Continual Reinforcement Learning with Complex Synapses

Christos Kaplanis [1 2] **Murray Shanahan** [1 3] **Claudia Clopath** [2]

Abstract

Unlike humans, who are capable of *continual learning* over their lifetimes, artificial neural networks have long been known to suffer from a phenomenon known as *catastrophic forgetting*, whereby new learning can lead to abrupt erasure of previously acquired knowledge. Whereas in a neural network the parameters are typically modelled as scalar values, an individual synapse in the brain comprises a complex network of interacting biochemical components that evolve at different timescales. In this paper, we show that by equipping tabular and deep reinforcement learning agents with a synaptic model that incorporates this biological complexity (Benna & Fusi, 2016), catastrophic forgetting can be mitigated at *multiple* timescales. In particular, we find that as well as enabling continual learning across sequential training of two simple tasks, it can also be used to overcome *within*-task forgetting by reducing the need for an experience replay database.

1. Introduction

One of the outstanding enigmas in computational neuroscience is how the brain is capable of *continual* or *lifelong learning* (Wixted, 2004), acquiring new memories and skills very quickly while robustly preserving old ones. Synaptic plasticity, the ability of the connections between neurons to change their strength over time, is widely considered to be the physical basis of learning in the brain and knowledge is thought to be *distributed* across neuronal networks, with individual synapses participating in the storage of several memories. Given this overlapping nature of memory storage, it would seem that synapses need to be both labile in response to new experiences and stable enough to retain old memories - a paradox often referred to as the *stability-plasticity dilemma* (Carpenter & Grossberg, 1987).

Artificial neural networks also have a distributed memory but, unlike the brain, are prone to *catastrophic forgetting* (McCloskey & Cohen, 1989; French, 1999); when trained on a nonstationary data distribution, such as two distinct tasks in sequence, a network can quickly forget what it learnt from earlier data. In reinforcement learning (RL), where data is typically accumulated online as the agent interacts with the environment, the distribution of experiences is often nonstationary over the training of a *single* task, as well as across tasks, since (i) experiences are correlated in time and (ii) the agent's policy changes as it learns. A typical way of addressing nonstationarity of data in deep RL is to store experiences in a replay database and use it to *interleave* old data and new data during training (Mnih et al., 2015). However, this solution does not scale well computationally as the number of tasks grows and the old data might also become unavailable at some point. Furthermore, it does not explain how the brain achieves continual learning, since the question remains as to how an ever-growing dataset is then stored without catastrophic forgetting.

One potential answer may arise from the experimental observations that synaptic plasticity occurs at a range of different timescales, including short-term plasticity (Zucker & Regehr, 2002), long-term plasticity (Bliss & Lømo, 1973) and synaptic consolidation (Clopath et al., 2008). Intuitively, the slow components to plasticity could ensure that a synapse retains memory of a long history of its modifications, while the fast components render the synapse highly adaptable to the formation of new memories, perhaps providing a solution the stability-plasticity dilemma.

In this paper, we explore whether a biologically plausible synaptic model (Benna & Fusi, 2016), which abstractly models plasticity over a range of timescales, can be applied to mitigate catastrophic forgetting in a reinforcement learning context. Our work is intended as a proof of principle for how the incorporation of biological complexity to an agent's parameters can be useful in tackling the lifelong learning problem. By running experiments with both tabular and deep RL agents, we find that the model helps continual learning across two simple tasks as well as within a single task, by allaying the necessity of an experience

[1] Department of Computing, Imperial College London [2] Department of Bioengineering, Imperial College London [3] Google DeepMind, London. Correspondence to: Christos Kaplanis <christos.kaplanis14@imperial.ac.uk>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

replay database, indicating that the incorporation of different timescales of *plasticity* can correspondingly result in improved *behavioural* memory over distinct timescales. Furthermore, this is achieved even though the process of synaptic consolidation has no prior knowledge of the timing of changes in the data distribution.

2. Background

2.1. The Benna-Fusi Model

In this paper, we make use of a synaptic model that was originally derived to maximise the expected signal to noise ratio (SNR) of memories over time in a population of synapses undergoing continual plasticity in the form of random, uncorrelated modifications (Benna & Fusi, 2016). The model assumes that a synaptic weight w at time t is determined by its history of modifications up until that time $\Delta w(t')$, which are filtered by some kernel $r(t-t')$, such that

$$w(t) = \sum_{t'<t} \Delta w(t') r(t-t'). \tag{1}$$

While constraining the variance of the synaptic weights to be finite, the expected (doubly logarithmic) area under the SNR vs. time curve of a given memory is typically maximised when $r(t) \sim t^{-\frac{1}{2}}$, i.e. the kernel decays with a power law.

