Sequential Replay and Network Instability in Deep Reinforcement Learning

by

Sebastian Jentgens
sebastian.jentgens@fu-berlin.de
Matriculation number: 5567685

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of

Bachelor of Computer Science

at

Freie Universität Berlin

Supervisor: M. Sc. Nicolas Diekmann, Institut für Neuroinformatik, Ruhr-Universität Bochum
nicolas.diekmann@ini.rub.de

Examining Board: Prof. Dr. Daniel Göhring, Institut für Informatik, Freie Universität Berlin
daniel.goehring@fu-berlin.de

Prof. Dr. Sen Cheng, Institut für Neuroinformatik, Ruhr-Universität Bochum
sen.cheng@rub.de

Contents

1 Introduction

The hippocampus is a neural structure in the limbic system of mammalian brains that was demonstrated to play an important role in both spatial and recognition memory (Broadbent et al., 2004). In the past, the neural correlate of spatial learning has been investigated comparatively well in rats and mice which exhibit profound impairment in spatial cognition when hippocampal lesion is inflicted (Deacon et al., 2002; Morris et al., 1982). Moreover, rats' hippocampi have been reported to host a type of neuron, place cells (O'Keefe & Dostrovsky, 1971), that show increased activation whenever the rat encounters specific regions in its environment (the place field associated with a cell). Interestingly, the close link between individual place cells and their place fields is not restricted to awake states. It was shown that sequential activation patterns of these neurons, that represent trajectories taken by a rat in its environment, are reactivated during different phases of sleep (Lee & Wilson, 2002; Louie & Wilson, 2001). Further, in learning paradigms in which rats had to navigate mazes, it was shown that these sequences are reactivated in reverse order in awake states immediately after phases of intense interaction with the environment (Foster & Wilson, 2006). According to Complementary Learning Systems Theory (CLS) (Kumaran et al., 2016; McClelland et al., 1995), this mechanism, coined replay, benefits learning by enabling the alternating presentation of old and novel information. CLS postulates that this kind of interleaved training reduces catastrophic interference (McCloskey & Cohen, 1989), the loss of embedded knowledge by overwriting it with incoming information. While this claim conceptualizes the role of replay from a functional perspective, several decades later, a detailed explanation of how the replay mechanism works, is still missing. However, deep reinforcement learning (RL), a paradigm within the field of artificial intelligence, turns out to be a promising tool for dismantling the mechanisms of hippocampal replay. Similar to biological learning, in deep RL, an agent that is represented by a neural network, e.g. a Deep Q-Network (DQN) (Mnih et al., 2013), learns by interacting with its environment. Although, the idea of applying replay buffers in order to accelerate or stabilize the training is not new in RL (Lin, 1992), the artificial replay mechanisms used in the past were not biologically plausible and thus could not be used to draw insights into hippocampal replay. The recently presented *Spatial structure and Frequency-weighted Memory Access*-algorithm (SFMA) (Diekmann & Cheng, 2023) on the other side, simulates the biological replay mechanism by sampling experiences from a memory module with respect to their strength, their inhibition as well as to their similarity to a previously replayed experience. Depending on these factors, SFMA assigns a priority score to experiences that a RL

agent made when interacting with its environment and uses them to define a probability distribution over all experiences according to which experience sequences are sampled for replaying. The skewness of this distribution can be adjusted by parameter modulation which directly effects whether replayed sequences correspond more or less to the agent's preceding trajectories through its environment. While it was shown that for small replay batches, more deterministic replay can improve a DQN's performance compared to completely randomly sampled sequences, it seems that a certain amount of stochasticity remains necessary in order to compete with random replay (Zeng et al., 2023). This leaves two important questions that will be investigated in this thesis paper:

1. In what regard does more deterministic SFMA destabilize the training of a DQN?

2. To what extend can instability be linked to specific pattern in the generated replay sequences?

These questions will be investigated by paying close attention to the aforementioned phenomenon of catastrophic interference (CI), which is strongly related to instability in the training of artificial neural networks (ANN). The term was first used to describe an observation of ANNs that are trained continuously on multiple tasks (McCloskey & Cohen, 1989). Once trained on a new task, previously acquired knowledge is forgotten. This is a result of the adjustment of an ANN's weights w.r.t. the drifted data distribution in the interim learning phase. While CI is best understood in this multitask Continual Learning (CL) paradigm, it is likewise prevalent in deep RL, even in many single-task settings with a fixed environment where it is associated with performance degradation (Fedus et al., 2020; Liu et al., 2019; Lo & Ghiassian, 2019; Schaul et al., 2019). The underlying reason behind this is that training an ANN often assumes identically and independently distributed (i.i.d.) data. It is obvious that this assumption does not hold in multitask CL. However, due to the specific learning dynamics in RL, that are thoroughly explained in the following section, it is not met in RL either (Khetarpal et al., 2022). Likewise, the characteristic sequentiality of samples replayed by more deterministic SFMA introduces correlation to the data that an agent is trained on. It is therefore reasonable to assume that adjustments of the stochastic properties of SFMA have an effect on the occurrence of CI, if the agent is represented by an ANN. If this is the case, due to the negative effect of CI on performance, there should be a link between the performance that results from a parameter setting and the amount of CI that the ANN is subject to. This is the hypothesis with regard to the first research question.

Regarding the second research question, I expect to find that information that is underrepresented in replay is generally more prone to be lost in the process of CI than information that is encountered more frequently.
The measure for CI that is applied in the analysis, as well as an introduction to deep RL and its specific learning dynamics is provided in the first section of chapter 2. Afterwards, the mechanics of SFMA and the experimental setup are laid out, followed by the experimental results and their evaluation in chapter 3. Finally, I will conclude by summarizing my findings and leading direction for future work.

2 Methods

2.1 Deep Reinforcement Learning

Reinforcement learning (RL) (Sutton & Barto, 2018) is a machine learning paradigm in which an agent learns to interact optimally with an environment. Each interaction is described as tuple (s_t, a_t, r_t, s_{t+1}), where r_t is the reward gained by the agent in time step t after transitioning from state s_t to state s_{t+1} by taking action a_t. The behavior of the agent is determined by the policy π. $\pi(s)$ is a a probability distribution that defines the likelihood of choosing an action a when the agent is in state s. The goal of RL is to find the optimal policy π^* that maximizes the expected *discounted return*[1]:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{1}$$

Where $\gamma \in [0, 1]$ is the factor that discounts future reward. In *Q-learning* (Watkins, 1989), a well known approach to solving the RL-problem, an *action-value* function Q_π is learned that maps each (s, a)-tuple to the (discounted) cumulative reward that is expected when taking action a in state s and following policy π forever after:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{2}$$

As the best possible policy is always the one that maximizes the expected reward, the following property, described by the Bellman equation, holds[2]:

$$Q_{\pi^*}(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'}(Q_{\pi^*}(s', a'))] \tag{3}$$

[1] In an episodic RL-task, T is the maximum number of steps per episode whereas in continuous RL-tasks, $T = \infty$.

This equation is used in Q-learning to iteratively update the Q-function in each interaction step according to the rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) - \alpha \underbrace{[\underbrace{Q(s_t, a_t)}_{\text{prediction}} - \underbrace{(r_t + \gamma \max_{a'}(Q(s_{t+1}, a')))}_{\text{target}}]}_{\text{TD-error}} \tag{4}$$

We call this method of updating Q via the temporal difference (TD) error w.r.t. an estimated target value bootstrapping. It was proven that, when the Q-values are represented discretely, i.e. in a tabular form, Q-learning converges towards Q^*[3](Watkins & Dayan, 1992).

