.500th episode. However, the step count cannot hit the lowest step count value of 19. The mean scores of the agent in the fourth training almost converge to the highest score. However, an increase in the standard deviation between the training seeds indicates that one of the random seeds cannot find the optimal policy to obtain the best score of 9.959. Figure 5.5 indicates that the mean

Figure 5.5: Mean scores of each SR training per episode, including standard deviation of five different random seeds as shaded area

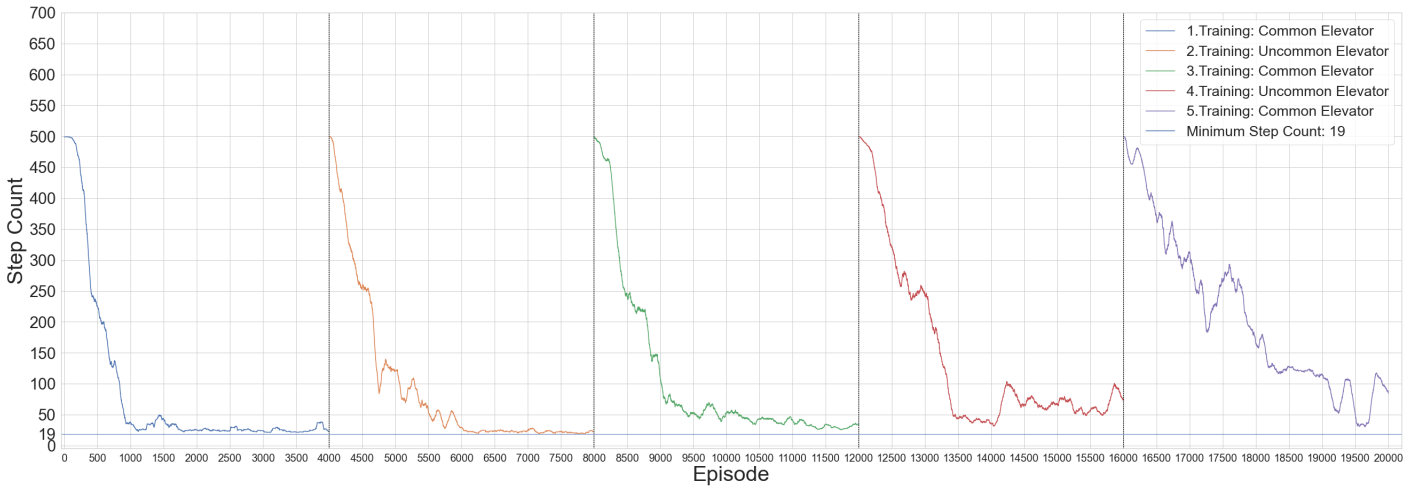

Figure 5.6: Mean step count results for each SR training per episode

scores converge to the best score approximately between episodes 19500 and 19750 in the fifth training. However, in the last 250 episodes, the mean score values deviate from the maximum score of 9.959. In addition, the fifth training has a significantly broader standard deviation than other trainings.

Figure 5.7 illustrates all SR training sections. According to figure 5.7, the first three training converge to the best score of 9.959. Despite the difference in performance rankings between the first three training in the first 1000th episode, where the epsilon