Implementing this model directly is impractical and unrealistic, since it would require recording the time and size of every synaptic modification; however, the authors show that the power law decay can be closely approximated by a synaptic model consisting of a finite chain of N communicating dynamic variables (as depicted in Figure 1). The dynamics of each variable u_k in the chain are determined by interaction with its neighbours in the chain:

$$C_k \frac{du_k}{dt} = g_{k-1,k}(u_{k-1} - u_k) + g_{k,k+1}(u_{k+1} - u_k) \tag{2}$$

except for $k = 1$, for which we have

$$C_1 \frac{du_1}{dt} = \frac{dw_{ext}}{dt} + g_{1,2}(u_2 - u_1) \tag{3}$$

where $\frac{dw_{ext}}{dt}$ corresponds to a continuous form of the $\Delta w(t')$ updates (Equation 1). For $k = N$, there is a leak term, which is constructed by setting u_{N+1} to 0. The synaptic weight itself w is just read off from the value of u_1, while the other variables are hidden and have the effect of regularising the value of the weight by the history of its modifications.

From a mechanical perspective, one can draw a comparison between the dynamics of the chain of variables and liquid flowing through a series of beakers with different base areas C_k connected by tubes of widths $g_{k-1,k}$ and $g_{k,k+1}$, where the value of a u_k variable corresponds to the level of liquid in the beaker (Figure 1).

Given a finite number of beakers per synapse, the best approximation to a power law decay is achieved by exponentially increasing the base areas of the beakers and exponentially decreasing the tube widths as you move down the chain, such that $C_k = 2^{k-1}$ and $g_{k,k+1} \propto 2^{-k-2}$. Beakers with wide bases and connected by smaller tubes will necessarily evolve at longer timescales. From a biological perspective, the dynamic variables can be likened to reversible biochemical processes that are related to plasticity and occur at a large range of timescales.

Importantly, the model abstracts away from the causes of the synaptic modifications Δw and so is amenable for testing in different learning settings. In the original paper (Benna & Fusi, 2016), the model was shown to extend lifetimes of random, uncorrelated memories in a perceptron and a Hopfield network, while in this work we test the capacity of the model to mitigate behavioural forgetting in more realistic tasks where synaptic updates are unlikely to be uncorrelated.

In all our experiments, we simulated the Benna-Fusi ODEs using the Euler method for numerical integration.

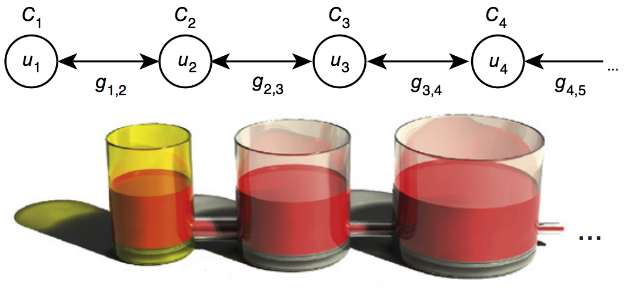

Figure 1. Diagrams adapted from (Benna & Fusi, 2016) depicting the chain model (top) and the analogy to liquid flowing between a series of beakers of increasing size and decreasing tube widths (bottom).

2.2. Reinforcement Learning

All experiments in this paper were conducted in an RL paradigm. The RL setting is formalised as a Markov Decision Process (MDP) defined by a tuple $\langle \mathcal{S}, \mathcal{A}, p_s, r \rangle$, whereby at time step t, an agent observes the state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$, resulting in a reward $r(s_t, a_t)$ and transition to the next state s_{t+1} with probability $p_s(s_{t+1}|s_t, a_t)$. The goal of the agent is to find a policy, defined by a probability distribution over actions given the state $\pi(a_t|s_t)$, that maximises its expected sum of discounted future rewards:

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{\pi}[r(s_t, a_t)] \tag{4}$$

where $\mathbb{E}_{\pi}$ is the expectation under the reward distribution defined by policy π.

2.2.1. Q-LEARNING

Q-learning (Watkins & Dayan, 1992) is a well-known reinforcement learning algorithm that involves learning the Q-values for each state-action pair which, given a policy π, are defined as:

$$Q^\pi(s,a) = \mathbb{E}_\pi\big[\sum_{i=t}^{\infty}\gamma^{i-t}r_i|s_t=s,a_t=a\big] \tag{5}$$

where γ is a temporal discount factor. By sampling experiences in the form (s_t, a_t, r_t, s_{t+1}) from a sufficiently exploratory policy and using them to update the Q-values as follows:

$$\delta_t \leftarrow r_t + \gamma V(s_{t+1}) - Q(s_t,a_t) \tag{6}$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \eta\delta_t, \tag{7}$$

where η is the learning rate and $V(s_{t+1}) = \max_a Q(s_{t+1},a)$, Q will eventually converge to Q^*, the value function of the optimal policy π^*, which is derived as

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a'} Q^*(s,a'), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

A common policy used for training, and the one used in this paper, is ϵ-greedy, whereby with probability $1-\epsilon$ the agent chooses the action with the highest Q-value and, with probability ϵ, chooses an action uniformly at random.

