bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

Optimal Control of Neural Systems

Bodo Rueckauer and Marcel van Gerven

Department of Artificial Intelligence, Donders Institute for Brain, Cognition and Behaviour, Nijmegen, The Netherlands

Abstract

Brain-machine interfaces have reached an unprecedented capacity to measure and drive activity in the brain, allowing restoration of impaired sensory, cognitive or motor function. Classical control theory is pushed to its limit when aiming to design control laws that are suitable for large-scale, complex neural systems. This work proposes a scalable, data-driven, unified approach to study brain-machine-environment interaction using established tools from dynamical systems, optimal control theory, and deep learning. To unify the methodology, we define the environment, neural system, and prosthesis in terms of differential equations with learnable parameters, which effectively reduce to recurrent neural networks in the discrete-time case. Drawing on tools from optimal control, we describe three ways to train the system: Direct optimization of an objective function, oracle-based learning, and reinforcement learning. These approaches are adapted to different assumptions about knowledge of system equations, linearity, differentiability, and observability. We apply the proposed framework to train an in-silico neural system to perform tasks in a linear and a nonlinear environment, namely particle stabilization and pole balancing. After training, this model is perturbed to simulate impairment of sensor and motor function. We show how a prosthetic controller can be trained to restore the behavior of the neural system under increasing levels of perturbation. We expect that the proposed framework will enable rapid and flexible synthesis of control algorithms for neural prostheses that reduce the need for in-vivo testing. We further highlight implications for sparse placement of prosthetic sensor and actuator components.

1 Introduction

Closed-loop recording and stimulation in neurotechnology is becoming increasingly feasible [44] and calls for algorithmic advances in developing feedback controllers that make best use of improved recording and stimulation capabilities. Studies in human and animal models come with a number of practical and ethical hurdles that motivate the development of algorithms in simulation prior to deployment in a clinical setting. Neural systems, from single neurons to large-scale populations, are characterized by complex dynamics that can be challenging to model and control [43]. Classical control theory [3, 24, 9] provides powerful tools for designing control laws, but they are often limited to linear(-izable) or lower-dimensional systems and require knowledge of the underlying dynamics [47]. The present paper addresses this issue by using scalable, recurrent neural networks (RNNs) as building blocks which are trained via an optimal control oracle or reinforcement learning (RL) [55].

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

We propose a control-theoretic framework to study the stability and controllability of biologically motivated artificial neural systems embedded in simulated environments. From a high-level perspective, this framework models the brain-machine-environment interaction. We first consider the problem of modelling a neural system to perform a behavioral task in a virtual environment. In the language of control theory, the neural system forms a feedback controller in closed loop with the environment process. In a second step, we simulate a deterioration of the neural system (e.g. at the sensor or actuator) and add a secondary controller (a prosthesis) with the goal to restore behavioral function. In doing so we account for uncertainty in the model of the brain, nonlinearities, measurement noise, and limited availability of observable states and controllable neurons.

Our approach is to develop an in-silico control synthesis framework which builds on tools from optimal control theory and reinforcement learning. The latter is particularly suited because in general we cannot assume knowledge of the environment and brain dynamics, and may only have access to some observations from the environment, the actuator output from the neural system, plus perhaps a temporally sparse reward signal. Even operating under these constraints, RL in principle enables learning arbitrarily complex nonlinear models. In Q-learning [62], a strategy underlying many state-of-the-art RL algorithms, an agent learns a control policy that optimizes a Hamilton-Jacobi-Bellman equation online and without knowing the system dynamics. In optimal control theory, equations of this type are solved analytically (offline and assuming system knowledge) to derive optimal controllers. This common class of equations forms the link between RL and optimal control [31]. Examples for the use of RL in neural control include [41, 54] (in vivo), [65] (in vitro), and [36, 10] (in silico). These studies show great promise of RL to reduce the need for repeated calibration and instead adapt controllers autonomously to changes in the neural code or the sensor / actuator space.

A distinguishing feature of our approach is the unified treatment of environment, brain, and prosthesis as dynamical systems using (stochastic) differential equations. Most of previous work focused either on the brain-environment loop (e.g. to model neuronal dynamics) [48, 46], or on the brain-prosthesis loop (e.g. to train a neural decoder to control a prosthetic limb) [29, 28, 17]. Combining the two loops as proposed here enables end-to-end optimization of the brain model and controller, with adaptation to noise or deterioration appearing in any of the subsystems. The proposed framework provides a substrate for neural control engineering to exploit recent advances in brain-inspired RL [7], safe RL [15, 16], few-shot learning [60], and continual learning [61, 59, 38, 58].

2 Methods

2.1 Using dynamical systems to model the brain-machine-environment interaction

Before considering the full brain-machine-environment loop, we first concentrate on the unimpaired case and model how an agent (represented by its neural system) interacts with the environment to generate meaningful behavior. From a high-level perspective (Figure 1), the neural system transforms state observations $\mathbf{y}_\mathrm{E}$ from the environment into motor commands $\mathbf{u}_\mathrm{E}$. Both the neural system and the environment can be modeled by controlled stochastic differential equations (SDEs) of the form

$$d\mathbf{x} = \mu(\mathbf{x}, \mathbf{u})dt + \sigma(\mathbf{x}, \mathbf{u})d\mathbf{w}\,, \tag{1}$$

where the drift μ models the evolution of the states $\mathbf{x}$ under some external input $\mathbf{u}$ and the diffusion σ models how Brownian noise $\mathbf{w}$ enters the system. In the absence of noise,

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

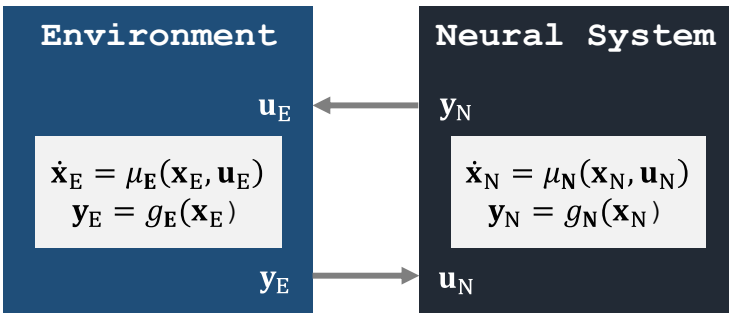


Figure 1: Classical agent-environment loop, where a neural system interacts with its environment by observing its states and applying feedback control. Here, the deterministic case is shown for simplicity; see Equation (1) for the stochastic form.

the model reduces to the ordinary differential equation $\dot{\mathbf{x}} = \mu(\mathbf{x}, \mathbf{u})$.

So far, we formulated the system dynamics as *continuous* differential equations. Various discretization methods can be applied for the purpose of simulating the system dynamics on digital hardware or solving for the states $\mathbf{x}$ numerically. A common approximation for SDEs is afforded by the Euler-Maruyama method. Starting from some initial value $\mathbf{x}_0$, the solution is recursively defined as

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t \cdot \mu(\mathbf{x}_t, \mathbf{u}_t) + \Delta \mathbf{w}_t \,. \tag{2}$$

Here, $\Delta t = T/N$ is the step size obtained by dividing the integration interval T into N steps. The noise term $\Delta \mathbf{w}_t$ is drawn from a normal distribution $\mathcal{N}(\mathbf{0}, \Delta t \cdot S)$ with $S = \Sigma_p \Sigma_p^T$. Here we assume a fixed diffusion matrix $\sigma(\mathbf{x}, \mathbf{u}) = \Sigma_p$ with Σ_p the process noise covariance.

Closed-loop control of a dynamical system requires observing its states. In real-world scenarios, one typically has access to just a subset of the states, which may moreover be perturbed by sensory noise. The resulting observations can be described as

$$\mathbf{y} = g(\mathbf{x}) + \boldsymbol{\epsilon}, \tag{3}$$

where $g(\mathbf{x})$ could be a matrix-vector product $C\mathbf{x}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \Sigma_o)$ with Σ_o the observation noise covariance.

2.2 Example environments

To illustrate the general applicability of the concepts developed in this paper, we introduce one linear and one nonlinear example of dynamical systems, which will be used to represent an environment for the neural system to act in.