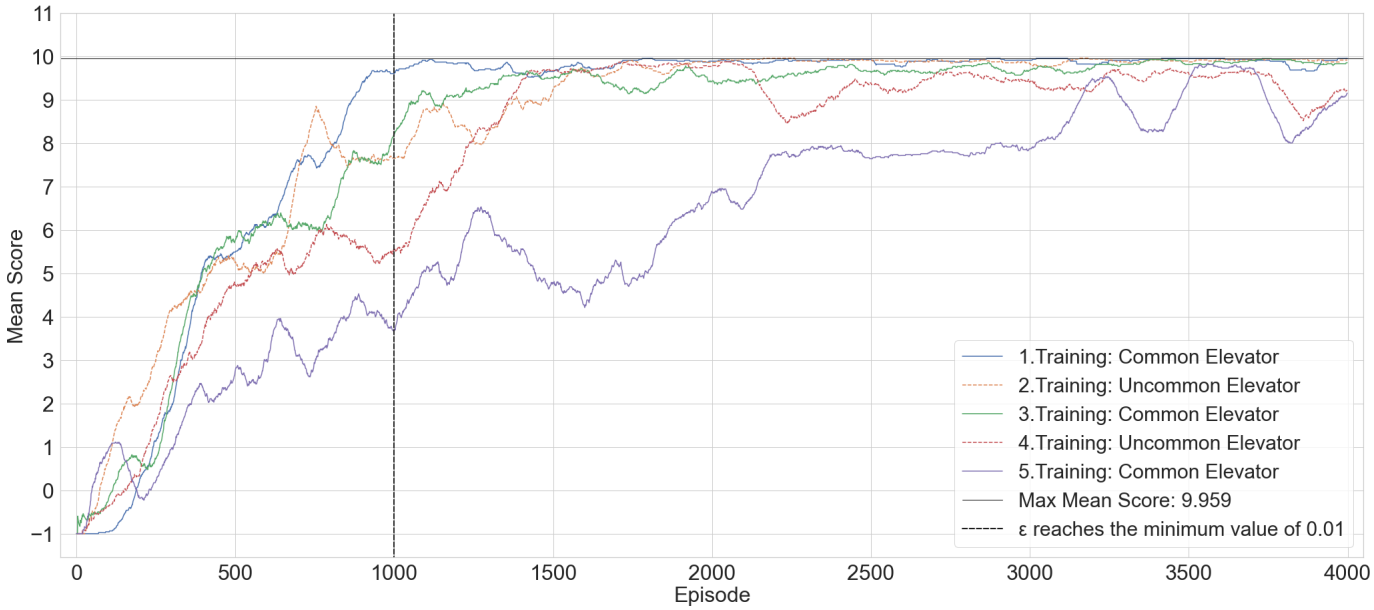

Figure 5.7: SR mean scores, all training in one plot

value reached the lowest value of 0.01, there are no significant differences between the performance of these algorithms in the last 2500 episode. The fourth training almost reaches the best mean score after the episode 1500. In addition, the fifth SR training converges the performance of the fourth training approximately after its 3500th episode. As figure 5.7 outlines, the fourth and fifth training have almost the same performance between their 3500th and 4000th episode.

DQN and Successor Representation Comparison

In figure 5.8, both DQN and SR algorithms are provided to compare performances. Figure 5.8 demonstrates that agent training with the SR algorithm finds the optimal policy in earlier episodes than the DQN agent in the first three training sections. In the fourth training, the SR algorithm almost converges for the best score, whereas the DQN mean score cannot exceed the value of 8. In the fifth training of the experiment, even though the DQN and SR have intimate performances between 16000th and 17000th episodes, after the 17000th episode, SR outperforms compared to DQN. During the last 500 episodes, the SR algorithm fluctuates between 8 and the maximum mean score values. However, the DQN algorithm converges to a mean score of 6.

