TYPE Original Research
PUBLISHED 08 March 2023
DOI 10.3389/fnins.2023.1141884

OPEN ACCESS

EDITED BY
Can A. Yucesoy,
Boğaziçi University, Türkiye

REVIEWED BY
Sinan Öncü,
Boğaziçi University, Türkiye
Vinícius Rosa Cota,
Italian Institute of Technology (IIT), Italy

*CORRESPONDENCE
Bodo Rueckauer
✉ bodo.rueckauer@donders.ru.nl

SPECIALTY SECTION
This article was submitted to Neural Technology, a section of the journal Frontiers in Neuroscience

RECEIVED 10 January 2023
ACCEPTED 20 February 2023
PUBLISHED 08 March 2023

CITATION
Rueckauer B and van Gerven M (2023) An *in-silico* framework for modeling optimal control of neural systems.
Front. Neurosci. 17:1141884.
doi: 10.3389/fnins.2023.1141884

An *in-silico* framework for modeling optimal control of neural systems

Bodo Rueckauer* and Marcel van Gerven

Department of Artificial Intelligence, Donders Institute for Brain, Cognition and Behavior, Radboud University, Nijmegen, Netherlands

Introduction: 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.

Methods: 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.

Results: 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.

Discussion: 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.

KEYWORDS

control theory, reinforcement learning, dynamical systems, neurotechnology, neural prosthesis

1. Introduction

Closed-loop recording and stimulation in neurotechnology is becoming increasingly feasible (Roelfsema et al., 2018) and calls for algorithmic advances in developing adaptive, self-calibrating feedback controllers (Ritt and Ching, 2015). Together with patient-specific models of neural systems, closed-loop controllers have the potential to tackle many of the challenges identified within the NeurotechEU initiative[1], including long-term stable prosthetics and precise neuromodulation in digital brain health pipelines. However, 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.

1 https://theneurotech.eu/mission-vision/

We propose a control-theoretic framework to study the stability and controllability of biologically motivated artificial neural systems (Sussillo, 2014) embedded in simulated environments. From a high-level perspective, this framework models the brain-machine-environment interaction. We first consider the problem of modeling 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.

Neural systems, from single neurons to large-scale populations, are characterized by complex dynamics that can be challenging to model and control (Ritt and Ching, 2015). Classical control theory (Khalil, 2002; Brunton and Kutz, 2017; Astrom and Murray, 2020) provides powerful tools for designing control laws and has found numerous applications in neurotechnology, for instance in closed-loop brain-machine interface (BMI) control of robotic arms or computer cursors (Shanechi et al., 2016), model-predictive control for epileptic seizure mitigation (Chatterjee et al., 2020), and a mechanistic explanation of the brain's transition between cognitive states (Gu et al., 2015). A particularly successful application of closed-loop control is the treatment of Parkinson's disease *via* deep brain stimulation. There, pathological beta-band oscillatory activity can be suppressed at a desired target level using threshold-based, proportional-integral, or self-tuning controllers (Fleming et al., 2020a,b). A canonical approach linking control theory to neuroscience and biomedicine has been established by Schiff (2011), where models of spatiotemporal cortical dynamics are combined with Kalman filters to estimate unobserved states and track unknown or drifting model parameters. Groups in the neuromorphic community have recently contributed to this work by implementing biologically plausible operations and learning rules for state-estimation and control (Friedrich et al., 2021; Linares-Barranco et al., 2022), as well as neuromorphic BMI circuits (Donati and Indiveri, 2023), which promise better biocompatibility at low-power operation.

Some of the challenges recurring in many of the mentioned approaches are the assumption of linear (-izable) or lower-dimensional systems, knowledge of the underlying dynamics, or availability of a desired target state (as in DBS for Parkinson's disease). The present paper makes two main contributions addressing these limitations. First, we propose the consistent use of dynamical systems to model the brain, the environment, and the prosthesis. Aside from unifying the methodology, this choice enables flexible experimentation with models of varying degrees of realism. Here we showcase the use of recurrent neural networks (RNNs) as simple, highly scalable building blocks for both the neural system and the prosthesis. Second, we stepwise remove the assumptions of linearity, system knowledge, full observability, and supervised target state, by using reinforcement learning (RL) (Sutton and Barto, 2020) both for system identification and synthesizing the prosthesis controller.

Reinforcement learning is particularly suited for the proposed *in-silico* control framework 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 (Watkins and Dayan, 1992), 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 (Lewis et al., 2012). Examples for the use of RL in neural control include (Pohlmeyer et al., 2014; Sussillo et al., 2016) (*in vivo*), Wülfing et al. (2019) (*in vitro*), Mitchell and Petzold (2018) and Castaño-Candamil et al. (2019) (*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) (Schiff and Sauer, 2008; Sauer and Schiff, 2009), or on the brain-prosthesis loop (e.g., to train a neural decoder to control a prosthetic limb) (Héliot et al., 2010; Kumar et al., 2013; Lagang and Srinivasan, 2013). 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 test new computational models of brain function (Wander and Rao, 2014), exploit recent advances in brain-inspired RL (Botvinick et al., 2020), safe RL (García and Fernández, 2015; Gu et al., 2022), few-shot learning (Wang et al., 2020), and continual learning (Parisi et al., 2019; Traoré et al., 2019; van de Ven et al., 2020; Wang et al., 2021).

2. Materials and 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

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

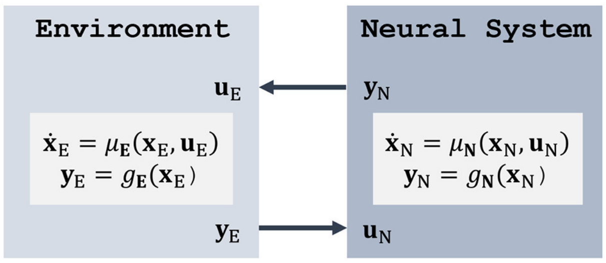
Environment
Neural System
$\mathbf{u}_E$
$\mathbf{y}_N$
$\dot{\mathbf{x}}_E = \mu_E(\mathbf{x}_E, \mathbf{u}_E)$
$\mathbf{y}_E = g_E(\mathbf{x}_E)$
$\dot{\mathbf{x}}_N = \mu_N(\mathbf{x}_N, \mathbf{u}_N)$
$\mathbf{y}_N = g_N(\mathbf{x}_N)$
$\mathbf{y}_E$
$\mathbf{u}_N$

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.

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, 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

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

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 (Barto et al., 1983). 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 known linear 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 (Equation 7). The combination of LQR with a Kalman filter is called Linear Quadratic Gaussian (LQG) control. In the next section, we consider the case where linearity and system knowledge cannot be assumed.