2.2.1 Double integrator for particle control

A canonical example from control theory is the double integrator, which models the dynamics of a particle in a one-dimensional space under the influence of an external force $\mathbf{u}$. The environment's states are given by the particle's position q and velocity $\dot{q}$. Their dynamics are determined by the differential equation $\ddot{q}(t) = u(t)$, which identifies the particle acceleration with the control force. For the stochastic double integrator, the combined state $\mathbf{x} = (q, \dot{q})$ evolves according to

$$d\mathbf{x} = \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u} \right) dt + \Sigma_p d\mathbf{w} \,. \tag{4}$$

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

Given a moving particle away from the origin ($\mathbf{x}(0) \neq 0$), the task of the neural system consists in determining the sequence of motor commands $\mathbf{u}(t)$ that force the particle to the origin and stabilize it there ($\mathbf{x}(0) = 0$). The task can be made more challenging by increasing the process noise covariance and by allowing the neural system to observe only a noisy version of the position but not the velocity:

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x} + \boldsymbol{\epsilon} \,. \tag{5}$$

We will discuss ways to solve this task in Section 2.4 below.

2.2.2 Balancing an inverted pendulum

Another classic benchmark for control algorithms is the inverted pendulum, or cartpole balancing problem [4]. It consists of a cart with a pole attached that has its center of mass above the pivot point. The task is to move the cart horizontally so as to keep the pole in its unstable vertical position. When limiting the pendulum to one degree of freedom, the system has four states: cart position x, velocity $\dot{x}$, pole angle θ, and angular velocity $\dot{\theta}$. The system is nonlinear: In absence of control forces, the angle evolves as $\ddot{\theta} = \frac{g}{l} \sin\theta$ with standard gravity g and pole length l. Thus,

$$d \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \left(\begin{bmatrix} \dot{x} \\ 0 \\ \dot{\theta} \\ \frac{g}{l} \sin\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \mathbf{u} \right) dt + \Sigma_p d\mathbf{w} \,. \tag{6}$$

2.3 Analytic solution of linear, known systems

For a linear system with known dynamics, an optimal direct feedback controller can be derived analytically. The central idea is that the controller should minimize the squared deviation of states from their desired values, while expending the least amount of energy to do so. In the case of static target states, the problem is called Linear Quadratic Regulator (LQR), with the quadratic cost function

$$J = \int (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt \,. \tag{7}$$

The matrices Q, R weigh the contribution of individual states and controls to the overall cost. In general, adjusting these parameters to a particular problem can be challenging; for the particle control we find a uniform weighting $Q = R = I$ sufficient. The LQR cost function is minimized by a simple proportional feedback rule $\mathbf{u} = K\mathbf{x}$. The LQR gain matrix K can be computed using standard numeric libraries.

In Equation (7) we assumed full access to the states, i.e., $g(\mathbf{x}) = \mathbf{x}$ and $\boldsymbol{\epsilon} = \mathbf{0}$ in Equation (3). More realistic is the case of partial noisy observations. Fortunately, the optimal solution may still be applied by combining the controller with a Kalman filter, which enables estimation of unobserved and denoising of observed states in linear systems. Given the system dynamics, the Kalman filter operation can be applied via a simple matrix-vector multiplication $\tilde{\mathbf{x}} = L\mathbf{y}$, mapping noisy partial observations to full state estimates, which can be used in the LQR loss (7). The combination of LQR with a Kalman filter is called Linear Quadratic Gaussian (LQG). In the next Section we consider the case where linearity and system knowledge cannot be assumed.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

2.4 Modelling the neural system using a recurrent neural network

Research in control theory over the past decades has resulted in a wealth of techniques to solve tasks like the ones described in Section 2.2. Which approach to take depends on the stringency of assumptions we make about the system. At one end of the spectrum, one can compute an optimal control policy analytically if the system is linear and the state equations are known (Section 2.3). The controller (or neural system in our terminology) is then a simple matrix that maps the state observation vector $\mathbf{x}$ to a control signal $\mathbf{u}$. At the other end of the spectrum, one can employ reinforcement learning to find a control policy for a nonlinear system with unknown dynamics and partial noisy observations. In that case, the neural system controller is usually an artificial neural network with a large number of parameters.

Here, we use recurrent neural networks (RNNs) to model the controller. RNNs have a long history in modeling of neural systems [52] and use in control applications [35, 53], making them an ideal candidate in the context of this paper. We will see in Section 2.5 that these networks are compatible with both ends of the spectrum - they can adapt to analytic optimal control laws just as to data-driven policy search through RL. In particular, the statefulness of RNNs turns out to be a crucial component in dealing with partially observed and/or noisy observations.

The continuous-time RNN used here follows work by [52] and is defined by the ordinary differential equation

$$\tau\dot{\mathbf{v}} = -\mathbf{v} + A\mathbf{x} + B\mathbf{u}\,, \tag{8}$$

where the vector $\mathbf{v}$ represents the neurons' membrane potentials, $\mathbf{x} = h(\mathbf{v})$ are the firing rates obtained by passing the membrane potential through a sigmoidal activation function h, and τ is the time constant of charge integration and decay. The neurons integrate inputs $\mathbf{u}$ via synaptic weights B, and are recurrently connected through the matrix A. Layers of such RNN units may be stacked hierarchically, so that the rates $\mathbf{x}$ of one layer become the input $\mathbf{u}$ to the next. The network is further assumed to include a fully-connected readout layer

$$\mathbf{y} = g(\mathbf{x}) = \tanh(C\mathbf{x})\,. \tag{9}$$

Using a (split-step) Euler method, the continuous-time equation (8) can be discretized. By choosing a step size Δt that matches the time constant τ, we obtain an expression for the state evolution that is equivalent to the RNN model commonly used by the AI community:

$$\mathbf{x}_t = \tanh(A\mathbf{x}_{t-1} + B\mathbf{u}_t)\,. \tag{10}$$

Choosing an RNN as controller has the important consequence that our neural system now becomes a dynamical system itself, just as the environment which it aims to control. This way we can unify the proposed methodology for studying the brain-environment interaction.

In terms of free parameters, the model described by equations (9) and (10) is characterized by the synaptic weight matrices A (recurrence), B (input), and C (output). In the context of this paper, we find it useful to interpret the input component $B\mathbf{u}$ as sensory subsystem, the output layer $C\mathbf{x}$ as motor system, and the recurrent term $A\mathbf{x}$ as an association area. Figure 2 illustrates these RNN components within the original control loop of Figure 1. The next Section describes three approaches to fit the model parameters $\boldsymbol{\theta} = \{A, B, C\}$. Note that biases are included in the models but not shown in the equations to simplify notation.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

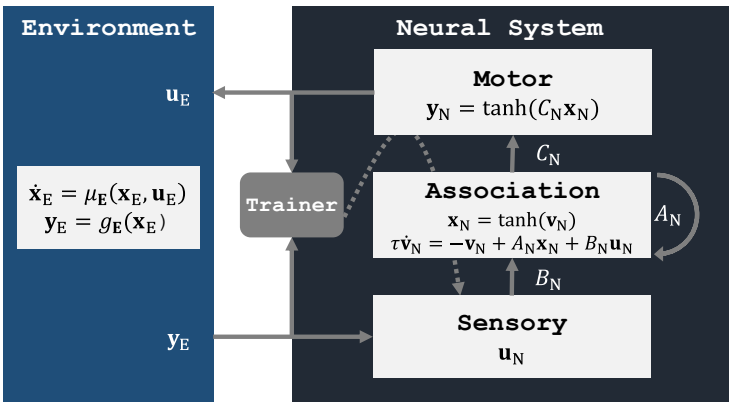


Figure 2: Both the environment and the neural system are modelled as dynamical systems using ordinary or stochastic differential equations. Here, the neural system is represented by an RNN with a sensory input component, a motor output component, and an association area which provides the input-output mapping. The system can be trained using backpropagation through time [63].

2.5 Learning a control policy in the neural system

While control theory provides tools to synthesize controllers for systems that are nonlinear, only partially observable, and whose dynamics are unknown, these methods often include an attempt to linearize the system around an operating point, perform system identification, or include an (extended / unscented) Kalman filter [23] to estimate unobserved states. If successful, these approaches result in a system where the control policy can be derived analytically, and often provide performance guarantees and safety / stability regions. On the other hand, they may not always be applicable or scale well to high-dimensional systems. The considerable complexity of the multi-stage processing pipeline makes some of these approaches difficult to maintain and to transfer between problem domains. Training an RNN controller using RL has the benefit of being conceptually simple and at the same time highly scalable. It assumes no linearity or prior knowledge of the system dynamics. Due to the persistent states of the neural network model, the RNN controller performs temporal integration and can adopt the role of a Kalman filter, estimating unobserved states while filtering out noise in the observed ones.