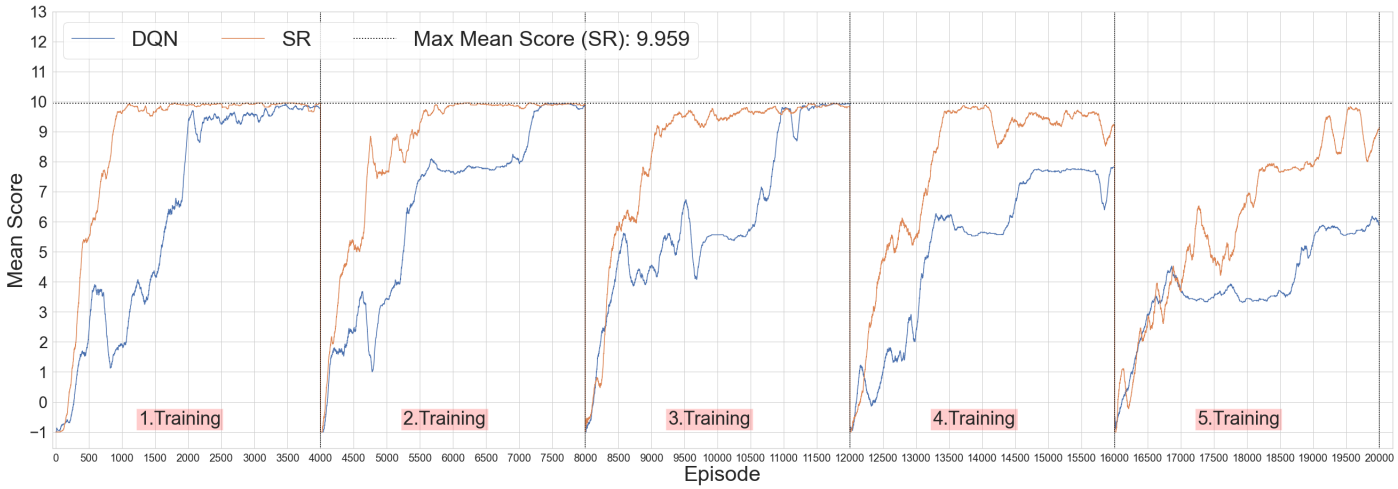

Figure 5.8: DQN and SR mean scores per episode

As a result, the SR agent outperforms to DQN agent against several changes in the starting point of the agent. In addition, the agent with the SR algorithm performs better in the room-finding task, even in the first training, which denotes habitation. Based on our assumptions, an increase in the episode amount or tuning the epsilon parameter might give rise to a definite convergence in the SR algorithm for the fourth and fifth training since fluctuation in the SR algorithm continues. However, figure 5.8 demonstrates that in the fourth and fifth algorithms, the DQN algorithms already converge below the maximum mean score.

5.2.2 Testing Results

After each training session, two tests for different starting locations in the environment are conducted: "test on the trained elevator" and "test on the not-trained elevator". In section 5.2.2, the DQN and SR testing mean scores of the "test on the trained elevator " are compared for the rehabituation performances. Trajectory error occurrences of "test on the not-trained elevator" testing are compared with human behavioural experiment's trajectory error occurrences in section 5.2.2.

Figure 5.9: Mean trial duration for each trial with standard deviation [NS22]

Test on Trained Elevator: Rehabituation Performances

Figure 5.9 outlines the mean trial duration from elevator door opening until the end of the trial for each trial with standard deviation in the human experiment [NS22]. In the human behavioural experiment, the 13, 18, 23, and 28th trials are specified as uncommon trials. As mentioned in the section 3.5, time duration negatively correlates with habitual behaviour. Therefore, Neto performs a two-sample Kolmogorov-Smirnov statistical test on the time duration for the uncommon trials. According to the statistical test results, only the first uncommon trial (13th trial) comes from different distribution from the others. Because of this reason, only the first uncommon trial is considered goal-directed behaviour. Based on the two-sample Kolmogorov-Smirnov statistical test results on time duration data, Neto hypothesizes that humans are one-shot learners by going further than Sanchez's "humans are few-shot learners" hypothesis [San20].

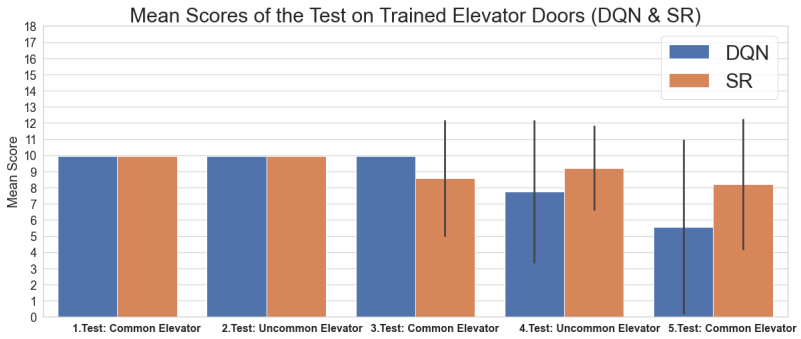

Figure 5.10: Mean scores of the "test on trained elevator" of DQN and SR with standard deviation error bars.

Test Name	DQN mean score	SR mean score
1.Test: Common Elevator	9.954	9.956
2.Test: Uncommon Elevator	9.958	9.961
3.Test: Common Elevator	9.956	8.579
4.Test: Uncommon Elevator	7.767	9.205
5.Test: Common Elevator	5.567	8.199

Table 5.1: Mean score of the "test on trained elevator".