In this paper, we use a variant of Q-learning called 'naive' $Q(\lambda)$ (Sutton & Barto, 1998), which can speed up the convergence of Q-learning and involves maintaining an eligibility trace $e(s,a)$ for each state-action pair. At each time step, all eligibility traces are updated as follows:

$$e_t(s,a) = \begin{cases} 1, & \text{if } s_t = s \text{ and } a_t = a, \\ \gamma\lambda e_{t-1}(s,a), & \text{otherwise.} \end{cases} \tag{9}$$

where $\lambda \in [0,1]$ is a constant decay parameter. All Q-values are then updated by:

$$Q(s,a) \leftarrow Q(s,a) + \eta\delta e(s,a) \tag{10}$$

As is elucidated further on in the Experiments section, the eligibility traces can also be used to modulate the rate of synaptic consolidation to improve memory retention.

2.2.2. DEEP Q NETWORKS

In high-dimensional, continuous state spaces, it is infeasible to maintain a table of Q-values for all state-action pairs; in order to learn a good policy, an agent must be able to use its experience to generalise to previously unseen situations. Deep Q Networks (DQN) (Mnih et al., 2015) are artificial neural networks that are trained to approximate a mapping from states to Q-values by optimising the following cost function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma V(s';\theta^-) - Q(s,a;\theta)\right)^2\right] \tag{11}$$

where θ are the parameters of the network, $V(s';\theta^-) = max_{a'}Q(s,a';\theta^-)$ and θ^- are the parameters of an older version of the network (referred to as the target network) used to counteract instability in training due to quickly changing target values. D is the *experience replay database*, which records the agent's experiences in a FIFO queue and is sampled from at random during training. Consecutive experiences are usually highly correlated with one another and thus training in an online fashion can cause the network to *overfit* to recent data; by jumbling together old and new data, the database thus plays an essential role in *decorrelating* updates to the network and preventing catastrophic forgetting of older experiences. In some of our experiments described later on, we show that equipping the parameters of the network with the Benna-Fusi model can attenuate the need for an experience replay database due to better retention of older memories.

In our experiments, we trained the agents using a *soft Q-learning* objective (Haarnoja et al., 2017), which generalises Q-learning by simultaneously maximising the entropy of the agent's policy:

$$\pi^* = \arg\max_\pi \sum_t \mathbb{E}_\pi[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))] \tag{12}$$

where α is a constant that controls the balance between reward and entropy maximisation.

One benefit of soft Q-learning is that it can generate a more robust policy as it encourages the agent to learn multiple solutions to the task and, in our experiments with DQN, we found that it helped to stabilise performance over time.

3. Experiments

The overarching goal of the experiments was to test whether applying the Benna-Fusi model to an agent's parameters could enhance its ability to learn continually in an RL setting. Our aim was to demonstrate the *potential* for the model in enabling continual learning and, for this reason, we tested it in relatively simple settings, where catastrophic forgetting is nevertheless still an issue.

The first experiments, which apply the model in a simple tabular Q-learning agent, were intended to serve as a proof of principle and as a means to gaining an intuition of the mechanics of the model through visualisation. Subsequently, we tested it in a deep RL agent to evaluate its effect on the agent's ability to learn continually across two simple tasks and also within a single task. The only Benna-Fusi parameters that were varied across experiments were the first tube width ($g_{1,2}$) and the number of hidden variables, which jointly determine the range of timescales that the model can capture.

3.1. Continual Q-learning

The first set of experiments were conducted in order to test whether applying the Benna-Fusi model to tabular Q-values could be used to facilitate continual reinforcement learning in a simple grid-world setting.

3.1.1. Experimental Setup

The environment consisted of 100 states organised into a 10x10 two-dimensional grid and the agent was equipped with 5 actions, 4 of which deterministically move the agent to a vertically or horizontally adjacent state and the last of which is a pick-up action that must be chosen to collect the reward when in the correct location. The agent was trained alternately on two different tasks; in the first, the reward was located in the upper right-hand corner of the grid and, in the second, it was in the bottom left-hand corner. An episode was terminated if the agent reached the goal state and successfully picked up the reward, or if it took a maximum number of steps without reaching the goal. In order to test the agent's ability to learn continually, the goal location was switched every 10,000 episodes (one epoch) and the time taken for the agent to *relearn* to capture the reward was measured.