In the domain of deep RL, however, the Q-function is approximated by a deep neural network (DNN) (Lecun et al., 1998). DNNs are multivariate, non-linear and differentiable functions that can be optimized iteratively by gradient descent to find the global minimum of a loss function $\mathcal{L}$ that defines the model's prediction error w.r.t. a set of target values. In each update step, after calculating the loss, the DNN's parameters θ can be updated by subtracting their gradients $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}}{\partial \theta}$, where the learning rate α determines the size of gradient descent steps. The gradients can efficiently be calculated via the backpropagation algorithm (Rumelhart et al., 1986) that is readily implemented in modern ML-software. Under the premis that learning parameters, i.e. α are selected correctly, convergence towards a local minimum of $\mathcal{L}$ is assured. There exist several methods in machine learning to prevent a DNN from getting stuck in a local minimum of $\mathcal{L}$. However, deep RL is particularly prone to instability in the learning process. This is due to multiple reasons. Firstly, since the DNN is updated during training, the distribution of target values that is used for bootstrapping, is shifted. We call this characteristic non-stationarity (Khetarpal et al., 2022). Therefore, deep RL algorithms often rely on an additional DNN that serves solely as approximation of the target values (Mnih et al., 2015). This target network is updated less frequently to give the online network, the DNN that determines the agent's behavior, a chance to converge. Still, this trick does not fully eradicate non-stationarity. Secondly, in most Deep RL algorithms, updates can be performed only for those transitions that the agent already experienced which potentially leads to a strong correlation of training data in update series which in turn poses a threat for proper convergence. In order to dissolve this correlation, modern RL algorithms, such as Deep Q-learning

[2]Note that the expected value is used only for probabilistic environments in which $p(s'|s, a)$ defines the probability of transitioning to state s' when taking action a in state s.

[3]From now on, Q^* is used throughout all chapters to abbreviate Q_{π^*}.

(DQN) (Mnih et al., 2015) often apply a method called experience replay. By storing experience tuples in a memory buffer $\mathcal{B}$ and updating the network weights θ on experiences randomly sampled from it, DQN has been shown to be the first artificial agent able to perform on a human level on a diverse set of tasks.

The instability that may arise due to the learning characteristics described in the preceding paragraph, are recently gaining more attention and novel measures that link learning instabilities to the phenomenon of CI have been proposed to study it (Liu et al., 2020, 2023). A particularly interesting approach that allows exploring instabilities in the training process both on the level of single transitions as well as on the level of training episodes, is the application of *Accuracy Change* and *Update Interference* proposed by Liu et al., 2023. Adapted versions of these measures were applied in this study. Very similar to the Accuracy Change, the interference for a specific state and action pair (s, a) that resulted from updating the model weights θ in between two points in time t, was measured by:

$$CI_{sa} \coloneqq \mathbb{E}_{e\in\mathcal{E}}[\max(MSE_{s,a}(Q_{\theta+1}, Q^*) - MSE_{s,a}(Q_\theta, Q^*), 0)] \tag{5}$$

where MSE is the mean squared error of the online Q-function w.r.t. the optimal Q-function Q^* and $\mathcal{E}$ is the set of all experimental runs. Briefly, CI_{sa} measures to what extend an update negatively effected the DQN's knowledge of its environment (on average). Contrary to Accuracy Change, CI_{sa} considers only positive values which was necessary in order to average the measure over multiple experimental runs. In addition, similar to Update Interference, to what extend and update affected the deviation of the Q-function from Q^* was measured by

$$CI_e \coloneqq \mathbb{E}_{e\in\mathcal{E}}[\max(MSE_{\mathcal{S},\mathcal{A}}(Q_{\theta_{t+1}}, Q^*) - MSE_{\mathcal{S},\mathcal{A}}(Q_{\theta_t}, Q^*), 0)] \tag{6}$$

where $\mathcal{S}$ and $\mathcal{A}$ denote the sets of all possible states and actions respectively. This function is almost identical to an averaged version of Update Interference except that Update Interference weights each tuple (s, a) according to a distribution d. d can be represented by a buffer that collects an agent's experiences, which allows taking into account the frequency with which the DQN agent faces experiences involving s and a. For the sake of computational efficiency, CI_e simply averages over all transitions instead.

The presented measures were used to analyze the instabilities that arise from the training dynamics of DQN agents that rely on SFMA, a prioritized replay algorithm that will be explained in the following section. The training procedure of these agents followed an adapted version of the original deep

Q-learning algorithm shown in algorithm 1. In each step, the agent selects an action according to an ϵ-greedy policy, that either exploits the knowledge of the agent by choosing the action with the greatest expected discounted return or fosters exploration of the environment by choosing a random action with probability ϵ. At the end of an episode, the Adam optimizer (Kingma & Ba, 2017) updates θ on the mean squared prediction error (MSE) $\mathcal{L}$ w.r.t. a minibatch of experiences sampled by SFMA. CI_{sa} and CI_e calculated with the recorded Q-functions before and after each series of minibatch updates.

Algorithm 1 Adapted deep Q-learning with experience replay

Initialize replay memory $\mathcal{B}$ to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for $episode = 1, M$ **do**
    for $t = 1, T$ **do**
        With probability ϵ select random action a_t
        otherwise select $a_t = \max_a Q(s_t, a; \theta)$
        Execute action a_t and observe reward r_t and state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) and terminal flag in $\mathcal{B}$
        if s_{t+1} is a goal state **then**
            break
        end if
    end for
    Sample minibatch B of experiences (s_j, a_j, r_j, s_{j+1}) from $\mathcal{B}$
    Set

$$y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'}(\hat{Q}(s_{j+1}, a'; \theta^-)) & \text{otherwise} \end{cases}$$

    $\mathcal{L} \leftarrow \mathbb{E}[(y_j - Q(s_j, a_j; \theta))^2]$
    Perform an update step on $\mathcal{L}$ w.r.t. θ ▷ Adam optimizer
    Every C steps reset $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$ ▷ Blend update
end for

2.2 SFMA

While the use of experience replay in DQN is biologically inspired (Mnih et al., 2015), the underlying mechanisms are much more complex in the hippocampus, where replay has been shown to appear in a sequential fashion (e.g. Buhry et al., 2011; Gupta et al., 2010; Louie and Wilson, 2001). SFMA,

in contrast, has proven itself capable of simulating multiple properties that have been observed in hippocampal replay (Diekmann & Cheng, 2023). The algorithm samples experienced transitions from a memory module according to a priority score that considers three factors:

1. The experience strength $C(e)$, which incorporates the frequency of an experience e and the amount of reward associated with it.

2. The similarity $D(e|e_t)$ is an indicator for how close two transitions e and e_t are in state-space.

3. The decaying inhibition $I(e)$ that is applied to prevent the consecutive reactivation of a replayed experience.

The calculation of the priority rating $R(e|e_t)$ as well as the complete algorithm can be found in algorithm 2. In the experiments, the threshold τ was set to 0 which resulted in minibatches of size $N = 32$ for each replay. The reactivation probability $P(e|e_t)$ is calculated as

$$P(e|e_t) = \frac{e^{\beta R(e|e_t)} - 1}{\sum_j (e^{\beta R(e|e_t)} - 1)} \tag{7}$$

Due to the prioritization function, SFMA favors replaying sequences of consecutive experience tuples, that is transitions that the agent made in two consecutive steps. As can be seen in eq. (7), the greater the inverse temperature factor β is, the more deterministic is the selection of experiences and therefore the greater the average length of these sequences. In contrast, for $\beta = 0$, SFMA samples uniformly from the set of state transitions that have been visited by the agent beforehand. This has been reported to result in sequences of minimal length (Zeng et al., 2023). In addition, Zeng et al., 2023 have investigated the effect of β on performance under hyperparameter settings that tend to generate sequences in reverse order. They have shown that, for medium β values, when a small batch size N is applied to replaying, SFMA in reverse mode can improve the performance of a DQN over random replay. This is due to the ability of reverse sequences to facilitate the propagation of reward information (Foster & Wilson, 2006). In contrary, when N was increased, especially for great β, a DQN relying on SFMA performed worse in comparison to one that uses random replay. According to the authors, this can be explained by the inverse relationship between β and the diversity of replayed sequences. When the variance in replayed sequences falls below a minimum value, a DQN cannot profit from the potential advantage, that replaying sequences has over replaying experiences randomly. Building on top of the study by Zeng et al., 2023, in the experiments of this

paper, SFMA was used in online mode with parameter settings that favor the generation of reverse sequences. The online mode is thought of by Diekmann and Cheng, 2023 to better simulate awake hippocampal replay.