Figure 5.10 and table 5.1 demonstrate the mean scores of the "test on the trained elevator" experiment with two different algorithms in our testing experiment. The introduced mean scores are the average of five different seeds and ten times testing. The mean score of the DQN algorithm decreases in each sequential test. However, the SR algorithm demonstrates a different trend than the DQN algorithm.

We perform the paired t-test on the testings in the identical starting points. The testing mean score results depend on each other since the loaded neural weights of the agents are obtained from sequential training sections. In addition, the dependent variables are approximately normally distributed. Therefore the obtained mean sample data satisfies paired sample t-test conditions. If the p-value of the t-test between paired samples is smaller than the p-value=0.05, the means of the pairs are statistically different. Based on paired t-test results given in table 5.2, the mean of the third training SR is not statistically different from the fifth training's mean in the common elevator environment. The other pairs are statistically different.

Test Pairs	Algorithm	Paired Sample T-Test: p-value	meaning
1.Test-3.Test(Common Elevator)	DQN	0.00201859555245	statistically different
3.Test-5.Test(Common Elevator)	DQN	6.07912961797e-07	statistically different
1.Test-5.Test (Common Elevator)	DQN	6.08464013253e-07	statistically different
2.Test-4.Test (Uncommon Elevator)	DQN	0.00099662537817	statistically different
1.Test-3.Test(Common Elevator)	SR	0.00895538271418	statistically different
3.Test-5.Test(Common Elevator)	SR	0.648611773246	**not statistically different**
1.Test-5.Test (Common Elevator)	SR	0.0035659974251	statistically different
2.Test-4.Test (Uncommon Elevator)	SR	0.0459207156250	statistically different

Table 5.2: Paired t-test results between the test mean scores distribution of the identical environments and algorithms

Test on not-trained elevator doors: Trajectory Errors Occurrences

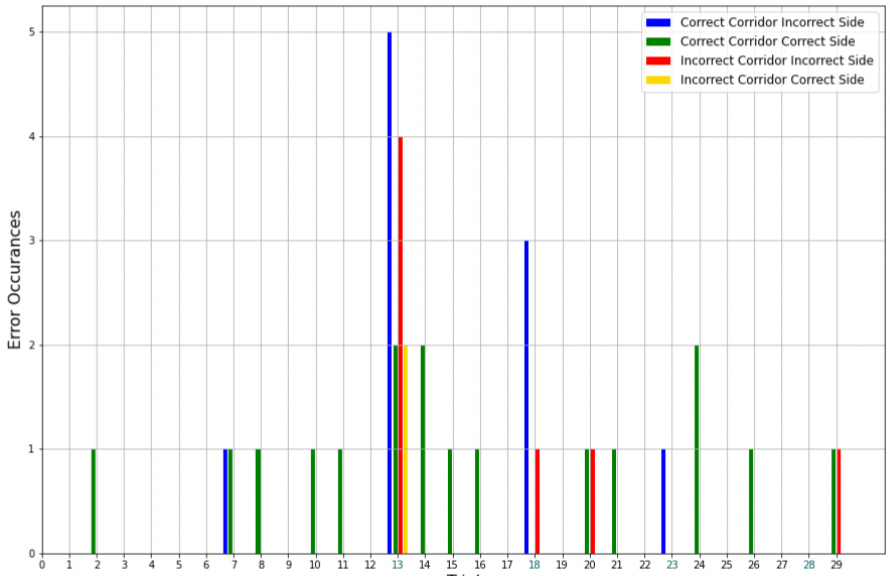

Figure 5.11: Trajectory errors occurrences in the Neto's human behavioural experiment [NS22]

Figure 5.11 shows each trial's trajectory error in the human behavioural experiment. According to Neto, [NS22], the blue type of trajectory error "Correct Corridor, Incorrect Side" results from systematic errors. In addition, she explains that the green type

of error stems from the distraction of the "Looking at the phone" task. Due to the distraction of the "Looking at the phone" task, the participants might turn earlier or later for the correct room. Briefly, the main focus of the human behavioural experiment is the incorrect side of the corridor (red and yellow). In the human behavioural experiment, the red type of error occurs in four trials (13, 18, 20, and 29th), and the yellow type occurs only during a single trial (13th).