Three different agents were trained and compared:

- A control agent trained in an online fashion with naive $Q(\lambda)$ using an ϵ-greedy policy.
- A Benna-Fusi agent, also trained with naive $Q(\lambda)$, but for which the tabular Q-values were modelled as a Benna-Fusi synapses, each with their own chain of interacting dynamic variables. For a given state-action pair (s, a), we denote the first variable in the chain as $Q^1(s, a)$, which corresponds to u_1 in Equation 3 and is the 'visible' Q-value that determines the agent's policy at any time. The Q-learning updates $\eta\delta(t)e_t(s, a)$ correspond to the $\Delta w(t)$ modifications. The deeper variables in the chain $Q^k(s, a)$, with $k > 1$, can be thought of as 'hidden' Q-values that 'remember' what the visible Q-value function was over longer timescales and regularise it by its history.
- A modified Benna-Fusi agent, whereby at every time step, the flow from $Q^k(s, a)$ to $Q^{k+1}(s, a)$ for all variables in the chain was scaled by a multiple of the eligibility trace $e_t(s, a)$. The flow from shallow variables to deeper variables in the chain can be thought of as a process of *consolidation* of the synapse, or in this case Q-value. The rationale for modulating this flow by the eligibility trace is that it only makes sense to consolidate parameters that are actually being used and modified; for example, if a state s has not been visited for a long time, we should not become increasingly sure of any of the Q-values $Q^1(s, a)$.

In a Benna-Fusi chain of length N, $\frac{C_1}{g_{1,2}}$ and $\frac{C_N}{g_{N,N+1}}$ determine the shortest and longest memory timescales of the hidden variables respectively. In our experiments, we set $g_{1,2}$ to 10^{-5} to correspond roughly to the minimum number of Q-learning updates per epoch, and the number of variables in each chain to 3, all of which were initialised to 0. The ODEs were numerically integrated after every Q-learning update with a time step of $\Delta t = 1$. A table of all parameters used for simulation is shown in Supp. Table 1.

3.1.2. Results

The Benna-Fusi agents learned to switch between good policies for each task significantly faster than the control agent, with the modified Benna-Fusi agent being the quickest to relocate the reward for the first time at the beginning of each epoch (Figure 2). After the agents have learned to perform the task in the first epoch, it takes them all a long time to find the reward when its location is switched to the opposite corner at the beginning of the second epoch, since their policies are initially tuned to move actively away from the reward. After subsequent reward switches, however, while the control agent continues to take a long time to relearn a good policy due to the negative transfer between the two tasks, both Benna-Fusi agents learn to re-attain a good level of performance on the task much faster than after the initial task-switch (see bottom of Figure 2).

In order to visualise the role of the hidden variables of the Benna-Fusi model in enabling continual learning, we define $V^k(s) := \max_{a'} Q^k(s, a')$; for $k = 1$, this simply corresponds to the traditional value function $V(s)$ and, for $k > 1$, one can interpret $V^k(s)$ as a 'hidden' value function that records the value function over longer timescales. Figure 3 shows a snapshot of the V^k values during training, depicting how the deeper Benna-Fusi variables remember the Q-values at a longer timescale, enabling the agent to quickly recall the location of the previous reward. See Supp. Video for an animation of this process.

3.2. Continual Multi-task Deep RL

The next set of experiments were to test if we could observe similar benefits to continual learning if the Benna-Fusi model were applied to the parameters of a deep RL agent alternately performing two simple tasks. While having a better memory retention for tabular Q-values has a direct impact on an agent's ability to recall a previous policy, it

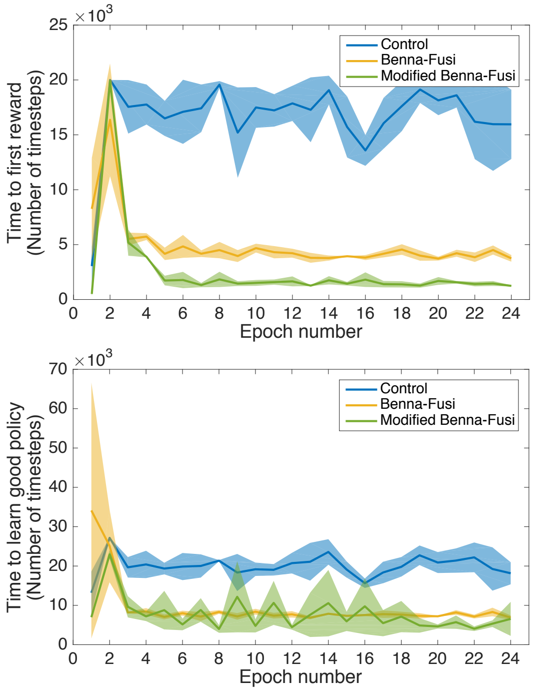

Figure 2. (Top) How long it took each agent to relearn to navigate to the first reward at the beginning of each epoch. (Bottom) How many time steps it took for the 20-episode moving average of episode lengths to drop below 13, as a measure of how long it took to (re)learn a good policy. Mean over 3 runs with 1 s.d. error bars.

is less obvious that longer memory lifetimes in individual synapses should yield better behavioural memory in a distributed system such as a deep Q-network.