2.4. Modeling 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 (Sussillo, 2014) and use in control applications (Sussillo et al., 2012; Meng et al., 2021), 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 Sussillo (2014) 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, 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.

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 (Julier and Uhlmann, 1997) 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 (Werbos, 1990), which updates the network parameters to optimize some objective function. This generic trainer component is depicted in Figure 2, with the gradient

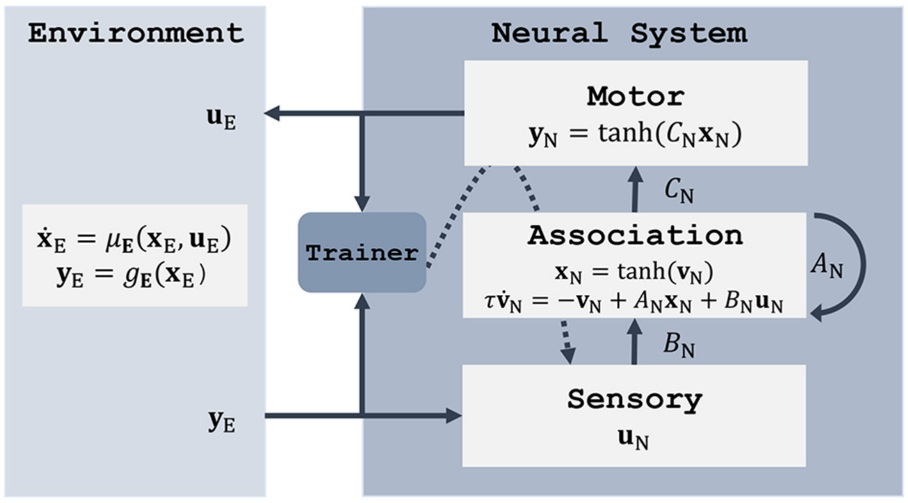
Environment
$\mathbf{u}_E$
$\dot{\mathbf{x}}_E = \mu_E(\mathbf{x}_E, \mathbf{u}_E)$
$\mathbf{y}_E = g_E(\mathbf{x}_E)$
$\mathbf{y}_E$
Trainer
Neural System
Motor
$\mathbf{y}_N = \tanh(C_N \mathbf{x}_N)$
C_N
Association
$\mathbf{x}_N = \tanh(\mathbf{v}_N)$
$\tau \dot{\mathbf{v}}_N = -\mathbf{v}_N + A_N \mathbf{x}_N + B_N \mathbf{u}_N$
A_N
B_N
Sensory
$\mathbf{u}_N$

FIGURE 2
Both the environment and the neural system are modeled 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 (Werbos, 1990).

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.

2.5.1. Direct optimization of LQR cost

We consider first the most optimistic case, where the state (Equation 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 (Baydin et al., 2017). 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[2]. 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.

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. At time t, an agent in state $s_t = s$ learns to take an action $a_t = a$ according to a policy $\pi(a, s) = \Pr(a_t = a | s_t = s)$ which maximizes

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

TABLE 1 Selecting a training method for the RNN neural system depends on properties of the environment.

System must be	Known?	Linear?	Differentiable?	Observable?
LQR direct	Yes	No	Yes	Yes
LQG oracle	Yes	Yes	No	Yes
RL	No	No	No	No

Note that observability can be achieved by adding a (non-)linear Kalman filter, provided that the system equations are known.

the discounted future reward $R = \sum_t \gamma^t r_t$, where r_t is the reward at time 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 $V_\pi(s) = \mathbb{E}[R|s_0 = s]$, starting from state s_0, 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 (Schulman et al., 2017), 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 (Raffin et al., 2021) 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 (Sorrell et al., 2021). 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 (Equation 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 (Equation 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 an external sensor (such as a camera or microphone), or a neural recording device such as a microelectrode array, or a combination of sensors. The output component $\mathbf{y}_\mathrm{P}$ represents an action on the environment (e.g., *via* a prosthetic limb), the brain (e.g., *via* stimulation electrodes), or both. 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:

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

With $C_\mathrm{EN} = I$, 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 $\mathbf{u}_\mathrm{EN}$. Following our interpretation of neural stimulation, we apply it in form of charge injected in neurons of the association layer:

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

With $B_\mathrm{EN} = I$. 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 are available online[3].

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

3 https://github.com/rbodo/neural_control

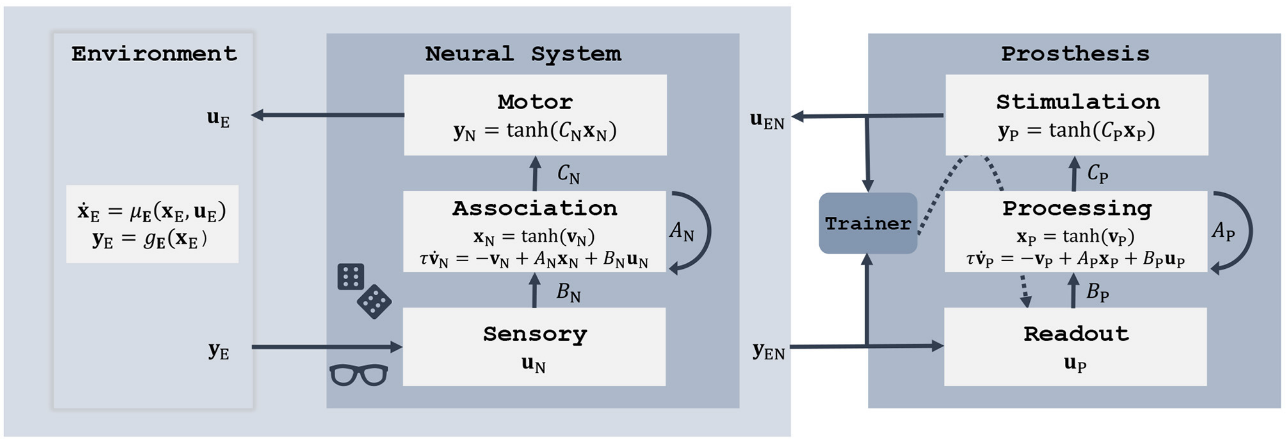
Environment
$\mathbf{u}_\mathrm{E}$
$\dot{\mathbf{x}}_\mathrm{E} = \mu_\mathbf{E}(\mathbf{x}_\mathrm{E}, \mathbf{u}_\mathrm{E})$
$\mathbf{y}_\mathrm{E} = g_\mathbf{E}(\mathbf{x}_\mathrm{E})$
$\mathbf{y}_\mathrm{E}$
Neural System
Motor
$\mathbf{y}_\mathrm{N} = \tanh(C_\mathrm{N}\mathbf{x}_\mathrm{N})$
C_N
Association
$\mathbf{x}_\mathrm{N} = \tanh(\mathbf{v}_\mathrm{N})$
$\tau\dot{\mathbf{v}}_\mathrm{N} = -\mathbf{v}_\mathrm{N} + A_\mathrm{N}\mathbf{x}_\mathrm{N} + B_\mathrm{N}\mathbf{u}_\mathrm{N}$
A_N
B_N
Sensory
$\mathbf{u}_\mathrm{N}$
$\mathbf{u}_\mathrm{EN}$
$\mathbf{y}_\mathrm{EN}$
Trainer
Prosthesis
Stimulation
$\mathbf{y}_\mathrm{P} = \tanh(C_\mathrm{P}\mathbf{x}_\mathrm{P})$
C_P
Processing
$\mathbf{x}_\mathrm{P} = \tanh(\mathbf{v}_\mathrm{P})$
$\tau\dot{\mathbf{v}}_\mathrm{P} = -\mathbf{v}_\mathrm{P} + A_\mathrm{P}\mathbf{x}_\mathrm{P} + B_\mathrm{P}\mathbf{u}_\mathrm{P}$
A_P
B_P
Readout
$\mathbf{u}_\mathrm{P}$