Algorithm 2 Spatial structure and Frequency-weighted Memory Access (SFMA) in online mode

Initialize e_t with current state and random action
for $t = 1; N$ **do** ▷ Select a minibatch of size N
  for experience e **do**
    Compute priority rating $R(e|e_t) = C(e)D(e|e_t)[1 - I(e)]$
  end for
  if max $R < \tau$ **then**
    break
  end if
  Compute reactivation probabilities $P(e|e_t)$
  Choose next experience e_{t+1} to be replayed
  Decay the inhibition factor for all stored experiences
  $I(e_{t+1}) \leftarrow 1$
end for

2.3 CoBeL-RL and Experimental Setup

CoBeL-RL (Diekmann et al., 2023) is a computational simulation framework that was developed to investigate the functional role of spatial navigation and extinction learning. It allows simulating spatial learning of rodents by training RL agents in virtual mazes. As can be seen in the schematic view of this framework (fig. 1), a separate module stores topological information of mazes designed with the unity game engine (https://unity.com/) and information flow between the environment and the agent is enabled by the Open AI Gym Interface (https://www.gymlibrary.dev/).

In this study, the same mazes (tunnel maze LV1 and tunnel maze LV4, fig. 4) and a modified version of the CoBeL-RL-based code from Zeng et al., 2023 were used to train DQN-agents that rely on SFMA. In addition to the original action space that allowed movement in the four cardinal directions as well as 90° rotations in clock-wise and anti-clock-wise direction, a reduced action space that allows only forward movement as well as 90°-rotations in both directions was applied. The agents were all trained for 500 episodes which ended either after a maximum of 600 steps or when the goal position was reached. Both environments were deterministic, which means that taking an action a in a specific state s always resulted in a transition to the same

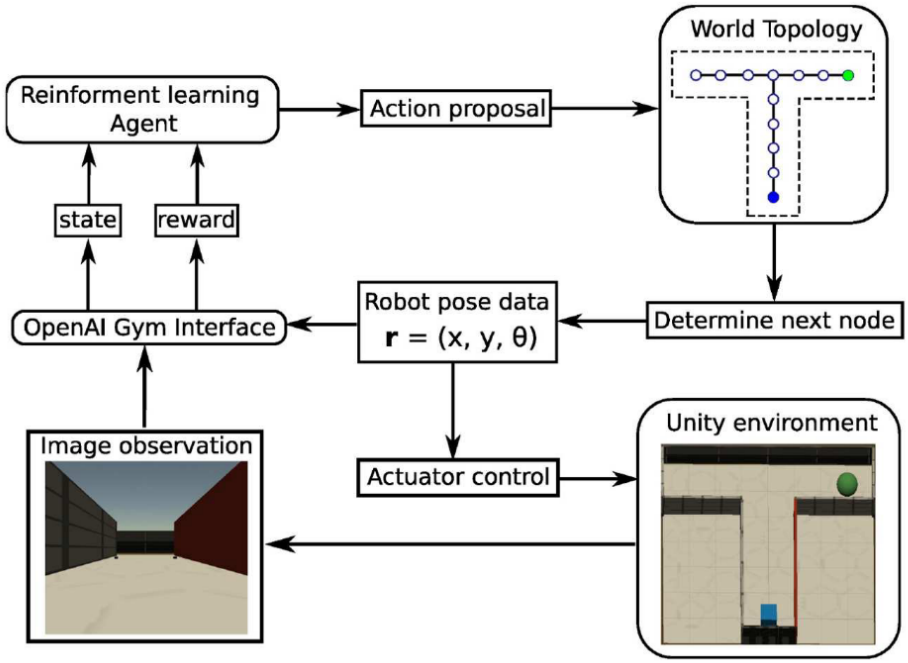

Figure 1: Schematic of the CoBeL-RL simulation framework. Modules are represented by rounded rectangles, data and commands by regular rectangles. The information or control flows are depicted by arrows. Adapted from "Modeling the function of episodic memory in spatial learning", by Zeng, Diekmann, Wiskott, and Cheng, 2023, *Front. Psychol.*, 14:1160648. doi: 10.3389/fpsyg.2023.1160648. Copyright 2023 by the authors under the Creative Commons Attribution License (CC BY).

state s'. Moreover, only transitions to the goal position were rewarded ($r = 1$). In each step, the agent observed the current state either as a $84 \times 84 \times 3$ RGB image or as one-hot encoding. In the former case, the DNN consisted of a convolutional layer with 16 filters, an 8×8 kernel and stride 4 followed by a second convolutional layer with 32 filters and kernel size 4×4 with stride 2. Thirdly, a fully connected layer with 256 units and the output layer with size equal to the size of the action space was used. For one-hot encoded input on the other side, the DNN consisted of only two hidden layers with 256 and 128 units respectively followed by the output layer. In both cases, a Rectified Linear Unit (ReLU) (Agarap, 2018) was used as activation function throughout all layers except for the output layer. A copy of the online network served as target network for bootstrapping. Target networks

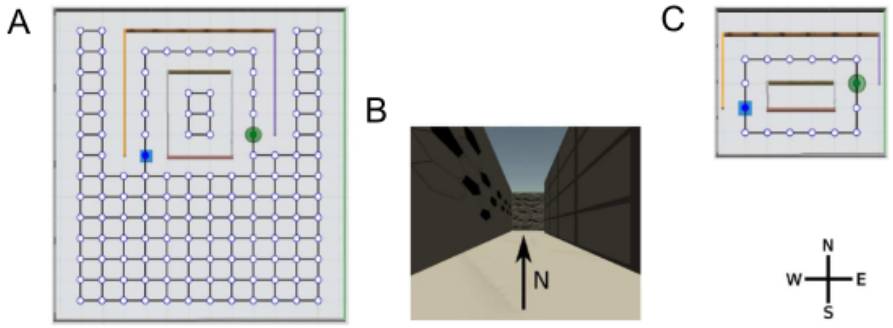

Figure 2: Virtual environments and their topology graphs. Positions allowed by the agent are represented by nodes. At each position, the agent can face one of four directions: north, west, south, east. Transitions between nodes are indicated by solid lines. However, whether a transition is valid depends on the agent's orientation and the applied action space. The starting and goal position (marked green and blue respectively) remain constant throughout all experiments. The topology graph is shown only for demonstration and was not visible to the agent. **(A)** Tunnel maze LV4, **(B)** Example view for the setup with RGB-encoded observations, **(C)** Tunnel maze LV1. Cropped and reassembled from "Modeling the function of episodic memory in spatial learning", by Zeng, Diekmann, Wiskott, and Cheng, 2023, *Front. Psychol.*, 14:1160648. doi: 10.3389/fpsyg.2023.1160648. Copyright 2023 by the authors under the Creative Commons Attribution License (CC BY).

were updated after a series of 50 mini-batch updates on the online network was performed at termination of each episode. In the study by Zeng et al., 2023, among the tested values, this number of mini-batch updates resulted in the greatest performance gap between SFMA with different β values and was therefore thought of to be most interesting to examine. Updates were performed with the Adam optimizer (Kingma & Ba, 2017) on the MSE-loss w.r.t. a minibatch of size 32 sampled by the SFMA-algorithm. For better generalization, two setups with different complexity were tested. The first setup combined tunnel maze LV4 with the full action space and RGB-encoded observations whereas the smaller setup consisted of the much smaller tunnel maze LV1 with one-hot encoded observations and a reduced action space. The code used for the simulations and analysis is available on GitHub (https://github.com/q-uijote/sequential-replay-and-network-instability-in-deep-reinforcement-learning.git).

3 Evaluation

In order to compare network interference for stochastic and more deterministic SFMA, agents with different values for $\beta \in \{0, 1, 2, 5, 10\}$ were trained in the episodic learning paradigm laid out in the preceding chapter. In this chapter, the experimental results will be laid out and analyzed. While the first section examines network instability from a broad, episode-level perspective in order to get a first grasp of overall trends, the second section analyses the relationship between interference and replay patterns more closely on the level of single transitions.