The present paper aims to study how a neural system (the primary controller) can be stabilized by a secondary controller in the presence of perturbations. In this context we do not attempt to find the best tool from classic control theory to solve the primary control problem. We focus instead on obtaining a neural-network-based controller that can serve as test bed for developing the secondary controller. These considerations motivate the use of an RNN-based controller.

Once settled on an RNN as neural system, the training method is usually going to involve a form of stochastic gradient descent with back-propagation through time [63], which updates the network parameters to optimize some objective function. This generic trainer component is depicted in Figure 2, with the gradient flow marked by a dashed arrow. The question that remains is how motor commands $\mathbf{u}_E$ and environment observations $\mathbf{y}_E$ are combined to obtain a learning signal. To answer it we need to consider the properties of the environment on which the controller acts, and how much knowledge of the environment dynamics we can assume. These considerations, together with the possible training methods, are listed in Table 1 and described in more detail in the following sections.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

Table 1: Selecting a training method for the RNN neural system depends on properties of the environment. Note that observability can be achieved by adding a (non-)linear Kalman filter, provided that the system equations are known.

System must be	known?	linear?	differentiable?	observable?
LQR direct	yes	no	yes	yes
LQG oracle	yes	yes	no	yes
RL	no	no	no	no

2.5.1 Direct optimization of LQR cost

We consider first the most optimistic case, where the state equations (1) are known and we assume that we may differentiate through the environment. Because the neural system is a differentiable RNN, the entire brain-environment model becomes end-to-end differentiable, allowing direct optimization of the parameters using automatic differentiation [5]. That is, after defining a suitable loss function, the neural system can be trained via gradient descent, with the environment in the loop. The samples for training can be generated on the fly: As the neural system steps through the environment, it observes states $\mathbf{x}_t$ and produces motor commands $\mathbf{u}_t$, which are passed to the loss function to generate a learning signal that is propagated back through the network. In the backwards pass, only the parameters in the neural system are updated; the environment parameters are frozen.

We now turn to defining the objective function. For a linear regularization problem like the particle stabilization described in Section 2.2.1, we can simply choose the LQR cost (7). The hypothesis is that the neural system will learn a control policy that follows the analytically derived optimal solution.

In case of partial or noisy observations, the same objective function may be used. We can estimate the latent state explicitly by including a Kalman filter, whose matrix-vector-product operation is straightforward to implement in a neural network and thus integrates well with the direct optimization approach. Alternatively, a Kalman filter can be learned implicitly by the stateful RNN as we demonstrate in Section 3.1.2.

2.5.2 LQG oracle

Next we consider the case that direct optimization on some loss function is not possible because the environment is not differentiable, for instance due to discontinuities in the dynamics. We may still be able to exploit tools from optimal control theory like the LQR and Kalman filter introduced in Section 2.3. As before, we assume knowledge of the state equations (1). In addition, the system should either be linear or linearizable. Full observability is not a requirement as a Kalman filter is available under the current assumptions.

Following Section 2.3, the optimal LQG controller, could be used directly as surrogate neural system. Here, however, we illustrate the case of using the analytically derived optimal controller as teacher for the RNN-based neural system.[1] The LQG teacher can be deployed in the environment to collect a dataset of inputs $\mathbf{y}$ and labels $\mathbf{u}$, which are then used for conventional supervised training of the RNN.

[1] A black-box source of knowledge about a target function is sometimes called an 'oracle' in the machine learning community.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

2.5.3 Reinforcement learning

The most general case, requiring the fewest assumptions, is to use RL for training the neural system RNN. Reinforcement learning is a form of machine learning which only requires reward signals r_t as typically encountered by agents in a realistic environment, rather than labeled input/output pairs as required in supervised learning. An agent in state s learns to take actions a according to a policy $\pi(a, s) = \Pr(a_t = a|s_t = s)$ which maximizes the discounted future reward $R = \sum_t \gamma^t r_t$. It does so by striking a balance between exploring new states and actions, and exploiting those already known to have high value. The optimization of a value function $V_\pi(s) = \mathbb{E}[R|s_0 = s]$ links RL to the field of optimal control and combines it with the benefits and drawbacks of data-driven model-free control.

In terms of the criteria in Table 1, RL is applicable to linear or nonlinear systems with known or unknown dynamics. While there are no guarantees for the learning performance, a noisy partial observation of states together with a scalar reward signal after possibly long time intervals is usually enough to train the neural system.

In the present work, we employ the proximal policy optimization (PPO) algorithm [49], which is an established online policy-gradient RL method that is relatively easy to use while achieving state-of-the-art results. Other benefits of PPO include its support of both discrete and continuous action spaces, and compatibility with recurrent models. We adapted the Stable-Baselines3 [42] implementation of PPO to work with our RNN model.

2.6 Restoring behavior of a perturbed neural system using a secondary controller

The previous sections outlined various approaches of training a neural system to perform meaningful behavior in an environment. With this model at hand we can now study ways to maintain the system performance when parts of the neural system begin to degrade.

Impairment of a biological sensory-motor controller can take many forms, such as cell death or loss of sensor function. Plasticity and a drift in neural representations call for controllers that coadapt with the neural code [51]. Here we implement model perturbation in a generic way by adding univariate Gaussian white noise of various strengths to the synaptic weights of the trained neural system. We consider perturbation of the association (A), sensory (B) and motor (C) populations separately.

Perturbing the neural system can be seen from a control-theoretic perspective as increasing model uncertainty in the SDE (1). The brain-environment loop (Figure 2) is then described by a set of coupled SDEs and can be redefined as a new composite dynamical system. With this unified view we can apply the same control techniques as earlier for the brain-environment interaction, but now with the aim to restore the function of the impaired neural system. Specifically, we add a secondary RNN controller as shown in Figure 3. It follows the same state equations (10) as the neural system RNN, but its components can be interpreted in terms of a neural prosthesis. The input $\mathbf{u}_\mathrm{P}$ takes on the role of a neural recording device such as microelectrodes or neural imagers. The output component $\mathbf{y}_\mathrm{P}$ represents a stimulation device. The hidden RNN layer closes the loop between neural recording and stimulus generation.

The observations $\mathbf{y}_\mathrm{EN}$ of the composite brain-environment system could be a combination of states from both subsystems. For instance, a visual prosthesis would process a camera feed of the environment in addition to neural activity measurements. In our experiments we read out the states $\mathbf{x}_\mathrm{N}$ from the association layer:

$$\mathbf{y}_\mathrm{EN} = C_\mathrm{EN}\mathbf{x}_\mathrm{N} \tag{11}$$

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

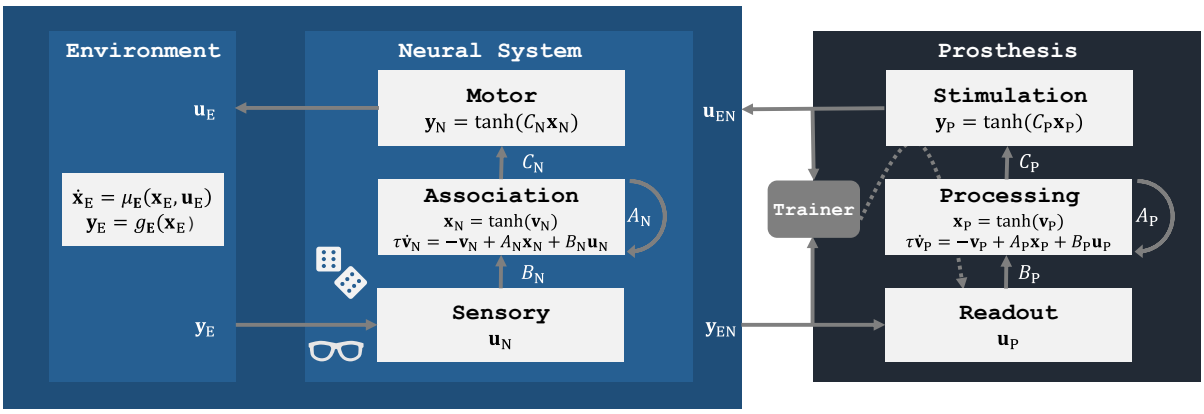


Figure 3: Framework to model a perturbed neural system in interaction with its environment (left) and restoring its performance by training a secondary controller (right).

with $C_{\text{EN}} = I$, assuming full observability initially. We show in Section 3.3 that equivalent performance is achieved when only a small fraction of neurons can be observed and controlled.