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)**.

association layer in Figure 2, and follow the dynamics (Equation 10). The sensory signal $\mathbf{y}_\mathrm{E}$ here consists of the full state $\mathbf{x}_\mathrm{E}$, 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 $\mathbf{y}_\mathrm{N} = \mathbf{u}_\mathrm{E}$ representing the acceleration applied to the particle. The environment (Equation 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.

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 (Todorov et al., 2012) implementation within the OpenAI Gym interface[4]. 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 1,000 steps, achieving the maximum reward of 1,000.

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.

4 https://www.gymlibrary.dev/environments/mujoco/inverted_pendulum/

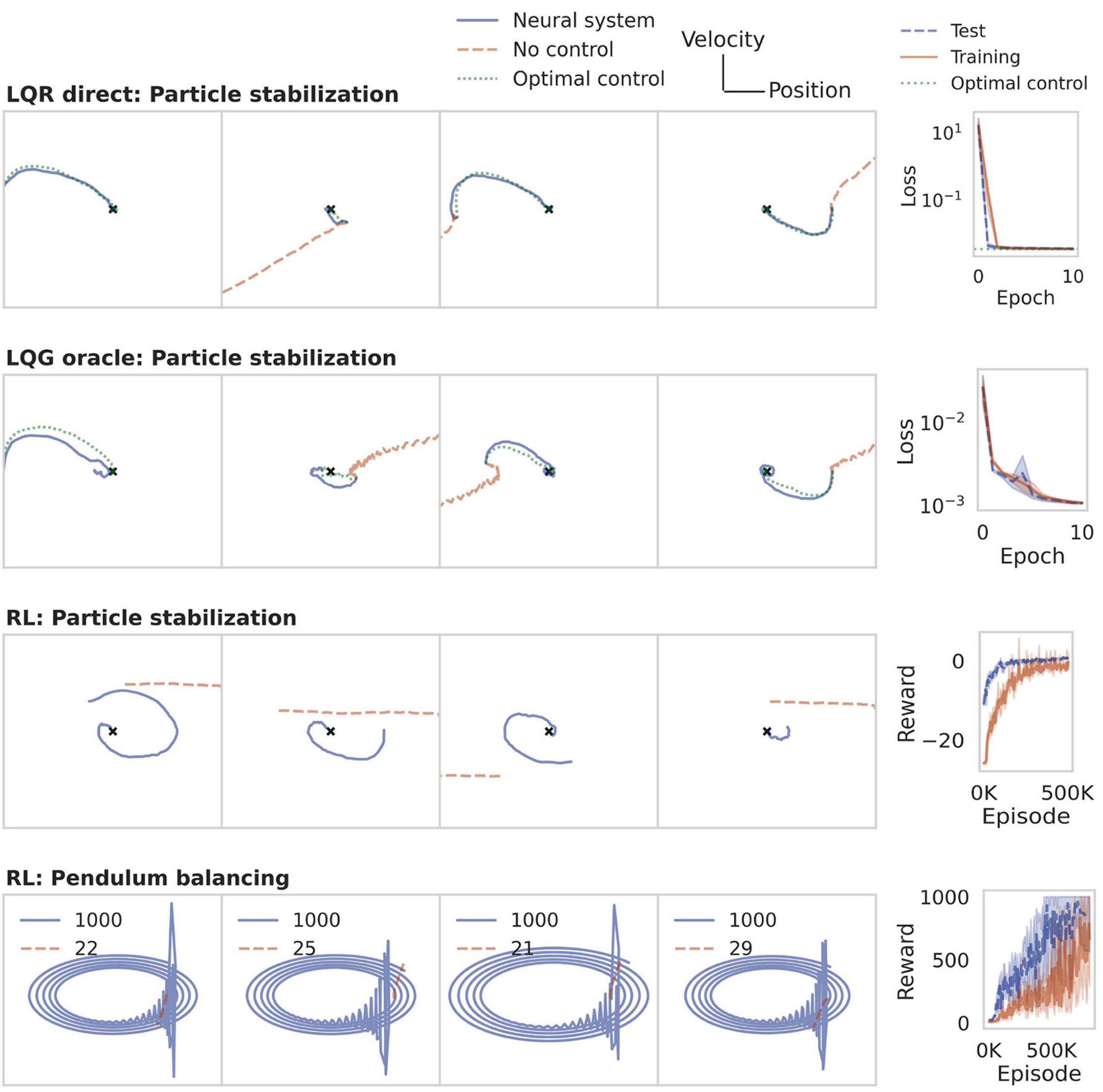

FIGURE 4
Learning a control policy in the neural system. The panels on the left illustrate the evolution of the environment in state space. The panels on the right show the training curves of the RNN neural system. The rows illustrate the training methods discussed in Section 2.5, applied to the particle 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.

The trained neural system is able to stabilize the pole up to the time limit of 1,000 steps. In phase space, a successful trial is often characterized by a harmonic oscillation, and a failure by an outward spiral or straight line.

3.2. 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.