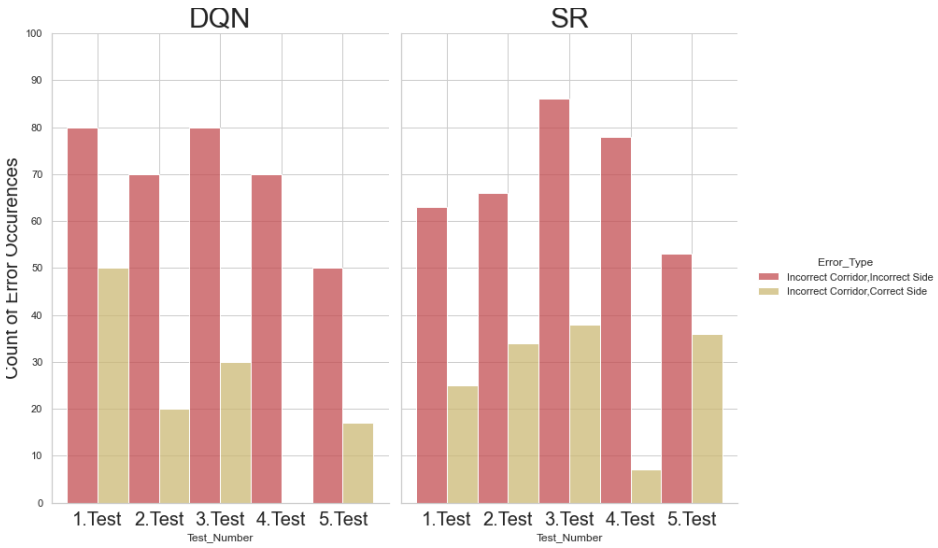

Figure 5.12: Trajectory error occurrences in the "test on the not-trained elevator"

Figure 5.12 demonstrates trajectory error types in our " test on the not-trained elevator" testing experiment. In our experiment, we only observe the red "Incorrect Corridor Incorrect Side" and the yellow "Incorrect Corridor and Correct Side" types of errors. In the DQN test, the red type of error decreases in the second test. Afterwards, this type of error has a downtrend between the third and five tests. However, in the SR test, the number of red types of mistakes increase between the first and third tests. After the third test, this error decreases. In the discussion section, we focus only on the red type error since it has the increase and decrease trend based on our results. Secondly, it is due to the yellow error that occurs once in the human behavioural experiment. Therefore, we do not have sufficient human trajectory error data to compare with the yellow type of trajectory error.

5.3 Discussion

Based on the training experiment results, the learning agent trained with the DQN algorithm finds the optimal policy for the first three training episodes in the last 2000th, 1000th, and 500th episodes. Normally, MF reinforcement learning algorithms are expected to be insensitive to changes. However, the epsilon value is readjusted to the maximum value of 1.0 at the beginning of each training session. Thus, the agent has sufficient time to improve the new policy for the change at the starting point. However, in the second and third training, higher standard deviation between the different random seeds indicates that the MF algorithm cannot even learn the optimal policy consistently like in the first training. Moreover, the agent cannot find the optimal policy in the fourth and fifth training. The performance of the MF algorithm dramatically decreases in each following training. This significant falling trend results from the value caching structure of MF-learning. Due to this structure, the learning agent deals with habitation failure that causes a delay in learning in the second and third training sessions and an inability to find the optimal policy in the fourth and fifth training sessions. This result demonstrates that the rehabituation task cannot be modelled with model-free learning. Secondly, the hypotheses of the [Rus+17; I+] that the MF algorithm is not sensitive against revaluation tasks is evidenced in a 3D environment.

On the other hand, the SR algorithm converges in the last 3000th, 2000th, and 750th episodes for the optimal policy for the room-finding task. The agent trained with the SR algorithm reaches the peak of the best mean score of 9.959 in each training session. We assume that if we run the training with more episodes or readjust the epsilon decay, the fourth and fifth training might converge for the best policy. Due to the symmetrical design of the 3D Her environment, the agent's starting point change in the room-finding task is considered both a reward revaluation task and a basic transition revaluation task. Therefore the SR training experiment evidence also proves the hypotheses of the [Rus+17; I+] that SR is robust against changes in the reward and basic alterations in the transition structure in a 3D environment.