Another design choice concerns the feedback control signal $\mathbf{u}_{\text{EN}}$. Following our interpretation of neural stimulation, we apply it in form of charge injected in neurons of the association layer:

$$\tau\dot{\mathbf{v}}_{\text{N}} = -\mathbf{v}_{\text{N}} + A_{\text{N}}\mathbf{x}_{\text{N}} + B_{\text{N}}\mathbf{u}_{\text{N}} + B_{\text{EN}}\mathbf{u}_{\text{EN}} \tag{12}$$

with $B_{\text{EN}} = I$. We use the same training approach for the prosthesis as for the neural system, i.e., if the latter was trained using RL, the former is as well. The neural system parameters remain fixed after perturbation, only the parameters of the secondary controller are allowed to evolve.

3 Results

In this Section we outline first how the neural system was trained on the two tasks described in Section 2.2. The brain model is then perturbed and a prosthetic controller is trained to restore the function of the neural system. Finally, we consider the case of an underactuated system, where the number of controls is lower than the number of neurons. Code to reproduce the results shown here is available online[2].

3.1 Training the neural system

3.1.1 Direct optimization of LQR cost

For stabilizing a drifting particle at the origin, we use a neural system consisting of 50 artificial neurons, which form the association layer in Figure 2, and follow the dynamics (10). The sensory signal $\mathbf{y}_{\text{E}}$ here consists of the full state $\mathbf{x}_{\text{E}}$, i.e., position and velocity of the particle. The motor system is represented by a single neuron which receives input from the association population. Following Equation (9), this neuron produces the real-valued control signal $\mathbf{y}_{\text{N}} = \mathbf{u}_{\text{E}}$ representing the acceleration applied to the particle. The environment (4) is implemented in Python, using the Euler method to approximate the continuous dynamics. The RNN neural system was trained for 10 epochs with the Adam optimizer. During each epoch, the system traversed 10k trajectories in state space, with initial starting points sampled at random from a grid of 100×100 locations.

[2] `https://github.com/rbodo/neural_control`

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

The training result is shown in the first row of Figure 4. The panels on the left show example trajectories of the drifting particle in phase space. When the neural system has not been trained (dashed line), the particle drifts off with uniform or increasing speed along the x-axis. After training directly on the LQR cost (solid line), the particle is reliably brought to rest at the origin (marked by a small cross). The trajectory then closely matches the analytically derived optimal control baseline (dotted line).

3.1.2 LQG oracle

To illustrate the oracle approach, we trained an RNN controller with the same specifications as in Section 3.1.1, with three main differences. First, only the position but not the velocity of the particle was observed, and overlaid with Gaussian white noise of variance 0.1, as in Equation (5). Second, rather than training online, the network was trained on batches of pre-collected data from an LQG optimal controller. Third, rather than using the LQR cost directly, we computed the mean squared difference between the control signal generated by the neural system and the control signal from the LQG oracle.

The second row in Figure 4 illustrates the training result. In this case of partial noisy observability, the neural system again succeeds in stabilizing the particle while following the optimal control baseline closely.

3.1.3 Reinforcement learning

For the RL approach we use the same architecture as in the previous sections. Again, the sensory signal is limited to a noisy version of the particle position. The neural system is trained for 500k episodes with a linearly decaying learning rate of initially $2 \cdot 10^{-4}$. All other hyperparameters of the PPO algorithm were left at their default value.

An important component of training RL agents is to design the reward function. Here, the reward is given by the negative LQR loss, plus a reward of 1 when stabilizing within a distance of 10^{-3} from the origin, at which time the episode terminates as successful. If the particle fails to stabilize within 100 steps, the episode times out and is considered a failure. Row three in Figure 4 illustrates that the sparse RL rewards are sufficient for the RNN to learn the regularization task.

Unlike the particle stabilization problem, the inverted pendulum (cf. Section 2.2.2) is nonlinear. To solve this second task, the only changes we make to the RL training pipeline are to increase the size of the association population to two layers of 128 recurrent neurons each and increase the training duration to 800k episodes. The sensory signal $\mathbf{y}_{\mathrm{E}}$ consists of the position x of the cart and angle θ of the pole, but not their respective velocities. The association population again projects to only one motor neuron, which produces a real-valued force to steer the cart left and right. To simulate the inverted pendulum environment, we use the MuJoCo [57] implementation within the OpenAI Gym interface[3]. The reward is defined as the number of steps for which the pole can be kept upright. A balancing episode is aborted as unsuccessful when the pole angle θ exceeds 11°. Successful episodes time out after 1000 steps, achieving the maximum reward of 1000.

We observe in row four of Figure 4 that the pole tips over within few steps ($r < 30$) when controlled by a randomly initialized RNN. The trained neural system is able to stabilize the pole up to the time limit of 1000 steps. In phase space, a successful trial is often characterized by a harmonic oscillation, and a failure by an outward spiral or straight line.

[3] `https://www.gymlibrary.dev/environments/mujoco/inverted_pendulum/`

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

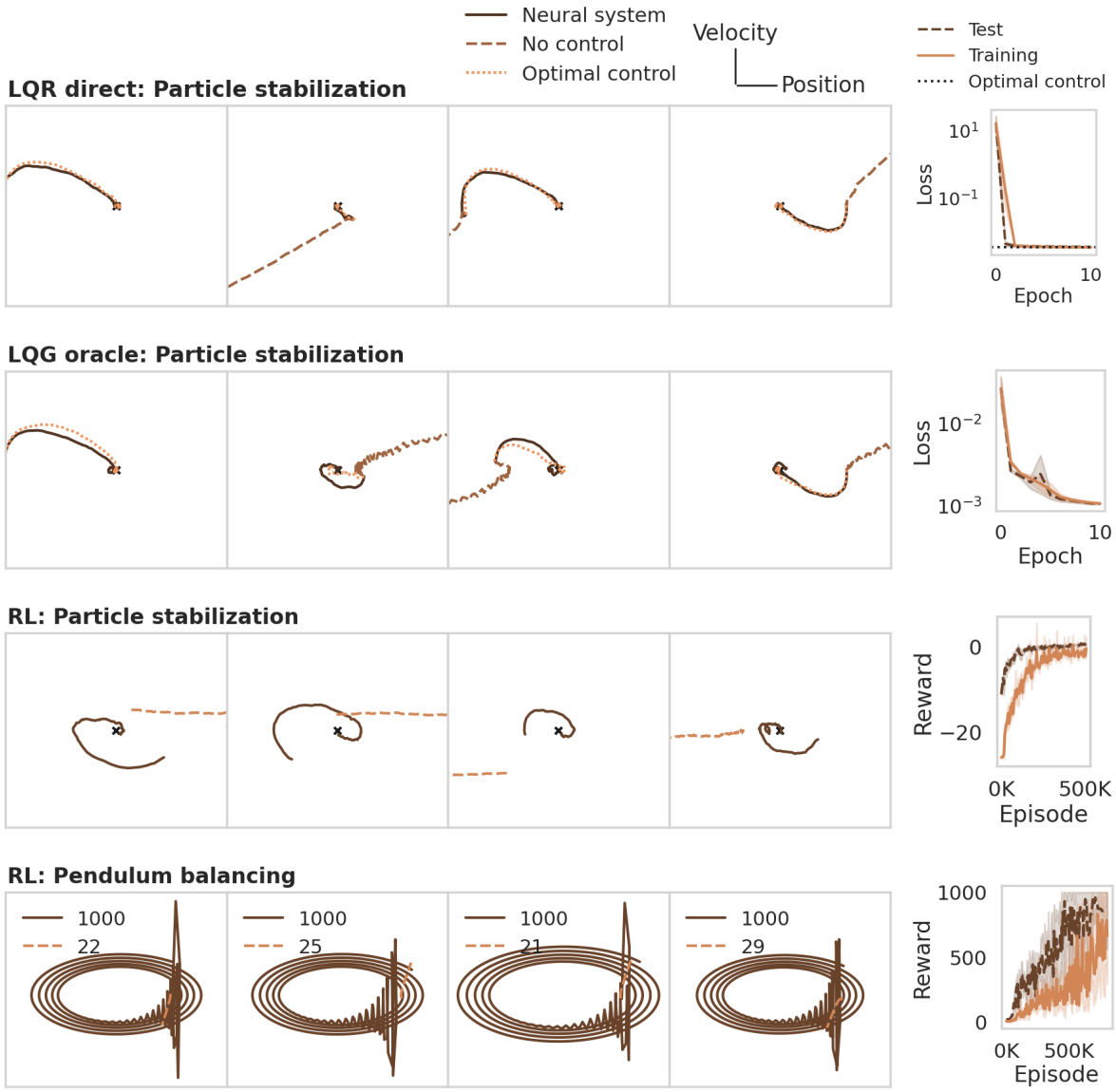


Figure 4: Learning a control policy in the neural system. The panels on the left illustrate the evolution of the environment in state space. The panels on the right show the training curves of the RNN neural system. The rows illustrate the training methods discussed in Section 2.5, applied to the particle stabilization and pendulum balancing problem. The last row also shows the final rewards. In all four cases, the RNN system succeeds in learning a control policy to solve the task.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

3.2 Training the prosthetic controller

The result of training the prosthesis directly on the LQR cost after perturbing sensor, association, and motor populations is shown in Figure 5 for the particle control task. The panels on the left show example trajectories for increasingly strong perturbation levels. The final column compares the performance of the perturbed system with and without a prosthesis against the optimal LQR baseline. The secondary RNN controller is able to restore the behavior of the neural system across a wide range of perturbation levels, regardless where the perturbation was applied.