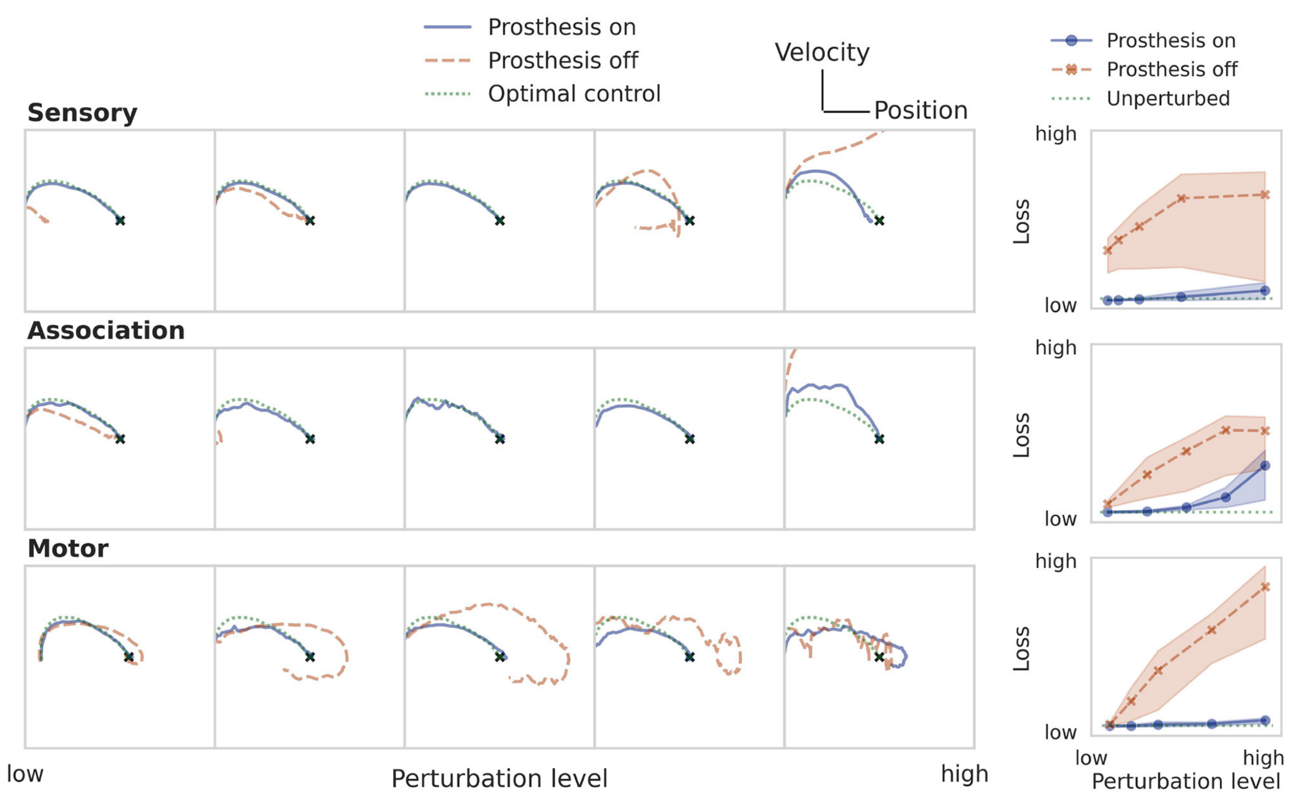

FIGURE 5
Restoring particle stabilization performance by training a prosthetic controller directly on the LQR cost. The panels on the left show example trajectories at increasing perturbation strengths (from left to right). The rightmost column compares the loss of the system after training the prosthetic 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 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 constraints. See Fernández et al. (2020) 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., Brunton and Kutz, 2017). 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 (Lall et al., 1999; Himpe, 2022). 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.

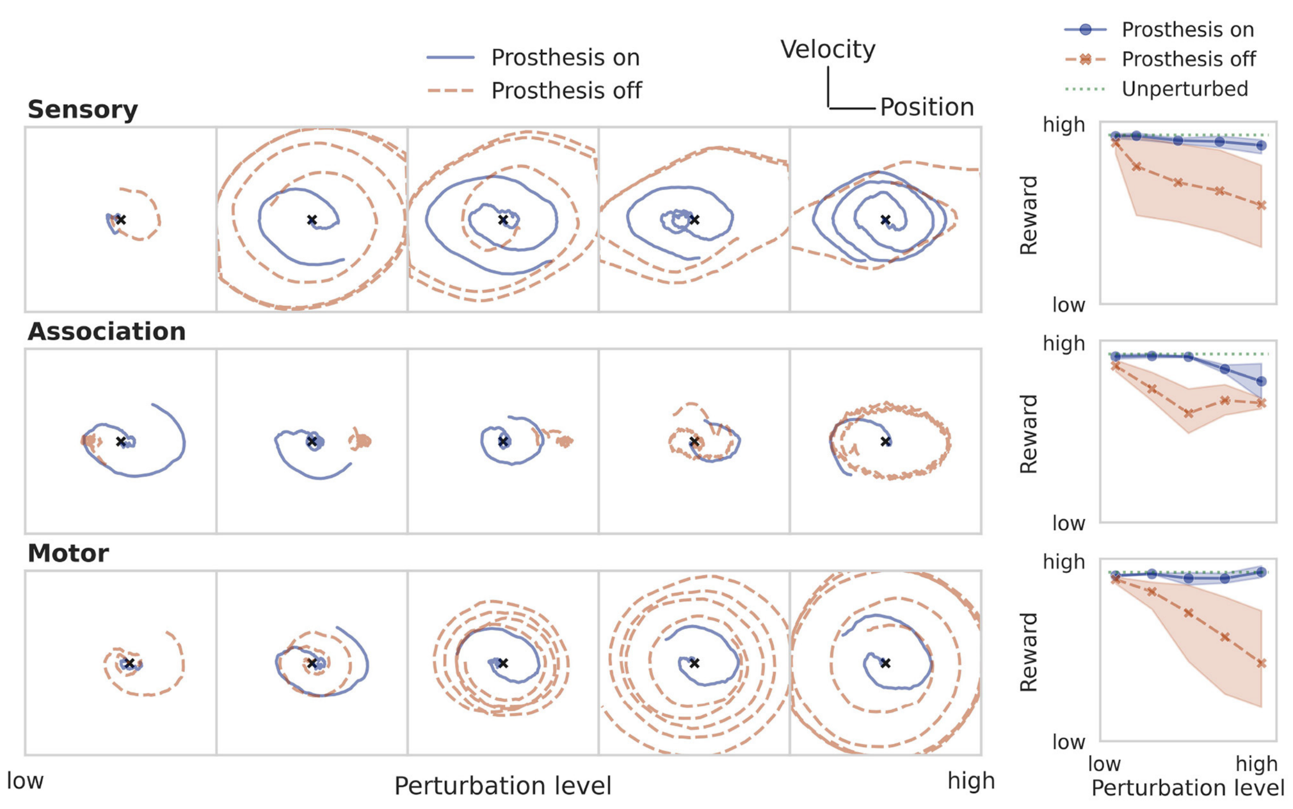

FIGURE 6
Restoring particle stabilization performance by training a prosthetic controller using RL. See caption of Figure 5 for a more detailed description.