The low standard deviation between different seeds in the second, third and fourth training indicates that SR provides consistent training results for the reward and basic transition revaluation tasks. In addition, the ability of the second, third, fourth, and fifth SR training sessions to reach the best mean score indicates the plausibility of the SR-monitored MF habitual behaviour against changes proposed by Sanchez [San20]. However, inconsistency between different seeds in the fifth SR training refers to the requirement of the model-based inference system against continuously occurring environmental changes.

The SR algorithm can solve the room-finding task with less minimum step count of 19 than the DQN minimum step count of 21. The reason for this result is SR's semi-model-based structure. SR algorithm models the environment through its expected future state occupancies matrix M. Thus, SR can find a shorter path for this task. However, initial habitual behaviour cannot be optimally modelled by SR since modelling habitual behaviour in high-dimensional problems might suffer from the curse of dimensionality due to SR's semi-model-based structure.

The significant increase in a trial duration is considered the indicator of the deliberative behaviour in human behavioural experiments [NS22]. The findings in human behavioural experiments indicate that the human participants behave deliberately (significant increase in time duration) only in the first uncommon trials based on the statistical tests applied to the mean trial duration for each trial. This result causes an assumption that humans learn to rehabituation; in other words, they habituate to rehabituation. Based on t-paired test results performed on the SR "test on the trained elevator" experiment, the mean score of the first rehabituation task (3.Test: Common Elevator) decreases significantly compared to the habituation task (1.Test: Common Elevator). However, the mean scores of the first and second rehabituation tasks (5. Test: Common Elevator) are not statistically different. We infer from these findings that the SR algorithm maintains the same rehabituation performance, whereas the rehabituation performance of DQN gradually decreases. We assume that if more tests are run, this stable trend in the SR algorithm's rehabituation can be more clarified.

The "test on the not-trained elevator" testing experiment is conducted to identify habituation trajectory anomalies in the 3D HER environment. According to our results, the red type of trajectory error has a decreasing trend after the third test for both algorithms. Then an upward trend in the SR test between the first and third tests is observed. We aim to compare these results with the identically defined trajectory errors with the human behavioural experiment in uncommon trails 5.11. This error occurs in the human behavioural experiment four times in the first uncommon trial and once in the second uncommon trial. Despite the decrease in the red type error, we cannot find the comparable amount of the red type of error and trend for this type of error to compare with our data. We assume that if the human behavioural experiment is conducted with the same structure, more participants and less systematic error, the obtained trajectory error data can be compared better.

6 Conclusion

6.1 Summary

One of the human decision-making mechanisms, habitual behaviour, relates to the automatic response of the decision-making system. The other mechanism, goal-directed behaviour, is flexible against environmental changes. Both mechanisms cannot take control simultaneously due to their different characteristics and suitability for the task. This situation raises the need for an arbitration mechanism between these two systems. Through this arbitration mechanism, goal-directed behaviour takes control when a habituation failure due to environmental changes occurs. To model this arbitration mechanism, Sanchez [San20] introduces an alarm mechanism of Successor Representation monitoring model-free behaviour in the Hotel Elevator Row(HER) environment, which causes habituation failure by changing starting points. Afterwards, Neto [NS22] conducts a virtual reality experiment in the Hotel Elevator Row(HER) environment to identify the habituation failures of human participants.

In this work, we designed a 3D Unity version of the Hotel Elevator Row(HER) environment to investigate the proposed alarm mechanism system[San20]. We then used the environment by training the learning agent by employing the Deep Q-learning(DQN) and deep Successor Representation(SR) reinforcement learning algorithms. At the same time, we evaluated the sensitivity of Deep Q-learning(DQN) and Successor Representation(SR) for the reward and basic transition revaluation task in the 3D design of the Hotel Elevator Row(HER) environment. Finally, we compared habituation failures with Neto's human behavioural experiment [NS22] trial duration and trajectory error occurrences per trial.

As a result, we concluded that Successor Representation provides a possible monitoring alarm system in 3D Hotel Elevator Row(HER) environment against habituation failures on model-free behaviour due to its robustness against reward and basic transition revaluations tasks. Secondly, we found that Successor Representation might be a plausible mechanism to investigate humans' habituation to rehabituation behavioural patterns. Finally, we observed a trending relation between symmetrical trajectory errors in each sequentially conducted test on obtained data from the Successor Representation

trained learning agent.