This direct learning shows what can be achieved under optimistic assumptions (Table 1). In a real-world behavioral setting, the environment dynamics will likely be unknown, nonlinear, noisy, and only partially observable. Then, one can approximate the optimal control solution via reinforcement learning, using only sparse reward signals to learn from. Figure 6 demonstrates that even under these conditions, the prosthesis manages to restore neural system performance. RL training results on the nonlinear pole balancing task are shown in Figure 7.

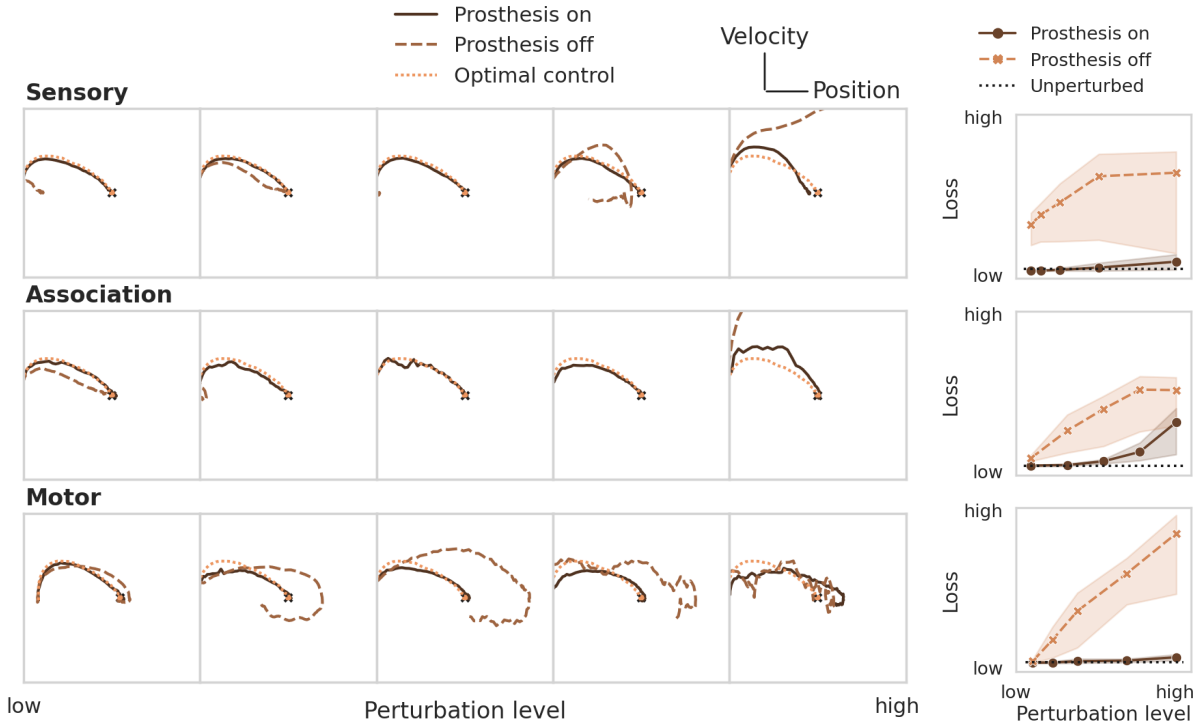


Figure 5: Restoring particle stabilization performance by training a prosthetic controller directly on the LQR cost. The panels on the left show example trajectories at increasing perturbation strengths (from left to right). The rightmost column compares the loss of the system after training the prosthetic controller against the uncontrolled and unperturbed baselines. The three rows illustrate the case of an impaired sensory, association, or motor population.

3.3 Underactuated case: Reducing controllability and observability

In the previous section we assumed access to all neurons in the association layer for recording and stimulation. With a microelectrode array for instance, only a limited number of recording and stimulation locations may be accessible. Due to reactions of the neural

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

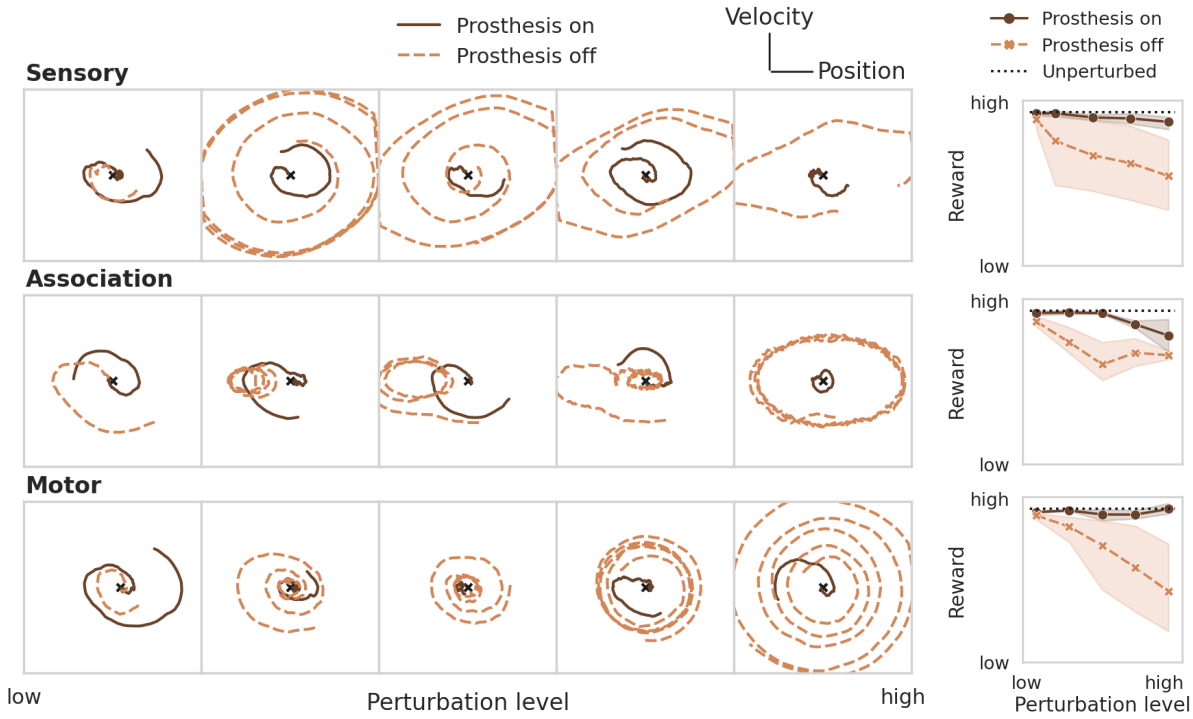


Figure 6: Restoring particle stabilization performance by training a prosthetic controller using RL. See caption of Figure 5 for a more detailed description.