3.1 Catastrophic Interference and Performance

This section focuses on the averaged learning curves and episode-level interference measured in the two setups described in section 2.3. However, results for slightly varying setups as well as convergence of the Q-functions towards the optimal Q^* are reported in the appendix figs. A8 and A9. The first part of this section provides information on the general observations while the second part contextualizes the findings w.r.t. the research question.

Zeng et al., 2023, who have compared SFMA with different values for β only in tunnel maze LV4 with full action space and RGB-encoding but with varying amount of mini-batch updates, found that SFMA with medium β values, that is $1 \leq \beta \leq 5$, learns faster than with $\beta = 0$ or $\beta = 10$. While $\beta = 10$ performs[4] worst in almost all setups that were tested in this study, SFMA with $\beta = 0$ learns fastest in the less complex setup (fig. 3). This observation arises likewise when agents are trained in tunnel maze LV4 with RGB-encodings but with a reduced action space (fig. A8). In addition, both in tunnel maze LV1 and tunnel maze LV4 did switching from the full action space to the reduced action space lead to greater performance gaps between different values for $\beta > 0$[5](figs. A8 and A9). Since an analysis of the relationship between action space and performance is beyond the scope of this study, this observation was not further investigated.

Most importantly with regard to the research question, the learning curves correlate visibly with the occurrence of interference throughout all tested setups. This relationship becomes most pronounced in the simpler setup, for

[4]In this chapter, the word 'performance' always refers to the ability to find the shortest path to the goal position, if not explicitly mentioned otherwise.

[5]The reason why this does not hold for all β is the sudden increase in escape latency in tunnel maze LV1 with full action space and one-hot encodings (fig. A9).

which the scatter plots in fig. 3 show that, on average, interference is greater for longer periods for β values that lead to slower learning. While this is not as clearly visible for other setups due to the overlapping of scatters, the marginally clinging density plots provide more insight. For example, in the more complex setup, a clear shift in the density distribution towards more extreme interference values can be seen for algorithms that learned worse (fig. 3).

More generally, the plots show that CI_e tends to be greater in the more complex setup than in the simple one (fig. 3). This is not surprising since, in a more complex environment with more possible transitions, a smaller fraction of total transitions is replayed when the replay batch remains the same over different setups. While this fraction was not directly measured in this study, this becomes obvious when looking at the replay frequencies reported in figs. A5 to A7, that will be subject of analysis in the following section. In addition to that, as explained in chapter 3, action space size and state encoding have a direct impact on the complexity of the applied DNNs. E.g. the number of hidden layers, which was greater for agents receiving RGB encoded inputs, has been demonstrated by Liu et al., 2023 to positively effect CI in single-task deep RL.

Another, potentially misleading, depiction is the following. The plots in fig. 3 suggest that CI_e is often greater in earlier episodes. A logical explanation for this is that in later episodes, the MSE-loss $\mathcal{L}$ that is used for updating θ, is already much smaller on average as models converge over time. Therefore, updates have a weaker effect on θ which results in only minimal changes in the MSE w.r.t. Q^* between any θ_t and θ_{t+1}. This, indeed, is rather an effect of averaging and it does therefore not imply that CI_e is smaller in later episodes for models that did not already converge.

A similar effect, caused by averaging over multiple experiments, explains why the amount of CI_e can not directly be inferred neither from looking at averaged escape latency curves (figs. 3, A8 and A9) nor when considering the averaged learning curves for the Q-functions itself (figs. A8 and A9). Instead, only looking at individual learning curves, such as the individual learning curves for Q-functions (figs. A8 and A9) reveal performance degradation. This insight provides further evidence that averaging learning curves over multiple runs can lead to delusive results (Gallistel et al., 2004). In contrast, would performance degradation appear consistently in episodes throughout multiple experimental runs, CI would manifest even in averaged curves. Obviously, the ralooking at averagendom weight initialization in each experi-

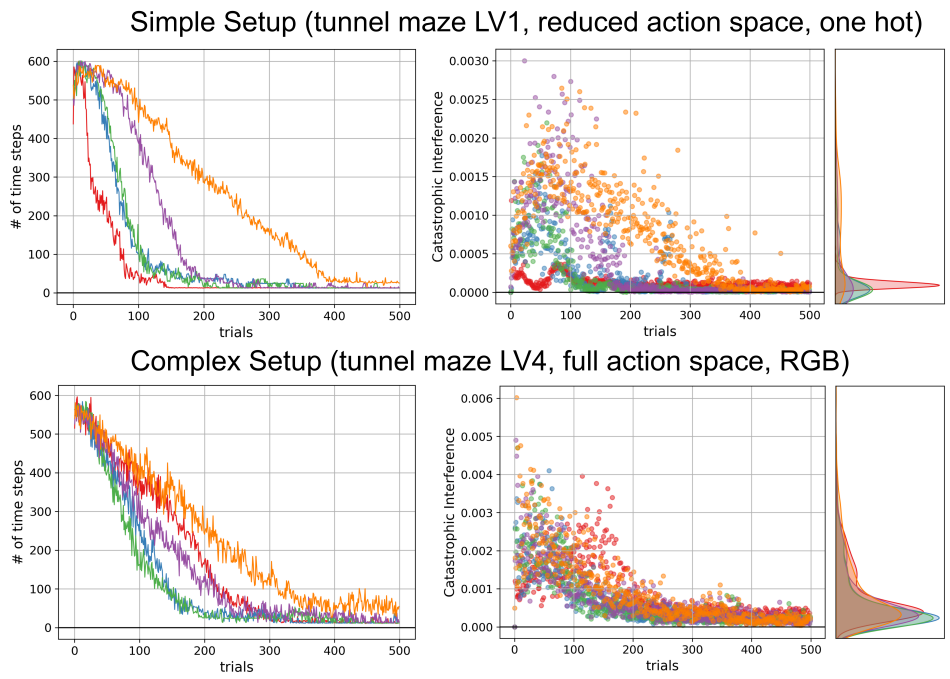

Figure 3: Learning curves (left) and interference (right) for both setups averaged over 50 independent experimental runs.

mental run has an effect on an agent's behavior early in training and might have long term effects that lead to variance in the occurrence of CI. Still, there could be a time-dependent pattern in the occurrence of CI that this study did not make any efforts to reveal. This speculation is supported by a regularity in the occurrence of CI for $\beta = 0$ in tunnel maze LV1 with one-hot encoding and full action space (fig. A9).

In conclusion, the analysis of CI_e shows a clear link between network interference and performance. That is, as expected, algorithms that learn slower exhibit greater levels of interference. In the tested setups, how $\beta = 0$ performs relative to other algorithms, depended on the action space. However, for $\beta > 0$, the ordering of algorithms with different β w.r.t. their performance was consistent throughout all tested setups. Importantly, the density of averaged CI_e values followed the same ordering for all setups. In combination, these observations suggest a link between network instability and the sequentiality of replayed experiences.

3.2 Replay Frequency and Network Instability

In order to explore in more depth why more deterministic SFMA exhibits worse performance, both replay frequency and CI were examined on the level of single transitions. The same setups and data as in fig. 3 were studied except that $\beta = 2$ was omitted in this analysis as its learning curves and CI_e were very similar to the curves of $\beta = 1$.

Before describing and interpreting the results of this analysis, one point should be noted. In the code that was used to plot the replay frequencies of transitions, a bug was detected that caused two anomalies in the plots shown in figs. A5 to A7. Firstly, throughout all figures, one transition in the lower left corner is always among the most replayed transitions. Secondly, for all actions but the forward movement, the figures show no visible variation in the replay frequencies of transitions. Unfortunately, only at the very end of the project phase could the underlying reason for this be identified. Therefore, only the results for transitions involving the forward movement are analyzed in this section, whereas results for other transitions are still reported for transparency. The bug itself is reported and explained in fig. A10.