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

4.1. Modeling 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 (Sussillo et al., 2012; Hosman et al., 2019). 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.

Degradation or impairment of this neural system can be modeled by perturbing the RNN parameters. We demonstrated that its function can be restored by adding a secondary controller (modeling a prosthesis), which interacts with a subset of neurons in the neural system.

Beyond using RNNs for the neural system, alternative model choices may be desired to increase the biological realism using e.g., Wilson-Cowan equations (Wilson and Cowan, 1972), a spike-based neuron model (Izhikevich, 2004), cortical microcircuits (Antolík et al., 2021), hierarchical (Antolík et al., 2016), or anatomical (Lindsey et al., 2019) 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 (Bradbury et al., 2018) or PyTorch (Paszke et al., 2019).

4.2. Choosing the learning method

We highlight three approaches to train the neural system and prosthesis, the choice of which depends on the properties and knowledge of the system. Direct optimization of some cost function (Section 2.5.1) is the most efficient in terms of training

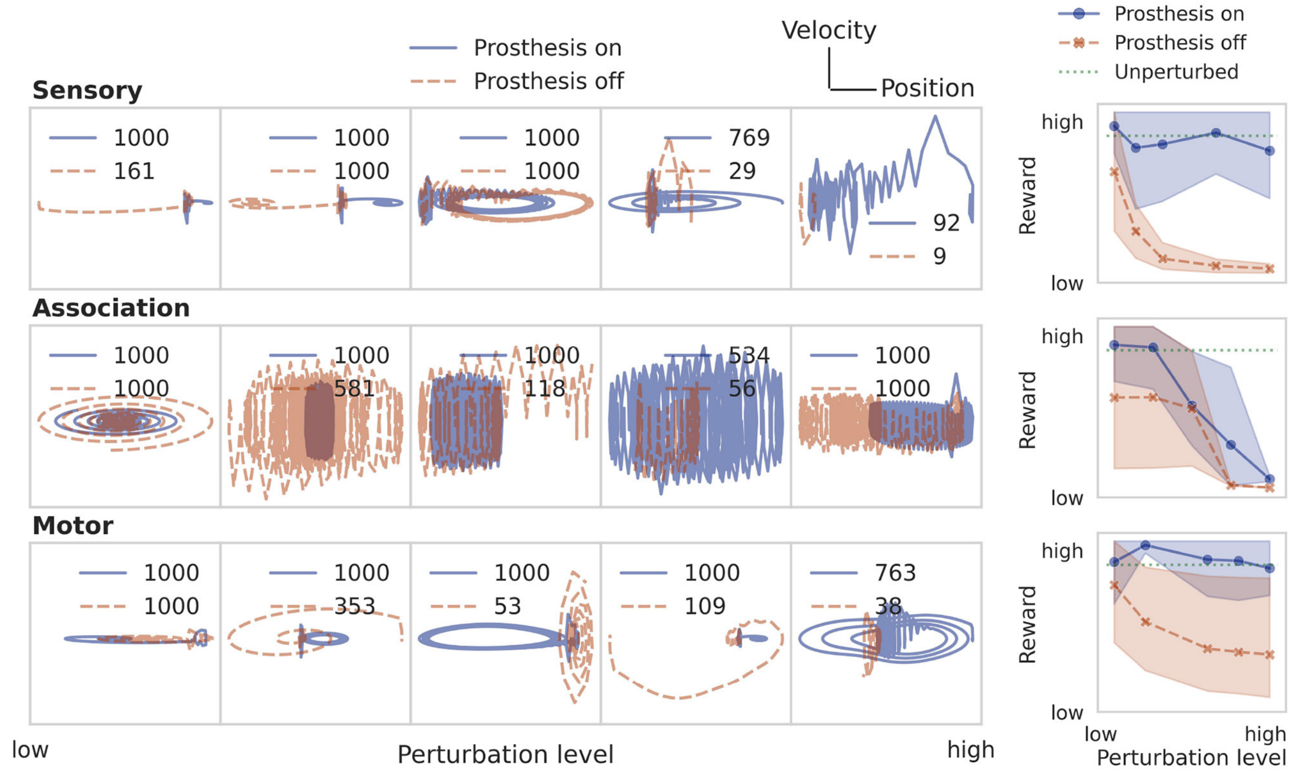

FIGURE 7
Restoring performance of the neural system on the pole balancing task by training a prosthetic controller using RL. See caption of Figure 5 for a more detailed description.

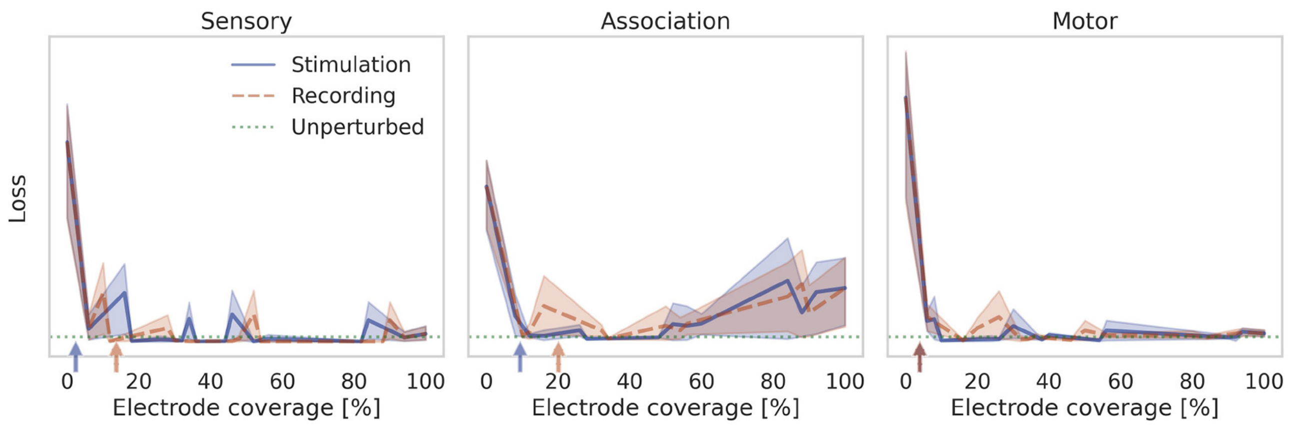

FIGURE 8
Effect of reducing controllability and observability. The loss is shown when the prosthesis can record from and stimulate only a fraction of neurons in the neural system. The prosthesis restores the performance of the unperturbed neural system when about 10% of the neurons are accessible. This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

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) toward behavioral objectives (of real-life tasks) (Küçükoğlu et al., 2022b). Further, 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 (Hessel et al., 2018; Küçükoğlu et al., 2022a) or few-shot learning (Wang et al., 2020).