3.2.1. Experimental Setup

The two tasks used for this experiment were Cart-Pole[1] and Catcher, which were suitable for training on the same network as they have the same size of state and action spaces respectively.

Similarly to the tabular Q-learning experiments, an agent was trained alternately on the two tasks (for 40 epochs of 20,000 episodes) and, as a measure of its ability to learn continually, the time taken for the agent to (re)learn the task after every switch was recorded. A task was deemed to have been (re)learnt if a moving average of the reward per episode moved above a predetermined level (450 for Cart-Pole, which has max reward 500, and 10 for Catcher, which has max reward about 14).

Experiments were run with two types of agent, a control agent and a Benna-Fusi agent. In order to ensure that the difference in performance of the two agents was not just due to differences in the effective learning rate (which is likely to be lower in the Benna-Fusi agent as the parameters are regularised by the hidden variables), the control agent was run with several different learning rates. The Benna-Fusi agent was only run with $\eta = 0.001$.

The control agent was essentially a DQN (Mnih et al., 2015) with two fully connected hidden layers of 400 and 200 ReLUs respectively, but with a number of modifications that were made in order to give it as good a chance as possible to learn continually. The network was trained with the soft Q-learning objective (Haarnoja et al., 2017), which helped to stabilise learning in each task, presumably by maintaining a more diverse set of experiences in the replay database[2]. Furthermore, as in (Kirkpatrick et al., 2017), while the network weights were shared between tasks, each layer of the network was allowed to utilise task-specific gains and biases, such that computations at each layer were of the form:

$$y_i = g_i^c \left(b_i^c + \sum_j W_{ij} x_j \right) \tag{13}$$

where c indexes the task being trained on. This helped overcome the issue of training a network on two different Q-functions, which has been reported to be very challenging even as a regression task (Rusu et al., 2015).

The experience replay database had a size of 2000, from which 64 experiences were sampled for training with Adam (Kingma & Ba, 2014) at the end of every episode. Crucially, the database was cleared at the end of every epoch in order to ensure that the agent was only training on one task at a time. The agent was ϵ-greedy with respect to the stochastic soft Q-learning policy and ϵ was decayed from 1 to almost 0 over the course of each epoch. Finally, 'soft' target network updates were used as in (Lillicrap et al., 2015), rather than hard periodic updates used in the original DQN. A full table of parameters used can be seen in Supp. Table 2.

The Benna-Fusi agent was identical to the control agent, except that each network parameter was modelled as a Benna-Fusi synapse with 30 variables with $g_{1,2}$ set to 0.001625, ensuring that the longest timescale ($\propto \frac{C_{30}}{g_{30,31}}$) comfortably exceeded the total number of updates over training ($\approx 2^{25}$). In order to speed up computation, rather than simulate 64 time steps of the ODEs after every replay batch, these were approximated by conducting one Euler update with $\Delta t = 64$. For this reason, the effective flow between u_1 and u_2 was $64 * 0.001625 = 0.1$; if it were larger than 1 this would lead to instability or unwanted oscillations or negative u-values, so we could not increase $g_{1,2}$ much more. The complexity of the algorithm is $\mathcal{O}(mN)$, where N is the number of

[1] The version used was CartPole-v1 from the OpenAI Gym (Brockman et al., 2016)

[2] In particular, we found this more effective than having a larger replay database, decaying ϵ to a positive value or just having a softmax policy.