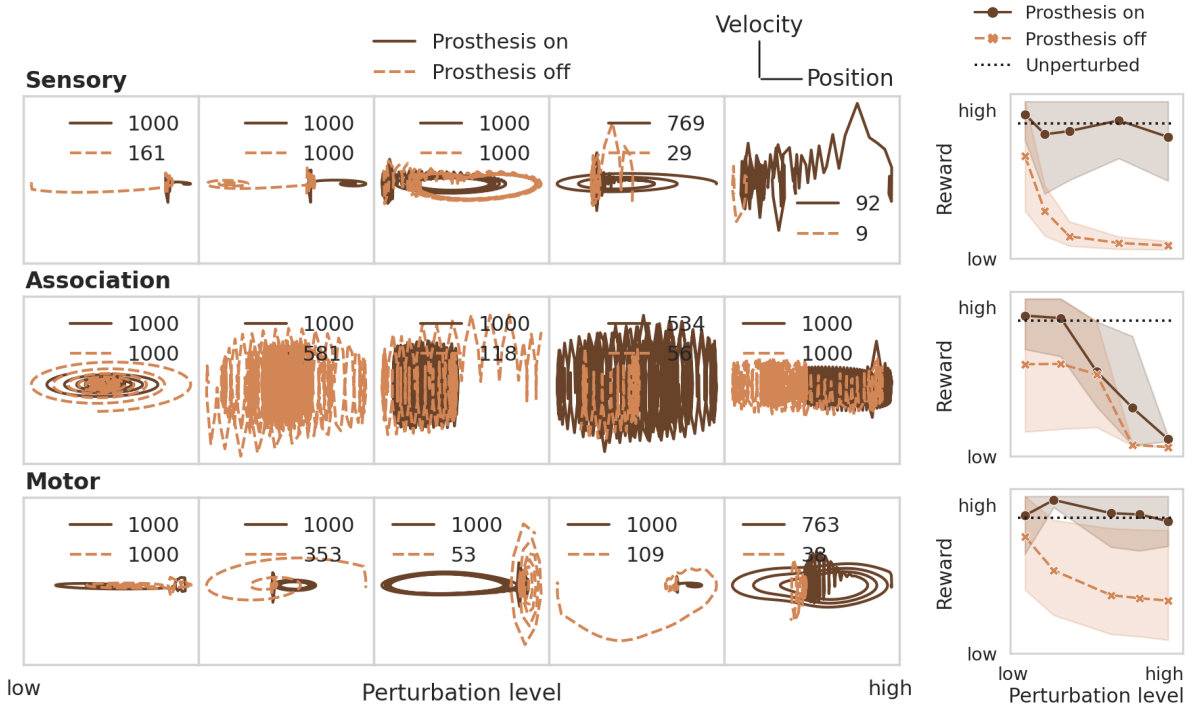


Figure 7: Restoring performance of the neural system on the pole balancing task by training a prosthetic controller using RL. See caption of Figure 5 for a more detailed description.

tissue, the usable set of electrodes may change over time, as well as the set of neurons corresponding to a given electrode. Safety limits for local charge density may add further

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

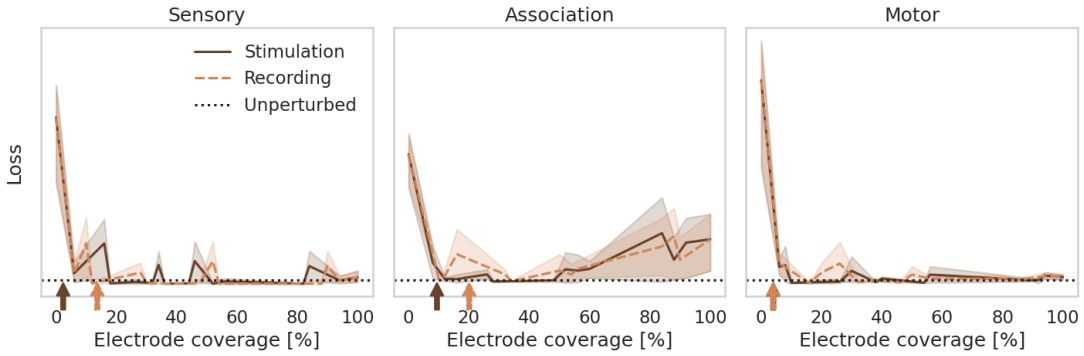


Figure 8: Effect of reducing controllability and observability. The loss is shown when the prosthesis can record from and stimulate only a fraction of neurons in the neural system. The prosthesis restores the performance of the unperturbed neural system when about 10% of the neurons are accessible. This number approximately matches the lower bound estimated from the controllability and observability Gramians (indicated by arrows).

constraints. See [13] for a review. To account for these properties we reduce the number of observable and controllable states in the association population by removing columns in B_{EN} and rows in C_{EN}. The prosthetic controller is then trained as before with the aim to restore the function of the perturbed neural system. Figure 8 shows the performance after training with different degrees of stimulation and recording electrode coverage. The combined brain-prosthesis system maintains the performance of the unperturbed neural system with as little as 10% of the association neurons being observed and controlled.

Control theory provides tools to measure the observability and controllability of a controlled system. In a known linear system, one can use the system matrices A, B, C to analytically compute an observability matrix $\mathcal{O}$ and controllability matrix $\mathcal{C}$ (see e.g. [9]). The system is called observable or controllable, if the corresponding matrix is full rank. In a nonlinear or unknown system, it is possible to estimate an *empirical* Gramian matrix by measuring the system response to stereotypical control impulses [30, 19]. Eigenspectrum analysis of the controllability and observability Gramians reveal directions (given by the eigenvectors with the largest eigenvalues) along which a system can be steered with the least amount of control energy, and observed with the highest signal-to-noise ratio. These directions lie in eigen- rather than state-space, so we cannot use them directly e.g. to select the most suitable subset of electrodes for recording and stimulation. However, the eigenspectrum does provide an indicator for the intrinsic dimensionality of the system and thus the number of electrodes required. Specifically, we can count how many eigenvectors are needed to explain 90% of the variance of the controllability and observability Gramians. This number of dimensions gives a lower bound on the number of electrodes needed for stimulation and recording. We indicate it with an arrow in Figure 8. It turns out that for the particle stabilization problem, this lower bound coincides with the minimum number of electrodes found empirically. In other words, the RNN controller learns to make optimal use of the small number of probes it has available.

4 Discussion

The present work proposes a framework based on dynamical systems to model interactions between the brain, its environment, and a prosthesis. Established methods from optimal control theory and reinforcement learning are applied to train the neural system to perform a task and then train a prosthesis to restore its function when the neural system is

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

perturbed. Here we discuss the design choices of this conceptually simple framework as well as implications for the development of neural controllers.

4.1 Modelling the neural system and prosthesis

In this work we used an RNN to model both neural system and prosthetic controller. Choosing this class of model has several advantages. An RNN can be described by a set of differential equations, which makes the combined environment-brain-prosthesis system uniform in structure and amenable to a common methodology. The statefulness of RNN neurons facilitates working with temporal data, which is ubiquitous in realistic settings. Statefulness has the additional advantage of enabling the RNN to adopt functions of a Kalman filter such as denoising and state estimation [53, 20]. The explicit inclusion of a (possibly nonlinear) Kalman filter becomes optional, thus lifting the requirement to identify the system equations. Further, RNNs have been used to describe actual neural dynamics and thus lend themselves to model sensorimotor impairment.

Other model choices may be desired to increase the biological realism using e.g. Wilson-Cowan equations [64], a spike-based neuron model [22], cortical microcircuits [2], hierarchical [1] or anatomical [33] constraints, or by explicitly including the resistive properties of neural tissue. To be compatible with our framework, the only requirement is that the model can be expressed in a form that supports automatic differentiation using e.g. Jax [8] or PyTorch [39].

Degradation or impairment of this neural system can be modelled by perturbing the RNN parameters. We demonstrated that its function can be restored by adding a secondary controller (modelling a prosthesis), which interacts with a subset of neurons in the neural system. We highlight three approaches to train the neural system and prosthesis, the choice of which depends on the properties and knowledge of the system.

4.2 Choosing the learning method

The three learning approaches considered here each come with their (dis-)advantages. Direct optimization of some cost function (Section 2.5.1) is the most efficient in terms of training data and time, but assumes a differentiable and known system, plus the existence of a suitable loss function. The use of an oracle teacher (Section 2.5.2) assumes a linear and known system, for which an analytic solution from optimal control theory can be computed. Then the training is similarly efficient as the direct method. Both achieve excellent accuracy as measured against the optimal control baseline and are a viable tool for the development of prosthetic controllers in silico. In practical settings, assumptions such as system knowledge or differentiability may not be satisfied, in which case one can resort to reinforcement learning. The outcome of RL-based training (Section 2.5.3) is less reliable and requires substantially more time, training data, and insight into relevant hyperparameters (though for the problems considered here, PPO defaults were sufficient). The major advantage of RL is its broad applicability even if the system equations are unknown, nonlinear, not differentiable, or only a part of the states is observed. Another promising feature of using RL is that it enables moving experimental design beyond mechanistic objectives (of neuronal activation) towards behavioral objectives (of real-life tasks) [27]. Finally, by using controllers based on deep neural networks, we gain access to powerful training tools from deep learning which are beneficial for neural control applications. In practice, the amount of training data will be limited, requiring sample-efficient [18, 26] or few-shot learning [60]. Plasticity in the neural system or a deterioration of the implant may require continuous adaptation of the prosthetic controller for which

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

techniques from continual learning are readily available [38].