Plasticity in the neural system or a deterioration of the implant is currently a major limiting factor for neurotechnology (Fernández et al., 2020; Sorrell et al., 2021). Periodic refinement of our neural system model would ensure that we maintain an accurate digital twin of the patient and enables updating the controller to take into account neural plasticity. A promising but still under-explored approach is the inclusion of techniques from continual learning (Parisi et al., 2019), which reduces "catastrophic forgetting" of previous knowledge when the model is updated. Other groups have successfully applied reinforcement learning to adapt the controller to neural variability and reorganization of neural inputs (e.g., neurons appearing or being lost amongst electrode recordings) by training online with the animal in the loop (Pohlmeyer et al., 2014) or offline on a large and diverse dataset (Sussillo et al., 2016). On the other hand, neural plasticity could be seen as a feature to be exploited. An exciting application of our framework would be to model the effect of repeated stimulation on neural plasticity, thereby guiding efforts in neurorehabilitation (Jackson et al., 2006).

4.3. Limitations and future work

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. Within the scope of this paper, the restoration of functional performance was sufficient. An accurate reconstruction of real neural dynamics will become relevant for instance to detect and suppress pathological band-power features in Parkinson's disease (Castaño-Candamil et al., 2020). A more mechanistic model of the neural system (c.f. Section 4.1) will be an important step toward neurobiological validation of our framework. We are therefore currently using electrophysiological recordings of behaving animals to identify the neural system model. By perturbing the resulting model as in Section 2.6 and repeating the training with a prosthesis model in the loop, we can design a prosthesis controller fully *in silico*. This controller can then be deployed in the original behavioral setting for validation and further refinement. We expect that such an offline pretraining will improve the efficacy of the prosthesis while reducing the burden on the animal or patient compared to fully online approaches (Pohlmeyer et al., 2014).

A limitation of our contribution is the small state space of the environments used. We are currently exploring behavioral tasks in higher-dimensional spaces with agents performing visual navigation in a virtual environment such as AI Habitat (Szot et al., 2022) and BEHAVIOR-1K (Li et al., 2022). These studies 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 (Lall et al., 1999) 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 (Dhingra et al., 2014; Münz et al., 2014; Pequito et al., 2016; Manohar et al., 2018), 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 (Gallego et al., 2020).

Recent work (Bonassi et al., 2020) 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 (Kieboom and Jafarian, 2022). These results are applicable to the RNNs used here. Together with recent advances in safe RL (Simão et al., 2021), these techniques enable AI-based controllers that adhere to ethical standards (Ienca and Haselager, 2016; Durán and Jongsma, 2021; Sand et al., 2022) and pave the way to a translation into the clinical setting.

Aside from its uses in research, we found the proposed framework to be valuable in teaching undergraduate students fundamental concepts from optimal control, dynamical systems, recurrent neural networks, and reinforcement learning. The code base can be converted directly to hands-on assignments applying these concepts to neurotechnology and could become a useful resource in the NeurotechEU initiative.

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, reducing the burden of *in-vivo* testing in animal models and/or human subjects.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

BR implemented the methods, performed the experiments and analysis, and wrote the initial draft of the manuscript. BR and MG contributed to conception and design of the study, contributed to manuscript revision, read, and approved the submitted version.

Funding

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

Acknowledgments

We thank SURF (https://www.surf.nl) for the support in using the Dutch National Supercomputer Snellius.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Antolík, J., Hofer, S. B., Bednar, J. A., and Mrsic-Flogel, T. D. (2016). Model constrained by visual hierarchy improves prediction of neural responses to natural scenes. *PLoS Comput Biol.* 12, e1004927. doi: 10.1371/journal.pcbi.1004927 "

Antolík, J., Sabatier, Q., Galle, C., Frégnac, Y., and Benosman, R. (2021). Assessment of optogenetically-driven strategies for prosthetic restoration of cortical vision in large-scale neural simulation of V1. *Sci. Rep.* 11, 10783. doi: 10.1038/s41598-021-88960-8

Astrom, K. J., and Murray, R. M. (2020). *Feedback Systems-an Introduction for Scientists and Engineers*. Princeton, NJ: Princeton University Press.

Barto, A., Sutton, R., and Anderson, C. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* SMC-13, 834–846. doi: 10.1109/TSMC.1983.6313077

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2017). Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.* 18, 5595–5637. doi: 10.48550/arXiv.1502.05767

Bonassi, F., Terzi, E., Farina, M., and Scattolini, R. (2020). "LSTM neural networks: input to state stability and probabilistic safety verification," in *Proceedings of Machine Learning Research*, 1–10.

Botvinick, M., Wang, J. X., Dabney, W., Miller, K. J., and Kurth-Nelson, Z. (2020). Deep reinforcement learning and its neuroscientific implications. *Neuron* 107, 603–616. doi: 10.1016/j.neuron.2020.06.014

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., etale (2018). *JAX: Composable Transformations of Python+NumPy Programs*.

Brunton, S. L., and Kutz, J. N. (2017). *Data Driven Science and Engineering*. Cambridge: Cambridge University Press. "

Castaño-Candamil, S., Piroth, T., Reinacher, P., Sajonz, B., Coenen, V. A., and Tangermann, M. (2020). Identifying controllable cortical neural markers with machine learning for adaptive deep brain stimulation in parkinson's disease. *Neuroimage Clin.* 28, 102376. doi: 10.1016/j.nicl.2020.102376 "

Castaño-Candamil, S., Vaihinger, M., and Tangermann, M. (2019). "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)* (Berlin: IEEE), 2900–2904."