For better overview, fig. 4 reports results for forward movements in the less complex setup, only. Results for the more complex setup and for other actions are reported in the appendix (fig. A7). The values for transition-level CI and replay frequency were linearly mapped onto the range indicated by the colorbars. The maximum value on the bars corresponds to the maximum 90th and 80th percentile across all $\beta > 0$ respectively. This setting was chosen only for the purpose of better highlighting the differences. As expected, the results for both setups express a visible relationship between the overall replay distribution and transition level CI. For $\beta = 10$, the replay distribution is most skewed and, regardless of the action under consideration, CI_{sa} is greatest for most transitions. On the other side, for $\beta = 0$, replay frequency is most equal among (s, a)-tuples. This, however, does not in general go hand in hand with a lower CI_{sa} when compared with medium β values, that is $\beta \in \{1, 5\}$. Instead, in both setups CI_{sa} is even greater for most transitions. Transitions involving the goal location were more frequently replayed for $\beta = 0$ which, at least in the less complex setup, correlates with a visibly reduced CI_{sa} value for these transitions. This pattern, that we see in the simpler setup but not in the complex setup, corresponds to the relations of learning curves of medium betas and $\beta = 0$ in fig. 3.
Apart from the relationship between replay frequency distribution and overall expression of transition level CI, one cannot directly predict the relative CI_{sa} score for a transition solely from its relative replay frequency. This

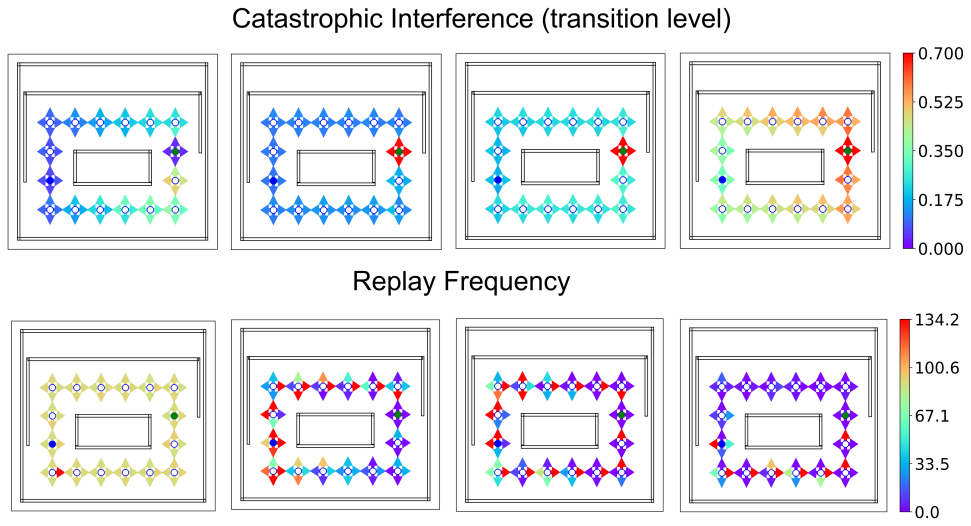

Figure 4: Experimental results for transitions involving forward movements in the less complex setup. The columns represent results for $\beta = 0$,1,5 and 10 respectively. The top row shows catastrophic interference per transition (CI_{sa}) whereas the frequency with which transitions were replayed is shown in the second row. Note that arrows represent states rather than transitions. All results are averaged over 50 independent experimental runs. The location where the agent started is marked blue. The goal location is marked green.

becomes clear when comparing both replay frequency and CI of different β for transitions that involve forward movements. For example, in the simple setup, for $\beta = 0$, the transitions involving the goal location were more frequently replayed than for $\beta = 1$ and the interference values measured for those transitions were much lower. Other transitions (e.g. in the lower right corner) were more frequently replayed for $\beta = 0$, too, yet interference scores were greater for those transitions than for $\beta = 1$ (fig. 4).

For some transitions, on the other side, a link between CI and replay frequency is clearly visible. For example, the already mentioned explicitly high CI_{sa} scores for transitions involving the goal states that were detected for all $\beta > 0$ (where these transitions were rarely replayed) but not for $\beta = 0$ (where forward transitions were replayed frequently). Or the fact that the walled states in the center of tunnel maze LV4, that were rarely replayed across all tested algorithms, exhibited among the greatest CI_{sa} values figs. A5 to A7. Nonetheless, the results show that CI for a transition depends not or not

only on the frequency with which it was replayed, but rather on the overall replay frequency distribution among transitions.

4 Summary and Future Work

Catastrophic interference (McCloskey & Cohen, 1989) has been known as a problem in training ANNs for a long time and it is recently gaining more attention in the domain of deep RL where multiple proposals have been made to mitigate it (e.g.Kirkpatrick et al., 2017). This thesis paper explores the occurrence of catastrophic interference in the application of a prioritized replay algorithm, SFMA, in deep RL. In the experiments, DQNs with different parameter sets for SFMA were trained in an episodic learning paradigm. The results presented in chapter 3.1 indicate that the performance of these algorithm can be linked to the occurrence of interference. In the tested setups, with one exception, that is $\beta = 0$, parameter settings that were previously shown to produce sequences of experiences more reliably, were always the ones performing worse than more stochastic settings. The follow-up analysis in section 3.2 therefore attempted to link the characteristic replay pattern of more deterministic SFMA to network instability by examining the interference for single transitions. While a clear correlation between interference and the overall distribution of replay frequency among transitions became visible, to what extent information associated with a specific transition was subject to interference, could not directly be linked to the representation of this transition in replay. Although these results provide more clarity with respect to the second research question, this study has its natural limitations. Most noticeably, a bug has partially rendered the analysis results unusable and only a fraction of the data could be used. Therefore, the experiments that were described here should be revisited with a corrected version of the code in order to exploit fully the potential of the analysis.

Due to the learning dynamics of RL and SFMA's sampling method, the replay frequency distribution shifts over time. Therefore, a major limitation of the conducted analysis is that replay frequency was only viewed at from a static perspective. Even though, an attempt to analyze the temporal correlation of replayed experiences was started, due to the limited time, it did not archive any presentable results. In order to motivate future work, I will, nonetheless, briefly present an approach for this analysis.

The autocorrelation function measures the correlation of a time series $x_1, ..., x_n$ with a delayed version of itself over varying time lags (Brockwell &

Davis, 2016). The sample autocorrelation function is given by:

$$ACF(h) = \frac{\sum_t^{n-|h|}(x_t - \bar{x})(x_{t+|h|} - \bar{x})}{\sum_t^{n-|h|}(x_t - \bar{x})^2} \tag{8}$$

where $\bar{x}$ represents the sample mean and h is the time lag. Importantly, in contrast to the replay frequencies in each update phase, that can be seen as a vector of size equal to the number of possible transitions, x_i is typically a scalar value. In the remaining time, I did not manage to find an example of an application of the sample ACF in an analogous case and to the best of my knowledge, the sample ACF has never been used to analyze sequential replay algorithms. Therefore, it seems to me that the equation either has to be adapted to the multidimensional case or another method, such as averaging or dimension reduction techniques, must be used to apply the formulae. To explore these options further, was beyond the scope of this thesis.

In addition to that, the experimental results provided in section 3.2 pointed at a relationship between action space and relative performances among SFMA with the tested parameter settings. While this observation was not studied further in this paper, future work could examine whether the action space effects the sequentiality of replayed sequences as well, and how other action spaces affect relative performances.

Lastly, another interesting question for work to follow is how the occurrence of catastrophic interference in more deterministic SFMA can be avoided while not cutting short on the biological plausibility. Besides modifying replay samples directly (e.g. by enriching mini-batch samples by randomly sampled experiences), one could explore how the adjustment of other learning parameters, such as the type of optimizer that is used for training, might be adjusted to stabilize the training. Although this might seem implausible at first sight, there is evidence suggesting that applying the Adam optimizer instead of other modern optimization algorithms, can increase catastrophic interference (Ashley et al., 2021).

5 Acknowledgement

All experiments were computed on the Freie Universität Berlin hpc cluster Curta (Bennett et al., 2020).

References

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Ashley, D. R., Ghiassian, S., & Sutton, R. S. (2021). Does standard backpropagation forget less catastrophically than adam? *CoRR*, *abs/2102.07686*.

Bennett, L., Melchers, B., & Proppe, B. (2020). Curta: A general-purpose high-performance computer at ZEDAT, freie universität berlin.