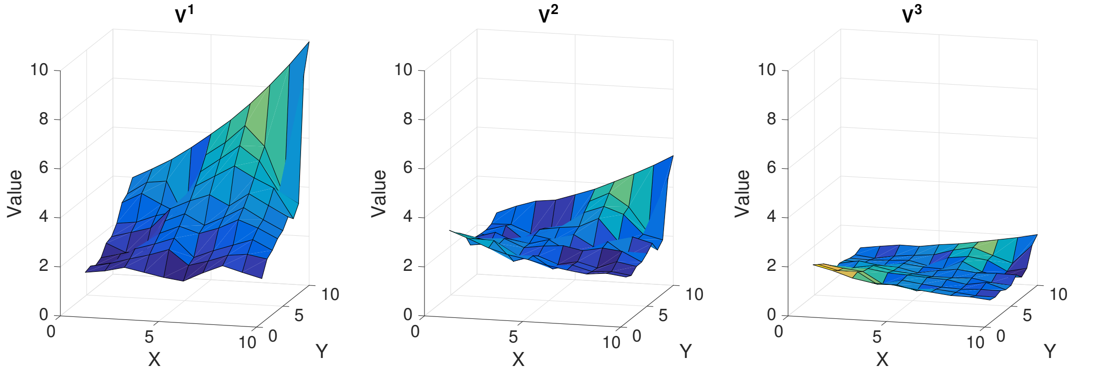

Figure 3. Surface plots of a snapshot of the visible (V^1) and hidden (V^2 and V^3) values of each state during training. While V^1 only appears to retain information about the current reward at (10,10), V^2 and V^3 still remember that there is value at (0,0). When the reward location is switched back to (0,0), flow from the deeper variables in the chain back into V^1 make it easier for the agent to recall the previous reward location. See Supplementary video for animation of values over training.

trainable parameters in the network and m is the number of Benna-Fusi variables per parameter. The compression of 64 Benna-Fusi updates into one resulted in the overall runtime being only 1.5-2 times longer than the control model.

The initial values of the hidden variables were normally distributed with variances decaying linearly with the depth in the chain, approximately matching the equilibrium distribution shown for random, uncorrelated memories in the original paper (Benna & Fusi, 2016). Furthermore, we incrementally allowed flow to occur from the deeper variables to the shallow ones so that the parameters were not constrained much by the random initialisation and only by hidden variables that have had enough time to adapt to the actual experiences of the agent. Specifically, flow from u_{k+1} to u_k was only enabled after $\frac{2^k}{g_{1,2}}$ gradient updates.

3.2.2. Results

Over the course of training the Benna-Fusi agent became faster at reaching at adequate level of performance on each task than the control agents (Figure 4, top), thus demonstrating a better ability for continual learning. Interestingly, while the control agents were all able to learn Cart-Pole at the beginning of training, subsequent training on Catcher then left the network at a starting point that made it very hard or impossible for the agents to relearn Cart-Pole (as evidenced by the number of epochs where an adequate performance was never reached), exhibiting a severe case of catastrophic forgetting. The Benna-Fusi agent did not display this behaviour and, instead, relearned the task quickly in all epochs. It is important to note that parameters were chosen such that the control agents were all capable of learning a very good policy for either task when trained from scratch. In Catcher, the Benna-Fusi agent sometimes took longer to converge to a good performance in the first few epochs of training, but subsequently became faster than all the control agents in recalling how to perform the task.

We tested the agents' ability to remember over multiple timescales by running the same experiments with different epoch lengths (2500 to 160k) and found that the Benna-Fusi agent demonstrated a better memory than the control in all cases (Supp. Figure 1). Furthermore, to ensure that these benefits are not limited to a two-task setting, we ran experiments rotating over three tasks and obtained similar results (Supp Figure 2).

3.3. Continual Learning within a Single Task

The continual learning problem is normally posed as the challenge of learning how to perform a series of well-defined tasks in sequence; in RL, however, the issue of nonstationary data often occurs within the training of *one* task. This effect occurs primarily due to (i) strong correlation in time between consecutive states and (ii) changes in the agent's policy altering the distribution of experiences. The most common way to deal with this problem is to use an experience replay database to decorrelate the data, without which the agent can struggle to learn a stable policy (Supp. Figure 3). In the final set of experiments, we wanted to see whether using the Benna-Fusi model could enable stable learning in a single task *without* the use of a replay database.

3.3.1. Experimental Setup

Control and Benna-Fusi agents were trained on Cart-Pole and Catcher separately in an *online* setting, such that there was no experience replay database and the agents were trained after every time step on the most recent experience.

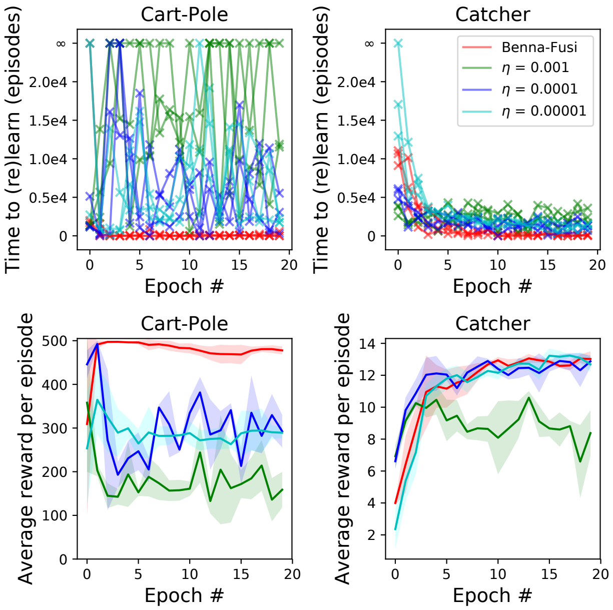

Figure 4. (Top) How long it took for the agents to relearn each task from the beginning of each epoch; the # of training episodes needed for the 10 test-episode moving average of reward to surpass the threshold is plotted for 3 runs per agent. Runs that did not relearn within the are marked at ∞. (Bottom) Reward per episode averaged over each epoch for each task; means with s.d. error bars over 3 runs.