Chatterjee, S., Romero, O., Ashourvan, A., and Pequito, S. (2020). Fractional-order model predictive control as a framework for electrical neurostimulation in epilepsy. *J. Neural Eng.* 17, abc740. doi: 10.1088/1741-2552/abc740

Dhingra, N. K., Jovanović,, M. R., and Luo, Z.-Q. (2014). "An ADMM algorithm for optimal sensor and actuator selection," in *53rd IEEE Conference on Decision and Control* (Los Angeles, CA: IEEE), 4039–4044. "

Donati, E., and Indiveri, G. (2023). Neuromorphic bioelectronic medicine for nervous system interfaces: from neural computational primitives to medical applications. *Progr. Biomed. Eng.* 2023, acb51c. doi: 10.1088/2516-1091/acb51c

Durán, J. M., and Jongsma, K. R. (2021). Who is afraid of black box algorithms? On the epistemological and ethical basis of trust in medical AI. *J. Med. Ethics* 47, 329–335. doi: 10.1136/medethics-2020-106820

Fernández, E., Alfaro, A., and González-López, P. (2020). Toward long-term communication with the brain in the blind by intracortical stimulation: challenges and future prospects. *Front. Neurosci.* 14, 681. doi: 10.3389/fnins.2020.00681

Fleming, J. E., Dunn, E., and Lowery, M. M. (2020a). Simulation of closed-loop deep brain stimulation control schemes for suppression of pathological beta oscillations in parkinson's disease. *Front. Neurosci.* 14, 166. doi: 10.3389/fnins.2020.00166

Fleming, J. E., Orłowski, J., Lowery, M. M., and Chaillet, A. (2020b). Self-tuning deep brain stimulation controller for suppression of beta oscillations: analytical derivation and numerical validation. *Front. Neurosci.* 14, 639. doi: 10.3389/fnins.2020.00639

Friedrich, J., Golkar, S., Farashahi, S., Genkin, A., Sengupta, A., and Chklovskii, D. (2021). "Neural optimal feedback control with local learning rules," in *Advances in Neural Information Processing Systems, Vol. 34*.

Gallego, J. A., Perich, M. G., Chowdhury, R. H., Solla, S. A., and Miller, L. E. (2020). Long-term stability of cortical population dynamics underlying consistent behavior. *Nat. Neurosci.* 23, 260–270. doi: 10.1038/s41593-019-0555-4

García, J., and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* 16, 1437–1480.

Gu, S., Pasqualetti, F., Cieslak, M., Telesford, Q. K., Yu, A. B., Kahn, A. E., etale (2015). Controllability of structural brain networks. *Nat. Commun.* 6, 8414. doi: 10.1038/ncomms9414

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., etale (2022). *A Review of Safe Reinforcement Learning: Methods, Theory and Applications. arxiv*.

Héliot, R., Ganguly, K., Jimenez, J., and Carmena, J. M. (2010). Learning in closed-loop brain-machine interfaces: modeling and experimental validation. *IEEE Trans. Syst. Man Cybern. B* 40, 1387–1397. doi: 10.1109/TSMCB.2009.2036931

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., etale (2018). Rainbow: Combining improvements in deep reinforcement learning. *Proc. AAAI Conf. Artif. Intell.* 32, 11796. doi: 10.1609/aaai.v32i1.11796

Himpe, C. (2022). *Emgr-Empirical Gramian Framework*.

Hosman, T., Vilela, M., Milstein, D., Kelemen, J. N., Brandman, D. M., Hochberg, L. R., etale (2019). "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)* (San Francisco, CA: IEEE), 1066–1071.

Ienca, M., and Haselager, P. (2016). Hacking the brain: brain–computer interfacing technology and the ethics of neurosecurity. *Ethics Inf. Technol.* 18, 117–129. doi: 10.1007/s10676-016-9398-9

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* 15, 1063–1070. doi: 10.1109/TNN.2004.832719

Jackson, A., Mavoori, J., and Fetz, E. E. (2006). Long-term motor cortex plasticity induced by an electronic neural implant. *Nature* 444, 56–60. doi: 10.1038/nature05226

Julier, S. J., and Uhlmann, J. K. (1997). "New extension of the kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, ed I. Kadar (SPIE), 1–12.

Khalil, H. K. (2002). *Nonlinear Systems*. Hoboken, NJ: Prentice Hall.

Kieboom, B., and Jafarian, M. (2022). *Utility of the Koopman Operator in Output Regulation of Disturbed Nonlinear Systems. arxiv*.

Küçükoğlu, B., Borkent, W., Rueckauer, B., Ahmad, N., Güçlü, U., and van Gerven, M. (2022a). *Efficient Deep Reinforcement Learning With Predictive Processing Proximal Policy Optimization." arxiv*.

Küçükoğlu, B., Rueckauer, B., Ahmad, N., van Steveninck,, J., d., R., Güçlü, U., and Van Gereven, M. (2022b). Optimization of neuroprosthetic vision via end-to-end deep reinforcement learning. *Int. J. Neural Syst*. 32, 2250052. doi: 10.1142/S0129065722500526

Kumar, G., Schieber, M. H., Thakor, N. V., and Kothare, M. V. (2013). "Designing closed-loop brain-machine interfaces using optimal receding horizon control," in *2013 American Control Conference* (Washington, DC), 5029–5034.

Lagang, M., and Srinivasan, L. (2013). Stochastic optimal control as a theory of brain-machine interface operation. *Neural Comput*. 25, 374–417. doi: 10.1162/NECO_a_00394

Lall, S., Marsden, J. E., and Glavaški, S. (1999). Empirical model reduction of controlled nonlinear systems. *IFAC Proc*. 32, 2598–2603. doi: 10.1016/S1474-6670(17)56442-3

Lewis, F. L., Vrabie, D., and Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: using natural decision methods to design optimal adaptive controllers. *IEEE Control Syst. Mag*. 32, 76–105. doi: 10.1109/MCS.2012.2214134

Li, C., Zhang, R., Wong, J., Gokmen, C., Srivastava, S., Martín-Martín, R., etale (2022). "BEHAVIOR-1K: a benchmark for embodied AI with 1,000 everyday activities and realistic simulation," in *6th Annual Conference on Robot Learning* (Auckland), 1–14. "

Linares-Barranco, A., Piñero-Fuentes, E., Canas-Moreno, S., Rios-Navarro, A., Maryada, W.u, C., Zhao, J., etale (2022). "Towards hardware Implementation of WTA for CPG-based control of a Spiking Robotic Arm," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)* (Austin, TX: IEEE), 1057–1061."

Lindsey, J., Ocko, S. A., Ganguli, S., and Deny, S. (2019). "A unified theory of early visual representations from retina to cortex through anatomically constrained deep CNNs," in *International Conference on Learning Representations (ICLR)* (New Orleans, LA), 1–17.