6.2 Future Work

The 3D Hotel Elevator Rows(HER) environment allows us to test the reward and basic transition revaluation tasks. However, in the [Rus+17; I+], more complex transition and policy revaluation tasks for Successor Representation are investigated. For future work, we suggest designing the second version of the 3D Hotel Elevator Rows(HER) environment, which includes investigating Successor Representation sensitivity against complex transition and policy revaluation tasks. Secondly, the model-based algorithm Dyna-Q can also be employed in future studies to investigate the model-based(MB) inference on Sanchez's alarm mechanism [San20]. Thirdly, the detected similarity in performance of Successor Representation on the first and second rehabituation can be demonstrated for the following fourth and fifth rehabituation. Finally, we suggest re-conducting human behavioural experiments with more participants to compare with our trajectory error findings in Deep Q-learning(DQN) and Successor Representation.

Bibliography

[A] D. A. "Actions and habits: the development of behavioural autonomyPhil. Trans. R. Soc. Lond. B30867–78." In: (). DOI: 10.1101/083824.

[Alv+08] A. Alvernhe, T. V. Cauter, E. Save, and B. Poucet. "Different CA1 and CA3 Representations of Novel Routes in a Shortcut Situation." In: *The Journal of Neuroscience* 28 (2008), pp. 7324–7333.

[BD98] B. W. Balleine and A. Dickinson. *Goal-directed instrumental action: contingency and incentive learning and their cortical substrates*. 1998, pp. 407–419.

[Bel57] R. Bellman. "A Markovian Decision Process." In: *Indiana University Mathematics Journal* 6 (1957), pp. 679–684.

[Bon] Bonniesjli. *Count based exploration with the successor representation for Unity ML's pyramid*.

[Daw+11] N. Daw, S. Gershman, B. Seymour, P. Dayan, and R. Dolan. "Model-Based Influences on Humans' Choices and Striatal Prediction Errors." In: *Neuron* 69 (Mar. 2011), pp. 1204–15. DOI: 10.1016/j.neuron.2011.02.027.

[Daw18] N. D. Daw. *Are we of two minds?* Nov. 2018. DOI: 10.1038/s41593-018-0258-2.

[Day] P. Dayan. *Improving Generalisation for Temporal Difference Learning: The Successor Representation*.

[DND05] N. D. Daw, Y. Niv, and P. Dayan. "Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control." In: *Nature Neuroscience* 8 (12 Dec. 2005), pp. 1704–1711. ISSN: 10976256. DOI: 10.1038/nn1560.

[GD13] R. Grieves and P. Dudchenko. "Cognitive maps and spatial inference in animals: Rats fail to take a novel shortcut, but can take a previously experienced one." In: *Learning and Motivation* 44 (May 2013), pp. 81–92. DOI: 10.1016/j.lmot.2012.08.001.

[Gef18] H. Geffner. "Model-free, Model-based, and General Intelligence." In: (2018). DOI: 10.48550/ARXIV.1806.02308.

[Ger+12] S. Gershman, C. Moore, M. Todd, K. Norman, and P. Sederberg. "The successor representation and temporal context." English (US). In: *Neural Computation* 24.6 (2012), pp. 1553–1568. ISSN: 0899-7667. DOI: 10.1162/NECO_a_00282.

[I+] M. I, R. Em, C. Jh, B. Mm, D. Nd, and G. Sj. "The successor representation in human reinforcement learning." In: (). DOI: 10.1101/083824.

[Jul+18] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange. "Unity: A General Platform for Intelligent Agents." In: (Sept. 2018).

[Jul19] A. Juliani. *The present in terms of the future: Successor representations in reinforcement learning*. Nov. 2019.

[Kah11] D. Kahneman. *Thinking, fast and slow*. New York: Farrar, Straus and Giroux, 2011. ISBN: 9780374275631 0374275637.

[LC06] C. Landisman and B. Connors. "Long-Term Modulation of Electrical Synapses in the Mammalian Thalamus." In: *Science (New York, N.Y.)* 310 (Jan. 2006), pp. 1809–13. DOI: 10.1126/science.1114655.