Broadbent, N. J., Squire, L. R., & Clark, R. E. (2004). Spatial memory, recognition memory, and the hippocampus. *Proceedings of the National Academy of Sciences*, *101*(40), 14515–14520. https://doi.org/10.1073/pnas.0406344101

Brockwell, P. J., & Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer International Publishing. https://doi.org/10.1007/978-3-319-29854-2

Buhry, L., Azizi, A. H., & Cheng, S. (2011). Reactivation, Replay, and Preplay: How It Might All Fit Together. *Neural Plasticity*, *2011*(1), 203462. https://doi.org/10.1155/2011/203462

Deacon, R. M. J., Bannerman, D. M., Kirby, B. P., Croucher, A., & Rawlins, J. N. P. (2002). Effects of cytotoxic hippocampal lesions in mice on a cognitive test battery. *Behavioural Brain Research*, *133*(1), 57–68. https://doi.org/10.1016/S0166-4328(01)00451-X

Diekmann, N., & Cheng, S. (2023). A model of hippocampal replay driven by experience and environmental structure facilitates spatial learning (P. Piray, L. L. Colgin, & H. F. Ólafsdóttir, Eds.). *eLife*, *12*, e82301. https://doi.org/10.7554/eLife.82301

Diekmann, N., Vijayabaskaran, S., Zeng, X., Kappel, D., Menezes, M. C., & Cheng, S. (2023). CoBeL-RL: A neuroscience-oriented simulation framework for complex behavior and learning. *Frontiers in Neuroinformatics*, *17*. https://doi.org/10.3389/fninf.2023.1134405

Fedus, W., Ghosh, D., Martin, J. D., Bellemare, M. G., Bengio, Y., & Larochelle, H. (2020). On Catastrophic Interference in Atari 2600 Games. *ArXiv*.

Foster, D. J., & Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, *440*(7084), 680–683. https://doi.org/10.1038/nature04587

Gallistel, C. R., Fairhurst, S., & Balsam, P. (2004). The learning curve: Implications of a quantitative analysis. *Proceedings of the National Academy of Sciences of the United States of America*, *101*(36), 13124–13131. https://doi.org/10.1073/pnas.0404965101

Gupta, A. S., van der Meer, M. A. A., Touretzky, D. S., & Redish, A. D. (2010). Hippocampal Replay Is Not a Simple Function of Experience. *Neuron*, *65*(5), 695–705. https://doi.org/10.1016/j.neuron.2010.01.034

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Khetarpal, K., Riemer, M., Rish, I., & Precup, D. (2022). Towards Continual Reinforcement Learning: A Review and Perspectives. *Journal of Artificial Intelligence Research*, *75*, 1401–1476. https://doi.org/10.1613/jair.1.13673

Kingma, D. P., & Ba, J. (2017, January). Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

Kumaran, D., Hassabis, D., & McClelland, J. L. (2016). What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends in Cognitive Sciences*, *20*(7), 512–534. https://doi.org/10.1016/j.tics.2016.05.004

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Lee, A. K., & Wilson, M. A. (2002). Memory of Sequential Experience in the Hippocampus during Slow Wave Sleep. *Neuron*, *36*(6), 1183–1194. https://doi.org/10.1016/S0896-6273(02)01096-6

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*(3), 293–321. https://doi.org/10.1007/BF00992699

Liu, V., Kumaraswamy, R., Le, L., & White, M. (2019). The Utility of Sparse Representations for Control in Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 4384–4391. https://doi.org/10.1609/aaai.v33i01.33014384

Liu, V., Wang, H., Tao, R. Y., Javed, K., White, A., & White, M. (2023). Measuring and Mitigating Interference in Reinforcement Learning. https://doi.org/10.48550/ARXIV.2307.04887

Liu, V., White, A., Yao, H., & White, M. (2020). Towards a practical measure of interference for reinforcement learning. *ArXiv*.

Lo, Y. L., & Ghiassian, S. (2019, October). Overcoming Catastrophic Interference in Online Reinforcement Learning with Dynamic Self-Organizing Maps. https://doi.org/10.48550/arXiv.1910.13213

Louie, K., & Wilson, M. A. (2001). Temporally Structured Replay of Awake Hippocampal Ensemble Activity during Rapid Eye Movement Sleep. *Neuron*, *29*(1), 145–156. https://doi.org/10.1016/S0896-6273(01)00186-6

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, *102*(3), 419–457. https://doi.org/10.1037/0033-295X.102.3.419

McCloskey, M., & Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation* (pp. 109–165, Vol. 24). Elsevier. https://doi.org/10.1016/S0079-7421(08)60536-8

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013, December). Playing Atari with Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Morris, R. G. M., Garrud, P., Rawlins, J. N. P., & O'Keefe, J. (1982). Place navigation impaired in rats with hippocampal lesions. *Nature*, *297*(5868), 681–683. https://doi.org/10.1038/297681a0

O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, *34*(1), 171–175. https://doi.org/10.1016/0006-8993(71)90358-1

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Schaul, T., Borsa, D., Modayil, J., & Pascanu, R. (2019, April). Ray Interference: A Source of Plateaus in Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1904.11455

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

Watkins. (1989). Learning From Delayed Rewards.

Watkins & Dayan. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292. https://doi.org/10.1007/BF00992698

Zeng, X., Diekmann, N., Wiskott, L., & Cheng, S. (2023). Modeling the function of episodic memory in spatial learning. *Frontiers in Psychology*, *14*, 1160648. https://doi.org/10.3389/fpsyg.2023.1160648