4.3 Extensions

The neural system RNN in the present work was trained to perform a control task in a simulated environment without constraining the resulting dynamics to approximate those observed in real neural systems. A next step is therefore to perform system identification using real neural recordings, similar to [41, 54]. Future work will also consider behavioral tasks in higher-dimensional environments, for instance with agents performing visual navigation in a real or virtual environment such as AI Habitat [56] or BEHAVIOR-1K [32]. Subsequent studies may also investigate different types of model degradation, e.g. simulating cell death or complete loss of sensory function.

We have shown in Section 3.3 that controllability and observability Gramians can be used to determine a lower bound on the number of stimulation and recording electrodes of an implant. The Gramians can be calculated analytically if the system is known and linear, but here we used a data-driven estimation technique [30] to demonstrate applicability to unknown nonlinear systems. In a behavioral setup, the required data could be obtained readily via psychophysical experiments. The Gramians indicate directions of efficient control and readout. A promising area of research concerns the combination of our work with the field of sparse sensor and actuator placement [34, 40, 11, 37], which will facilitate the optimal selection of stimulation and recording probes. This approach aligns well with recent studies that reveal stable low-dimensional latent dynamics of cortical neurons in a behavioral task [14].

Recent work [6] used Lyapunov theory to derive conditions for input-to-state stability in a multi-layer RNN. These conditions depend only on the learnable network parameters and can be used for safety verification, i.e., to ensure that outputs lie within a predefined safe region [25]. These results are applicable to the RNNs used here. Together with recent advances in safe RL [50], these techniques enable AI-based controllers that adhere to ethical standards [45, 21, 12] and pave the way to a translation into the clinical setting.

5 Conclusion

In light of these opportunities, we expect that the proposed framework will have a significant impact on the development of neural prostheses as it enables flexible in-silico testing of algorithms for stimulation and closed-loop control before validation using animal models or deployment in patients.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 899287.

References

[1] Ján Antolík et al. "Model Constrained by Visual Hierarchy Improves Prediction of Neural Responses to Natural Scenes". In: *PLOS Computational Biology* 12.6 (June 27, 2016), e1004927. DOI: 10.1371/journal.pcbi.1004927.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

[2] Ján Antolík et al. "Assessment of Optogenetically-Driven Strategies for Prosthetic Restoration of Cortical Vision in Large-Scale Neural Simulation of V1". In: *Scientific Reports* 11.1 (May 24, 2021), p. 10783. DOI: 10.1038/s41598-021-88960-8.

[3] Karl Johan Astrom and Richard M Murray. *Feedback Systems - An Introduction for Scientists and Engineers*. Princeton University Press, 2020.

[4] Andrew Barto, Richard Sutton, and Charles Anderson. "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (Sept. 1983), pp. 834–846. DOI: 10.1109/TSMC.1983.6313077.

[5] Atılım Günes Baydin et al. "Automatic Differentiation in Machine Learning: A Survey". In: *The Journal of Machine Learning Research* 18.1 (Jan. 1, 2017), pp. 5595–5637.

[6] Fabio Bonassi et al. "LSTM Neural Networks: Input to State Stability and Probabilistic Safety Verification". In: Proceedings of Machine Learning Research. 2020, pp. 1–10.

[7] Matthew Botvinick et al. "Deep Reinforcement Learning and Its Neuroscientific Implications". In: *Neuron* 107.4 (Aug. 19, 2020), pp. 603–616. DOI: 10.1016/j.neuron.2020.06.014.

[8] James Bradbury et al. *JAX: Composable Transformations of Python+NumPy Programs*. Version 0.3.13. 2018.

[9] Steven L. Brunton and J. Nathan Kutz. *Data Driven Science & Engineering*. Cambridge University Press, 2017.

[10] Sebastián Castaño-Candamil, Mara Vaihinger, and Michael Tangermann. "A Simulated Environment for Early Development Stages of Reinforcement Learning Algorithms for Closed-Loop Deep Brain Stimulation". In: 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). July 2019, pp. 2900–2904. DOI: 10.1109/EMBC.2019.8857533.

[11] Neil K. Dhingra, Mihailo R. Jovanović, and Zhi-Quan Luo. "An ADMM Algorithm for Optimal Sensor and Actuator Selection". In: 53rd IEEE Conference on Decision and Control. Dec. 2014, pp. 4039–4044. DOI: 10.1109/CDC.2014.7040017.

[12] Juan Manuel Durán and Karin Rolanda Jongsma. "Who Is Afraid of Black Box Algorithms? On the Epistemological and Ethical Basis of Trust in Medical AI". In: *Journal of Medical Ethics* 47.5 (May 1, 2021), pp. 329–335. DOI: 10.1136/medethics-2020-106820. pmid: 33737318.

[13] Eduardo Fernández, Arantxa Alfaro, and Pablo González-López. "Toward Long-Term Communication With the Brain in the Blind by Intracortical Stimulation: Challenges and Future Prospects". In: *Frontiers in Neuroscience* 14 (Aug. 11, 2020), p. 681. DOI: 10.3389/fnins.2020.00681.

[14] Juan A Gallego et al. "Long-Term Stability of Cortical Population Dynamics Underlying Consistent Behavior". In: *Nat. Neurosci.* 23.2 (Feb. 2020), pp. 260–270. DOI: 10.1038/s41593-019-0555-4.

[15] Javier García and Fernando Fernández. "A Comprehensive Survey on Safe Reinforcement Learning". In: *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480.

[16] Shangding Gu et al. *A Review of Safe Reinforcement Learning: Methods, Theory and Applications*. May 20, 2022. DOI: 10.48550/arXiv.2205.10330.

[17] Rodolphe Héliot et al. "Learning in Closed-Loop Brain-Machine Interfaces: Modeling and Experimental Validation". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 40.5 (Oct. 2010), pp. 1387–1397. DOI: 10.1109/TSMCB.2009.2036931.

[18] Matteo Hessel et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 29, 2018). DOI: 10.1609/aaai.v32i1.11796.

[19] Christian Himpe. *Emgr - EMpirical GRamian Framework*. Version 5.99. Apr. 13, 2022. DOI: 10.5281/zenodo.6457616.

[20] Tommy Hosman et al. "BCI Decoder Performance Comparison of an LSTM Recurrent Neural Network and a Kalman Filter in Retrospective Simulation". In: 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER). Mar. 2019, pp. 1066–1071. DOI: 10.1109/NER.2019.8717140.

[21] Marcello Ienca and Pim Haselager. "Hacking the Brain: Brain–Computer Interfacing Technology and the Ethics of Neurosecurity". In: *Ethics and Information Technology* 18.2 (June 1, 2016), pp. 117–129. DOI: 10.1007/s10676-016-9398-9.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

[22] Eugene M. Izhikevich. "Which Model to Use for Cortical Spiking Neurons?" In: *IEEE Transactions on Neural Networks* 15.5 (Sept. 2004), pp. 1063–1070. DOI: 10.1109/TNN.2004.832719.

[23] Simon J. Julier and Jeffrey K. Uhlmann. "New Extension of the Kalman Filter to Nonlinear Systems". In: Signal Processing, Sensor Fusion, and Target Recognition VI. Ed. by Ivan Kadar. SPIE, July 28, 1997. DOI: 10.1117/12.280797.

[24] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[25] Bart Kieboom and Matin Jafarian. *Utility of the Koopman Operator in Output Regulation of Disturbed Nonlinear Systems*. Sept. 30, 2022. DOI: 10.48550/arXiv.2210.00040. arXiv: 2210.00040 [cs, eess].

[26] Burcu Küçükoğlu et al. *Efficient Deep Reinforcement Learning with Predictive Processing Proximal Policy Optimization*. Nov. 11, 2022.

[27] Burcu Küçükoğlu et al. "Optimization of Neuroprosthetic Vision via End-to-End Deep Reinforcement Learning". In: *Int. J. Neural Syst.* 32.11 (Nov. 2022), p. 2250052. DOI: 10.1142/S0129065722500526.

[28] Gautam Kumar et al. "Designing Closed-Loop Brain-Machine Interfaces Using Optimal Receding Horizon Control". In: 2013 American Control Conference. June 2013, pp. 5029–5034. DOI: 10.1109/ACC.2013.6580619.