The architectures of the control and Benna-Fusi agents were the same as in the previous set of experiments bar a couple of differences: the network was smaller (two hidden layers of 100 and 50 units respectively) and, in the Benna-Fusi agent, $g_{1,2}$ was set to a larger value of 0.01 in order to be able to remember experiences over shorter timescales.

3.3.2. Results

While none of the control agents were able to learn and maintain a consistently good policy for the Cart-Pole task, the Benna-Fusi agent learned to perform the task to perfection in most cases (Figure 5). For Catcher, however, all agents were able to learn a consistently good policy, with the control agent learning a bit faster (see Supp. Figure 4).

The reason that the control agents struggle to learn a stable policy for Cart-Pole in an online setting, but not Catcher, could be that the distribution of the training data is more non-stationary and thus the agents are more prone to catastrophic forgetting as they learn. A common aspect among control tasks, such as Cart-Pole, is that a successful policy often involves restricting experiences to a small part of the state space (de Bruin et al., 2016). For example, in Cart-Pole the aim is to keep the pole upright, and so if an agent trains for a while on a good policy, it may begin to overwrite knowledge of Q-values in states where the pole is significantly tilted. Since the agent is constantly learning, it could at some point make an update that causes it to make a wrong action that causes the pole to tilt to an angle that it has not experienced in a while. At this point, the agent might not only perform poorly since it has forgotten the correct policy in this region of the state space, but its policy might be further destabilised by training on these 'new' experiences. Furthermore, at this stage the exploration rate might have decayed to a low level, making it harder to relearn.

One idea is not to let ϵ to decay to 0, but in practice we found that this does not solve the problem and can actually make learning less stable (Supp. Figure 5). This could be (i) because the agent still overfits to states experienced during a good policy and the extra exploration just serves to perturb it into the negative spiral described above faster than otherwise, or (ii) as noted in (de Bruin et al., 2016), in control tasks the policy often needs to be very fine-tuned in an unstable region of the state space; this requires high-frequency sampling of a good policy and so makes excessive exploration undesirable (de Bruin et al., 2015). In Cart-Pole, the Benna-Fusi agent succeeds in honing its performance with recent experiences of a good policy while simultaneously remaining robust to perturbations by maintaining a memory of what to do in suboptimal situations that it has not experienced for a while.

In Catcher, a good policy will still visit a large part of the state space and consecutive states are also less correlated in time since fruit falls from random locations at the top of the screen. This may explain why the control agent does not have a problem learning the task successfully.

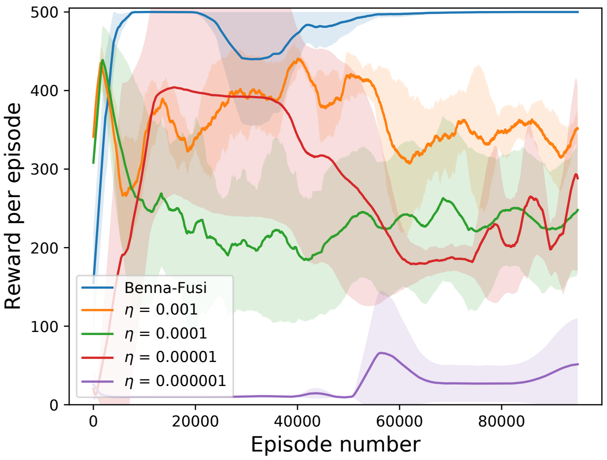

Figure 5. The 1000 test-episode moving average of reward in Cart-Pole for the Benna-Fusi agent and control agents with different learning rates; means and s.d. errorbars over 3 runs per agent.

4. Related Work

The concept of synaptic consolidation has been applied in a number of recent works that tackle the continual learning problem by adding quadratic terms to the cost function that selectively penalise moving parameters according to how important they are for the recall of previous tasks (Ruvolo & Eaton, 2013; Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2017). In (Kirkpatrick et al., 2017), the importance of each parameter for a task is proportional to its term in the diagonal of the Fisher Information matrix at the end of training. In (Zenke et al., 2017), the importance factor for a parameter is calculated in an online fashion by determining its contribution to the drop in the loss function over training; in contrast to (Kirkpatrick et al., 2017), which uses a local approximation of importance, this method provides a more global interpretation by considering a parameter's impact over the whole learning trajectory. In (Aljundi et al., 2017), importance is made proportional to the derivative of the L2-norm of the network output with respect to each parameter; by not relying on the loss function for consolidation, their method can be flexibly used to constrain the model with different data than those that were trained on.