[LG08] B. Lau and P. Glimcher. "Value Representations in the Primate Striatum during Matching Behavior." In: *Neuron* 58 (June 2008), pp. 451–63. DOI: 10.1016/j.neuron.2008.02.021.

[LSO14] S. W. Lee, S. Shimojo, and J. P. O'Doherty. "Neural Computations Underlying Arbitration between Model-Based and Model-free Learning." In: *Neuron* 81 (3 Feb. 2014), pp. 687–699. ISSN: 08966273. DOI: 10.1016/j.neuron.2013.11.028.

[MBB18] M. C. Machado, M. G. Bellemare, and M. Bowling. *Count-Based Exploration with the Successor Representation*. 2018. DOI: 10.48550/ARXIV.1807.11622.

[Mni+13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. DOI: 10.48550/ARXIV.1312.5602.

[Mog19] P. H. Moghadam. *Deep reinforcement learning: DQN, double DQN, dueling DQN, noisy DQN and DQN with prioritized...* July 2019.

[NS22] L. F. Neto and M. Sànchez-Fibla. *Master thesis on Cognitive Systems and Interactive Media Behavioral Correlates of Habituation and Deliberation A virtual reality study with the Hotel Elevator Rows task*. 2022.

[Rus+17] E. M. Russek, I. Momennejad, M. M. Botvinick, S. J. Gershman, and N. D. Daw. "Predictive representations can link model-based reinforcement learning to model-free mechanisms." In: *PLoS Computational Biology* 13 (9 Sept. 2017). ISSN: 15537358. DOI: 10.1371/journal.pcbi.1005768.

[San20] M. Sanchez-Fibla. "HABITUATION AND GOAL-DIRECTED ARBITRATION MECHANISMS AND FAILURES UNDER PARTIAL OBSERVABILITY." In: (2020). DOI: 10.1101/2020.11.24.396630.

[SB] R. S. Sutton and A. G. Barto. *Reinforcement learning : an introduction*, p. 526. ISBN: 9780262039246.

[SBG14] K. Stachenfeld, M. Botvinick, and S. Gershman. "Design principles of the hippocampal cognitive map." In: *Advances in Neural Information Processing Systems* 3 (Jan. 2014), pp. 2528–2536.

[SBG17] K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman. "The hippocampus as a predictive map." In: *Nature Neuroscience* 20 (11 2017), pp. 1643–1653. ISSN: 15461726. DOI: 10.1038/nn.4650.

[Sch+13] A. Schapiro, T. Rogers, N. Cordova, N. Turk-Browne, and M. Botvinick. "Neural representations of events arise from temporal community structure." English (US). In: *Nature Neuroscience* 16.4 (Apr. 2013), pp. 486–492. ISSN: 1097-6256. DOI: 10.1038/nn.3331.

[SG15] H. J. Spiers and S. J. Gilbert. *Solving the detour problem in navigation: A model of prefrontal and hippocampal interactions*. Mar. 2015. DOI: 10.3389/fnhum.2015.00125.

[SL14] H. Shteingart and Y. Loewenstein. "Reinforcement learning and human behavior." In: *Current opinion in neurobiology* 25C (Apr. 2014), pp. 93–98. DOI: 10.1016/j.conb.2013.12.004.

[Ten+11] J. Tenenbaum, C. Kemp, T. Griffiths, and N. Goodman. "How to Grow a Mind: Statistics, Structure, and Abstraction." In: *Science (New York, N.Y.)* 331 (Mar. 2011), pp. 1279–85. DOI: 10.1126/science.1192788.

[Tol48] E. C. (Tolman. "Cognitive Maps in Rats and Men." In: *Psychological Review* 55 (1948), pp. 189–208. DOI: http://dx.doi.org/10.1037/h0061626.

[Uda] Udacity. *Deep-reinforcement-learning/DQN at master · udacity/deep-reinforcement-learning*.

[VS19] E. Vertes and M. Sahani. "A neurally plausible model learns successor representations in partially observable environments." In: (June 2019).

[Wat89] C. Watkins. "Learning From Delayed Rewards." In: (Jan. 1989).

[Wil] T. Williams. *Reinforcement learning vs. Deep Reinforcement Learning*.