[29] Manuel Lagang and Lakshminarayan Srinivasan. "Stochastic Optimal Control as a Theory of Brain-Machine Interface Operation". In: *Neural Computation* 25.2 (Feb. 2013), pp. 374–417. DOI: 10.1162/NECO_a_00394.

[30] Sanjay Lall, Jerrold E. Marsden, and Sonja Glavaški. "Empirical Model Reduction of Controlled Nonlinear Systems". In: *IFAC Proceedings Volumes* 32.2 (July 1, 1999), pp. 2598–2603. DOI: 10.1016/S1474-6670(17)56442-3.

[31] Frank L. Lewis, Draguna Vrabie, and Kyriakos G. Vamvoudakis. "Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers". In: *IEEE Control Syst. Mag.* 32.6 (Dec. 2012), pp. 76–105. DOI: 10.1109/MCS.2012.2214134.

[32] Chengshu Li et al. "BEHAVIOR-1K: A Benchmark for Embodied AI with 1,000 Everyday Activities and Realistic Simulation". In: 6th Annual Conference on Robot Learning. Nov. 16, 2022.

[33] Jack Lindsey et al. "A Unified Theory of Early Visual Representations From Retina To Cortex Through Anatomically Constrained Deep CNNs". In: ICLR. 2019, pp. 1–17.

[34] Krithika Manohar et al. "Data-Driven Sparse Sensor Placement for Reconstruction: Demonstrating the Benefits of Exploiting Known Patterns". In: *IEEE Control Systems Magazine* 38.3 (June 2018), pp. 63–86. DOI: 10.1109/MCS.2018.2810460.

[35] Lingheng Meng, Rob Gorbet, and Dana Kulić. *Memory-Based Deep Reinforcement Learning for POMDPs*. Feb. 24, 2021.

[36] B. A Mitchell and Linda R. Petzold. "Control of Neural Systems at Multiple Scales Using Model-Free, Deep Reinforcement Learning". In: *Scientific Reports* 8.1 (July 16, 2018), p. 10721. DOI: 10.1038/s41598-018-29134-x.

[37] Ulrich Münz, Maximilian Pfister, and Philipp Wolfrum. "Sensor and Actuator Placement for Linear Systems Based on H_2 and H_∞ Optimization". In: *IEEE Transactions on Automatic Control* 59.11 (Nov. 2014), pp. 2984–2989. DOI: 10.1109/TAC.2014.2351673.

[38] German I. Parisi et al. "Continual Lifelong Learning with Neural Networks: A Review". In: *Neural Netw.* 113 (May 2019), pp. 54–71. DOI: 10.1016/j.neunet.2019.01.012.

[39] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: Advances in Neural Information Processing Systems 32. Ed. by H Wallach et al. Vancouver, Canada: Curran Associates, Inc., 2019, pp. 8026–8037.

[40] Sérgio Pequito, Soummya Kar, and A. Pedro Aguiar. "A Framework for Structural Input/Output and Control Configuration Selection in Large-Scale Systems". In: *IEEE Transactions on Automatic Control* 61.2 (Feb. 2016), pp. 303–318. DOI: 10.1109/TAC.2015.2437525.

[41] Eric A. Pohlmeyer et al. "Using Reinforcement Learning to Provide Stable Brain-Machine Interface Control despite Neural Input Reorganization". In: *PLoS One* 9.1 (Jan. 30, 2014), e87253. DOI: 10.1371/journal.pone.0087253.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

[42] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

[43] Jason T. Ritt and Shinung Ching. "Neurocontrol: Methods, Models and Technologies for Manipulating Dynamics in the Brain". In: 2015 American Control Conference (ACC). July 2015, pp. 3765–3780. DOI: 10.1109/ACC.2015.7171915.

[44] Pieter R.. Roelfsema, Damiaan Denys, and P. Christiaan Klink. "Mind Reading and Writing: The Future of Neurotechnology". In: *Trends in Cognitive Sciences* 22.7 (2018), pp. 598–610. DOI: 10.1016/j.tics.2018.04.001.

[45] Martin Sand, Juan Manuel Durán, and Karin Rolanda Jongsma. "Responsibility beyond Design: Physicians' Requirements for Ethical Medical AI". In: *Bioethics* 36.2 (2022), pp. 162–169. DOI: 10.1111/bioe.12887.

[46] Timothy D. Sauer and Steven J. Schiff. "Data Assimilation for Heterogeneous Networks: The Consensus Set". In: *Physical Review E: Statistical, Nonlinear and Soft Matter Physics* 79 (5 Pt 1 May 2009), p. 051909. DOI: 10.1103/PhysRevE.79.051909.

[47] Steven J Schiff. *Neural Control Engineering: The Emerging Intersection between Control Theory and Neuroscience*. The MIT Press, 2011. DOI: 10.7551/mitpress/8436.001.0001.

[48] Steven J. Schiff and Timothy D. Sauer. "Kalman Filter Control of a Model of Spatiotemporal Cortical Dynamics". In: *Journal of Neural Engineering* 5.1 (Mar. 2008), pp. 1–8. DOI: 10.1088/1741-2560/5/1/001.

[49] John Schulman et al. *Proximal Policy Optimization Algorithms*. July 20, 2017.

[50] T. D. Simão, Nils Jansen, and M. T. J. Spaan. "AlwaysSafe: Reinforcement Learning without Safety Constraint Violations during Training". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (2021).

[51] Ethan Sorrell, Michael E. Rule, and Timothy O'Leary. "Brain–Machine Interfaces: Closed-Loop Control in an Adaptive System". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (May 3, 2021), pp. 167–189. DOI: 10.1146/annurev-control-061720-012348.

[52] David Sussillo. "Neural Circuits as Computational Dynamical Systems". In: *Current Opinion in Neurobiology* 25 (Apr. 2014), pp. 156–163. DOI: 10.1016/j.conb.2014.01.008.

[53] David Sussillo et al. "A Recurrent Neural Network for Closed-Loop Intracortical Brain-Machine Interface Decoders". In: *Journal of Neural Engineering* 9.2 (Apr. 2012), p. 026027. DOI: 10.1088/1741-2560/9/2/026027.

[54] David Sussillo et al. "Making Brain-Machine Interfaces Robust to Future Neural Variability". In: *Nature Communications* 7 (Dec. 13, 2016), p. 13749. DOI: 10.1038/ncomms13749.

[55] Richard Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 2020.

[56] Andrew Szot et al. "Habitat 2.0: Training Home Assistants to Rearrange Their Habitat". In: Advances in Neural Information Processing Systems. Jan. 15, 2022.

[57] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A Physics Engine for Model-Based Control". In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Oct. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[58] René Traoré et al. "DisCoRL: Continual Reinforcement Learning via Policy Distillation". In: Advances in Neural Information Processing Systems. July 11, 2019.

[59] Gido M. van de Ven, Hava T. Siegelmann, and . Tolias. "Brain-Inspired Replay for Continual Learning with Artificial Neural Networks". In: *Nature Communications* 11.1 (Aug. 13, 2020), p. 4069. DOI: 10.1038/s41467-020-17866-2.

[60] Yaqing Wang et al. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Computing Surveys* 53.3 (June 12, 2020), pp. 1–34. DOI: 10.1145/3386252.

[61] Zhi Wang, Chunlin Chen, and Daoyi Dong. "Lifelong Incremental Reinforcement Learning With Online Bayesian Inference". In: *IEEE Transactions on Neural Networks and Learning Systems* PP (Feb. 11, 2021). DOI: 10.1109/TNNLS.2021.3055499.

[62] Christopher J. C. H. Watkins and Peter Dayan. "Technical Note: Q-Learning". In: *Machine Learning* 8.3 (May 1, 1992), pp. 279–292. DOI: 10.1023/A:1022676722315.

bioRxiv preprint doi: https://doi.org/10.1101/2022.12.24.521852; this version posted December 24, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

[63] Paul John Werbos. "Backpropagation through Time: What It Does and How to Do It". In: *Proceedings of the IEEE* 78.10 (Oct. 1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[64] Hugh Robert Wilson and Jack Cowan. "Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons". In: *Biophysical Journal* 12.1 (Jan. 1972), pp. 1–24. pmid: 4332108.

[65] Jan Manuel Wülfing et al. "Adaptive Long-Term Control of Biological Neural Networks with Deep Reinforcement Learning". In: *Neurocomputing* 342 (May 21, 2019), pp. 66–74. DOI: 10.1016/j.neucom.2018.10.084.