The Benna-Fusi model also constrains parameters to be close to their previous values but, in contrast to the approaches described above, consolidation occurs (i) over a *range* of timescales, (ii) without any derived importance factors, and (iii) without any knowledge of task boundaries. These characteristics are useful for situations where you do not have prior knowledge of when and over what timescale the training data will change, a possibly realistic assumption for robots deployed to act and learn in the real world. Furthermore, the importance factors derived in the other works could feasibly be used to modulate the flow between the hidden variables as a way of combining approaches.

It must be noted that the idea of modelling plasticity at different timescales to mitigate catastrophic forgetting in a neural network is not new: in (Hinton & Plaut, 1987), each weight is split into separate 'fast' and 'slow' components, which allows the network to retrieve old memories quickly after training on new data. However, this model was only tested in a very simple setting, matching random binary inputs and outputs, and it is shown in (Benna & Fusi, 2016) that allowing the different components to *interact* with each other theoretically yields much longer memory lifetimes than keeping them separate. The momentum variables in Adam (Kingma & Ba, 2014) and the soft target updates in (Lillicrap et al., 2015; Polyak & Juditsky, 1992) also effectively remember the parameter values at longer timescales, but their memory declines exponentially, i.e. much faster than the power law decay in the Benna-Fusi model.

Other approaches to the continual learning problem that include the use of networks that grow to incrementally learn new skills (Ring, 1997; Rusu et al., 2016), implicitly training multiple models in one network (Goodfellow et al., 2013; Fernando et al., 2017) and building generative models to mimic old datasets that are no longer available (Shin et al., 2017), but these are all orthogonal to the approach used in this paper and could be combined with it.

5. Conclusion

In this paper, we took inspiration from a computational model of biological synapses (Benna & Fusi, 2016) to show that expressing each parameter of a tabular or deep RL agent as a dynamical system of interacting variables, rather than just a scalar value, can help to mitigate catastrophic forgetting over multiple timescales.

Our work is intended as a proof of concept that we envisage being extended in several ways. First, the sensitivity of continual learning performance to the parameters of the model, such as the number of hidden variables, should be analysed in order to optimise it and an investigation into the information content held at different depths of the chain could yield more effective readout schemes for the value of each weight. Furthermore, it will be important to test the model's capabilities in a more challenging setting by increasing the number and complexity of tasks, potentially using different architectures such as state-of-the-art actor-critic models (Lillicrap et al., 2015), as well as to see if the model can facilitate transfer learning in a series of related tasks. In some initial experiments with larger DQN on tasks from the Arcade Learning Environment (Bellemare et al., 2013; Brockman et al., 2016), we found that Benna-Fusi agents struggled to reach the same level of performance as the control agents - the reasons for this will need to be investigated in future work.

Finally, it would be interesting to adapt the model in light of the fact that synaptic consolidation is known to be regulated by neuromodulators such as dopamine, which, for example, has been associated with reward prediction error and exposure to novel stimuli (Clopath et al., 2008). One could modulate the flow between the hidden variables in the model by factors such as these, or by one of the importance factors cited in the previous section, in order to consolidate memory more selectively and efficiently.

Acknowledgements

We would like to thank Tom Schaul for his insightful comments and suggestions.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. *arXiv preprint arXiv:1711.09601*, 2017.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Benna, M. K. and Fusi, S. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19 (12):1697–1706, 2016.

Bliss, T. V. P. and Lømo, T. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of Physiology*, 232(2):331–356, 1973.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Carpenter, G. A. and Grossberg, S. Art 2: Self-organization of stable category recognition codes for analog input patterns. *Applied optics*, 26(23):4919–4930, 1987.

Clopath, C., Ziegler, L., Vasilaki, E., Büsing, L., and Gerstner, W. Tag-trigger-consolidation: a model of early and late long-term-potentiation and depression. *PLoS Computational Biology*, 4(12):e1000248, 2008.

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. The importance of experience replay database composition in deep reinforcement learning. In *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS)*, 2015.

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. Off-policy experience retention for deep actor-critic learning. In *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS)*, 2016.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Hinton, G. E. and Plaut, D. C. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pp. 177–186, 1987.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

McCloskey, M. and Cohen, J. N. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24: 109–165, 1989.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Ring, M. B. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Ruvolo, P. and Eaton, E. Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pp. 507–515, 2013.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

Wixted, J. T. The psychology and neuroscience of forgetting. *Annual Review of Psychology*, 55:235–269, 2004.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995, 2017.

Zucker, R. S. and Regehr, W. G. Short-term synaptic plasticity. *Annual Review of Physiology*, 64(1):355–405, 2002.