Manohar, K., Brunton, B. W., Kutz, J. N., and Brunton, S. L. (2018). Data-driven sparse sensor placement for reconstruction: demonstrating the benefits of exploiting known patterns. *IEEE Control Syst. Mag*. 38, 63–86. doi: 10.1109/MCS.2018.2810460

Meng, L., Gorbet, R., and Kulić, D. (2021). "Memory-based deep reinforcement learning for POMDPs," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Prague: IEEE).

Mitchell, B. A., and Petzold, L. R. (2018). Control of neural systems at multiple scales using model-free, deep reinforcement learning. *Sci. Rep*. 8, 10721. doi: 10.1038/s41598-018-29134-x

Münz, U., Pfister, M., and Wolfrum, P. (2014). Sensor and actuator placement for linear systems based on H_2 and H_∞ optimization. *IEEE Trans. Automat. Contr*. 59, 2984–2989. doi: 10.1109/TAC.2014.2351673

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: a review. *Neural Netw*. 113, 54–71. doi: 10.1016/j.neunet.2019.01.012

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., etale (2019). "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (Vancouver, BC: Curran Associates, Inc.), 8026–8037.

Pequito, S., Kar, S., and Aguiar, A. P. (2016). A framework for structural input/output and control configuration selection in large-scale systems. *IEEE Trans. Automat. Contr*. 61, 303–318. doi: 10.1109/TAC.2015.2437525

Pohlmeyer, E. A., Mahmoudi, B., Geng, S., Prins, N. W., and Sanchez, J. C. (2014). Using reinforcement learning to provide stable brain-machine interface control despite neural input reorganization. *PLoS ONE* 9, e87253. doi: 10.1371/journal.pone.0087253

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: reliable reinforcement learning implementations. *J. Mach. Learn. Res*. 22, 1–8.

Ritt, J. T., and Ching, S. (2015). "Neurocontrol: methods, models and technologies for manipulating dynamics in the brain," in *2015 American Control Conference (ACC)* (Chicago, IL), 3765–3780.

Roelfsema, P. R., Denys, D., and Klink, P. C. (2018). Mind reading and writing: the future of neurotechnology. *Trends Cogn. Sci*. 22, 598–610. doi: 10.1016/j.tics.2018.04.001

Sand, M., Durán, J. M., and Jongsma, K. R. (2022). Responsibility beyond design: physicians' requirements for ethical medical AI. *Bioethics* 36, 162–169. doi: 10.1111/bioe.12887

Sauer, T. D., and Schiff, S. J. (2009). Data assimilation for heterogeneous networks: the consensus set. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys*. 79(5 Pt 1), 051909. doi: 10.1103/PhysRevE.79.051909

Schiff, S. J. (2011). *Neural Control Engineering: The Emerging Intersection between Control Theory and Neuroscience*. Cambridge, MA: The MIT Press.

Schiff, S. J., and Sauer, T. D. (2008). Kalman filter control of a model of spatiotemporal cortical dynamics. *J. Neural Eng*. 5, 1–8. doi: 10.1088/1741-2560/5/1/001

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). *Proximal Policy Optimization Algorithms. arxiv*.

Shanechi, M. M., Orsborn, A. L., and Carmena, J. M. (2016). Robust brain-machine interface design using optimal feedback control modeling and adaptive point process filtering. *PLoS Comput. Biol*. 12, e1004730. doi: 10.1371/journal.pcbi.1004730

Simão, T. D., Jansen, N., and Spaan, M. T. J. (2021). "AlwaysSafe: reinforcement learning without safety constraint violations during training," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*.

Sorrell, E., Rule, M. E., and O'Leary, T. (2021). Brain–machine interfaces: closed-loop control in an adaptive system. *Ann. Rev. Control Robot. Auton. Syst*. 4, 167–189. doi: 10.1146/annurev-control-061720-012348

Sussillo, D. (2014). Neural circuits as computational dynamical systems. *Curr. Opin Neurobiol*. 25, 156–163. doi: 10.1016/j.conb.2014.01.008

Sussillo, D., Nuyujukian, P., Fan, J. M., Kao, J. C., Stavisky, S. D., Ryu, S., etale (2012). A recurrent neural network for closed-loop intracortical brain-machine interface decoders. *J. Neural Eng*. 9, 026027. doi: 10.1088/1741-2560/9/2/026027

Sussillo, D., Stavisky, S. D., Kao, J. C., Ryu, S. I., and Shenoy, K. V. (2016). Making brain-machine interfaces robust to future neural variability. *Nat. Commun*. 7, 13749. doi: 10.1038/ncomms13749

Sutton, R., and Barto, A. (2020). *Reinforcement Learning*. Cambridge, MA: MIT Press.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J. M., etale (2022). "Habitat 2.0: training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems* (New Orleans, LA), 1–16.

Todorov, E., Erez, T., and Tassa, Y. (2012). "MuJoCo: a physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Vilamoura-Algarve: IEEE), 5026–5033.

Traoré,, R., Caselles-Dupré,, H., Lesort, T., Sun, T., Cai, G., Díaz-Rodríguez,, N., etale (2019). "DisCoRL: continual reinforcement learning via policy distillation," in *Advances in Neural Information Processing Systems* (Vancouver), 1–14.

van de Ven, G. M., Siegelmann, H. T., and Tolias,. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nat. Commun*. 11, 4069. doi: 10.1038/s41467-020-17866-2

Wander, J. D., and Rao, R. P. (2014). Brain–computer interfaces: a powerful tool for scientific inquiry. *Curr. Opin. Neurobiol*. 25, 70–75. doi: 10.1016/j.conb.2013.11.013

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: a survey on few-shot learning. *ACM Comput. Surveys* 53, 1–34. doi: 10.1145/3386252

Wang, Z., Chen, C., and Dong, D. (2021). Lifelong incremental reinforcement learning with online bayesian inference. *IEEE Trans. Neural Netw. Learn. Syst*. 33, 4003–4016. doi: 10.1109/TNNLS.2021.3055499

Watkins, C. J. C. H., and Dayan, P. (1992). Technical note: q-learning. *Mach. Learn*. 8, 279–292. doi: 10.1007/BF00992698

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 1550–1560. doi: 10.1109/5.58337

Wilson, H. R., and Cowan, J. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophys. J*. 12, 1–24. doi: 10.1016/S0006-3495(72)86068-5

Wülfing, J. M., Kumar, S. S., Boedecker, J., Riedmiller, M., and Egert, U. (2019). Adaptive long-term control of biological neural networks with deep reinforcement learning. *Neurocomputing* 342, 66–74. doi: 10.1016/j.neucom.2018.10.084