A Appendix

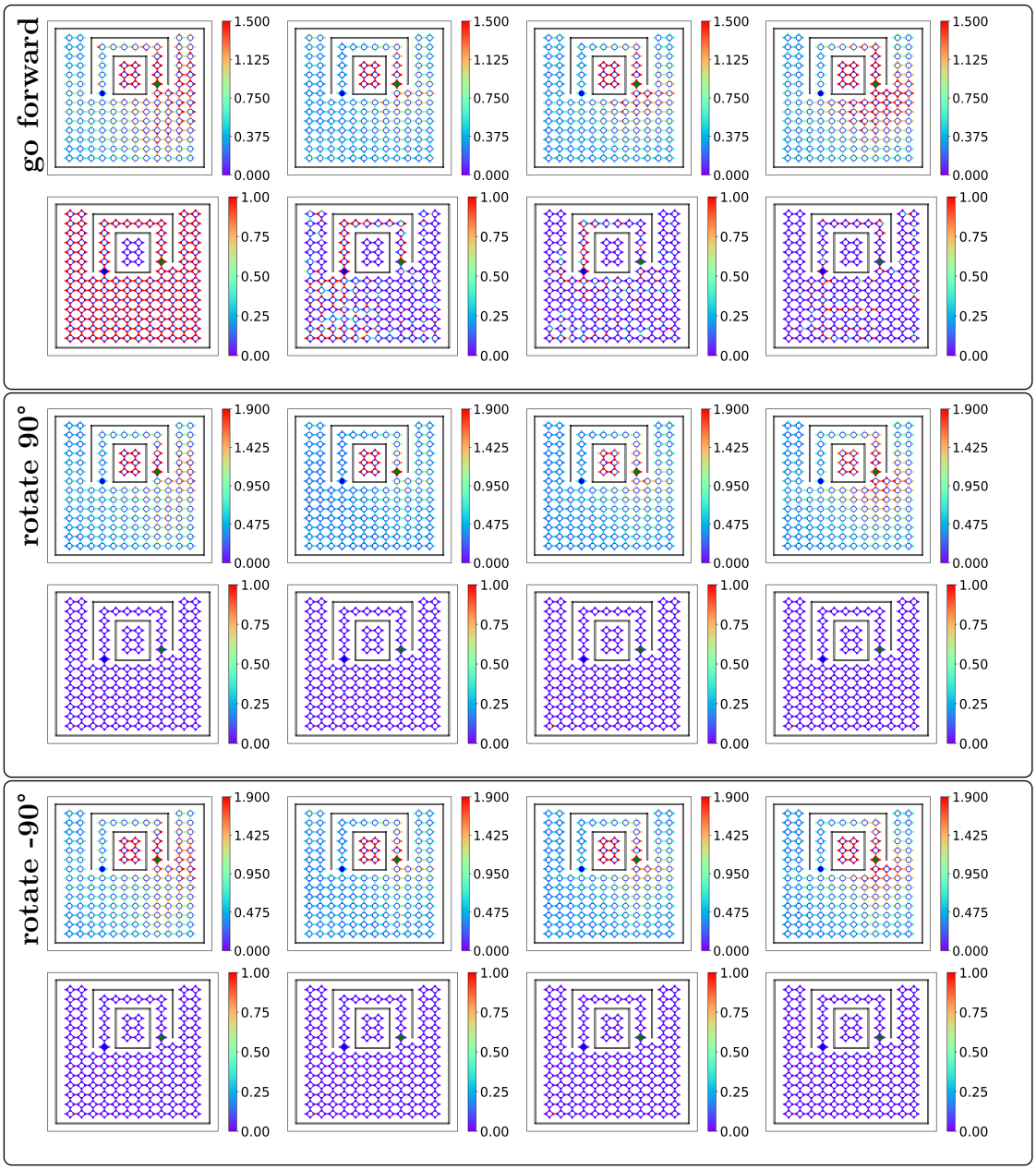

Figure A5: Experimental results for transitions in the complex setup. The columns represent results for $\beta = 0{,}1{,}5$ and 10 respectively. In each box, the top row shows Catastrophic Interference per transition. Beneath is the corresponding replay frequency. All results are averaged over 50 independent experimental runs. The location where the agent started is marked blue. The goal location is marked green.

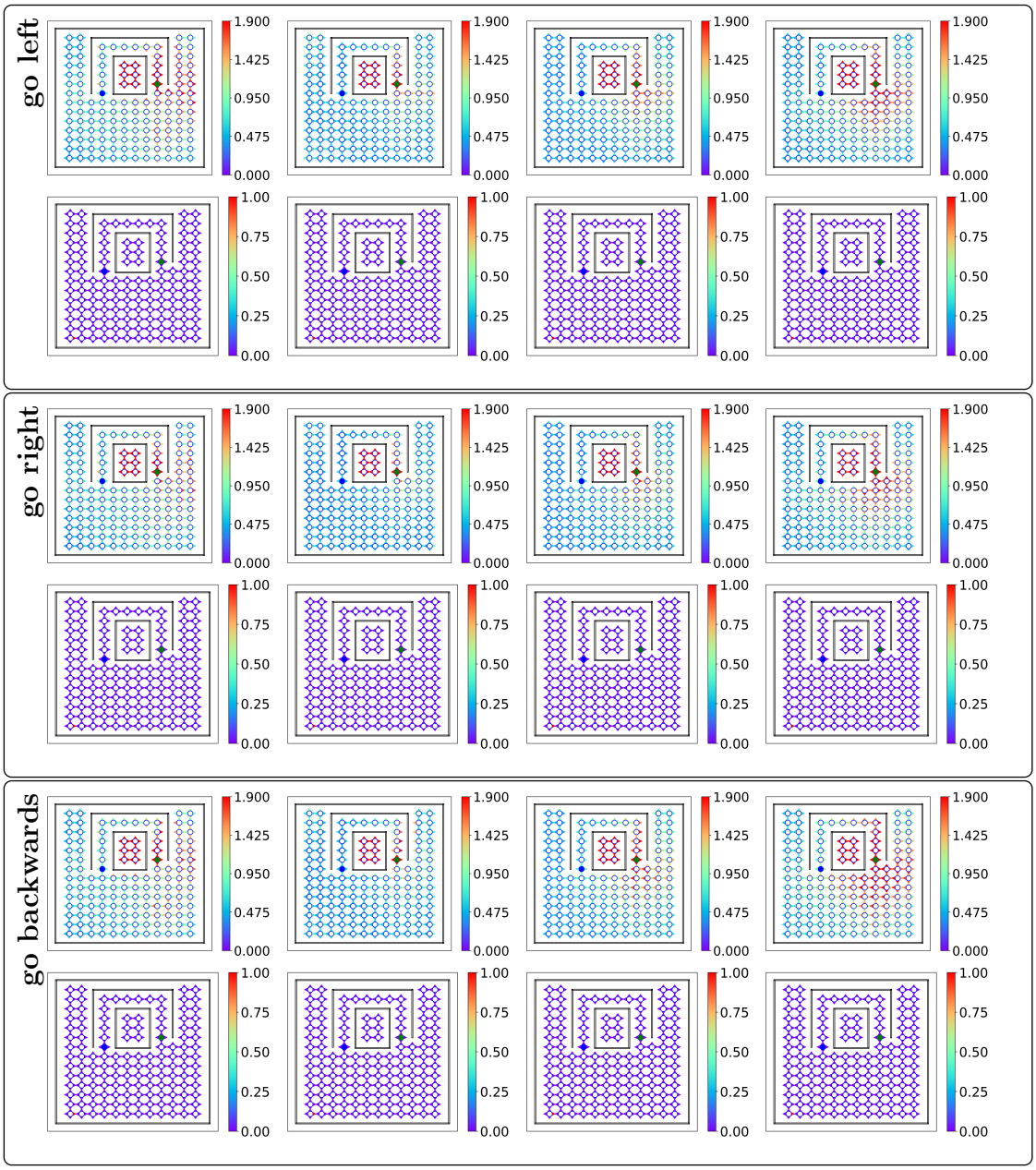

Figure A6: Experimental results for transitions in the complex setup. The columns represent results for $\beta = 0{,}1{,}5$ and 10 respectively. In each box, the top row shows Catastrophic Interference per transition. Beneath is the corresponding replay frequency. All results are averaged over 50 independent experimental runs. The location where the agent started is marked blue. The goal location is marked green.

Figure A7: Experimental results for transitions in the less complex setup. The columns represent results for $\beta = 0{,}1{,}5$ and 10 respectively. In each box, the top row shows Catastrophic Interference per transition. Beneath is the corresponding replay frequency. All results are averaged over 50 independent experimental runs. The location where the agent started is marked blue. The goal location is marked green.

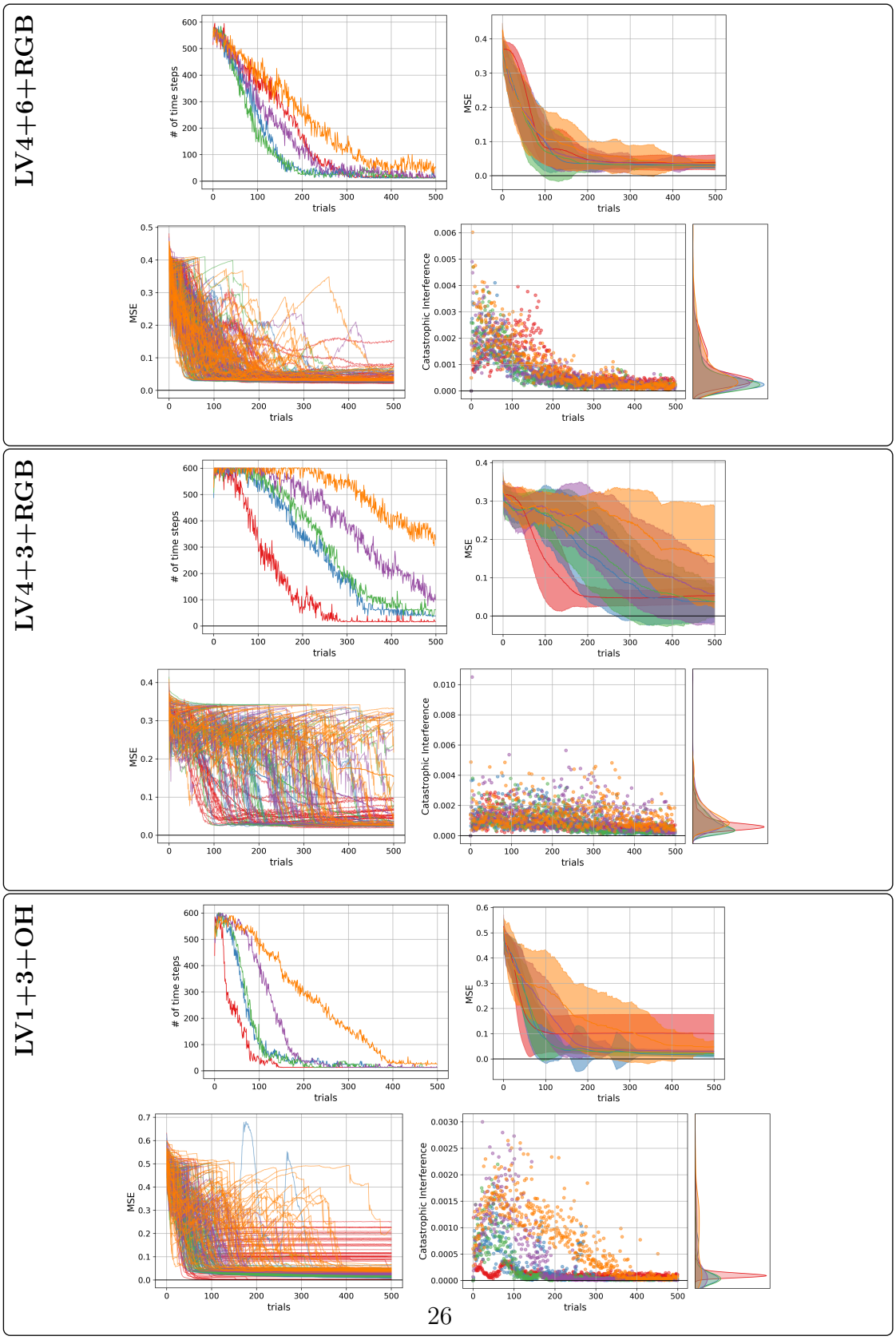

Figure A8: In each box, experimental results are shown for a different setup indicated by the descriptor. Each box reports learning curves of escape latencies, the average and std of MSE w.r.t. Q^*, MSE w.r.t. Q^* for all individual experiments and catastrophic interference per episode. All results are averaged over 50 independent experimental runs.

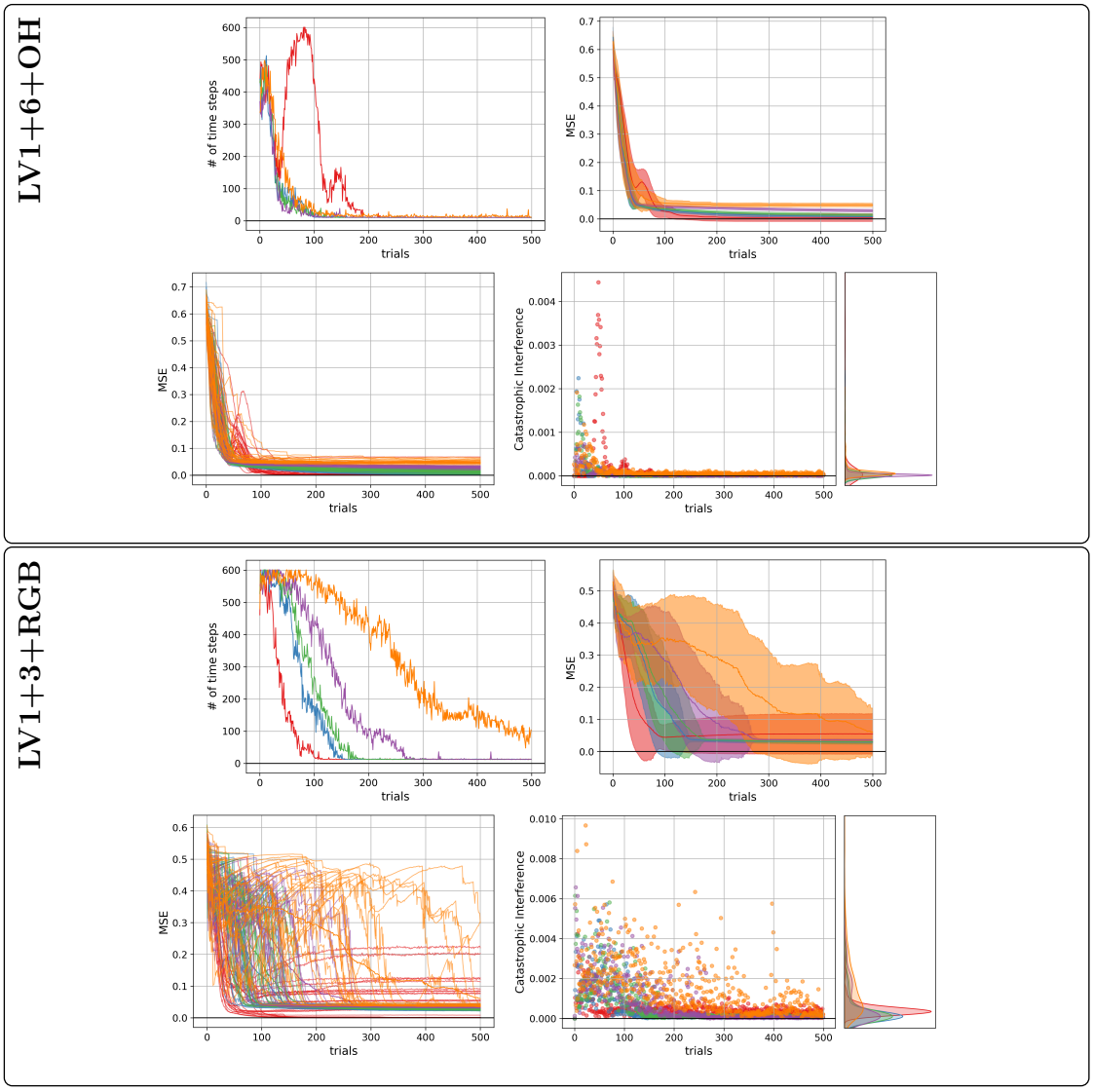

Figure A9: In each box, experimental results are shown for a different setup indicated by the descriptor. Each box reports learning curves of escape latencies, the average and std of MSE w.r.t. Q^*, MSE w.r.t. Q^* for all individual experiments and catastrophic interference per episode. All results are averaged over 25 independent experimental runs.

```
1     # Assumptions:
2     # - numpy has been imported as np
3     # - np.newaxis has been aliased as na for brevity
4     # - replayed_states and replayed_actions are numpy
          arrays of shape (500,1600)
5     # - num_states and num_actions are the number of states
          and actions respectively
6     # - actions is a list containing all actions ([0..5] or
          [0..2])
7     # - replayed_count is a numpy array of shape (500,
          num_states, num_actions)
8     #   that stores the replay frequency of all transitions
          in all episodes
9
10    expanded_states = np.arange(num_states)[na, na, :]
11
12    for action in actions:
13        # Bug 1: Incorrect masking
14        # Bug 2: Unintended masking in consecutive iterations
15        replayed_states *= (replayed_actions == action)
16
17        # Preparing for broadcasting
18        expanded_replayed_states = replayed_states[:, :, na]
19
20        # Broadcasting to count occurrences of each state
              within each episode
21        replayed_count[:, :, action] =
              np.sum(expanded_replayed_states ==
              expanded_states, axis=1)
```

Figure A10: Two bugs in the code written in Python 3 (Van Rossum & Drake, 2009), that caused the anomalies that were described in chapter 4. The shown code represents a snippet from a function that calculates the replay frequency for all transitions and episodes by applying the NumPy broadcasting method (Harris et al., 2020). Both bugs refer to line 15. Bug 1 is the incorrect masking of replayed states by multiplication with zero. This leads to unintentionally counting the state with index 0 in line 22. This bug explains why one state (the state with index 0) is always marked red in the replay frequency plots. Bug 2 is the consecutive masking in the loop starting in line 14, which causes the masking of all states in all loop iterations but the first. This bug explains why transitions involving the forward movement (the first iteration respectively) are the only ones with any variance in their replay frequency plots.