Offline Policy Evaluation for Learning-based Deep Brain Stimulation Controllers

Qitong Gao
Electrical and Computer Engineering
Duke University
qitong.gao@duke.edu

Stephen L. Schmidt
Biomedical Engineering
Duke University
stephen.schmidt@duke.edu

Karthik Kamaravelu
Biomedical Engineering
Duke University
karthik.kamaravelu@duke.edu

Dennis A. Turner
Neurosurgery
Duke University
dennis.turner@duke.edu

Warren M. Grill
Biomedical Engineering
Duke University
warren.grill@duke.edu

Miroslav Pajic
Electrical and Computer Engineering
Duke University
miroslav.pajic@duke.edu

ABSTRACT

Deep brain stimulation (DBS) is an effective procedure to treat motor symptoms caused by nervous system disorders such as Parkinson's disease (PD). Although existing implantable DBS devices can suppress PD symptoms by delivering fixed periodic stimuli to the Basal Ganglia (BG) region of the brain, they are considered inefficient in terms of energy and could cause side-effects. Recently, reinforcement learning (RL)-based DBS controllers have been developed to achieve both stimulation efficacy and energy efficiency, by adapting stimulation parameters (*e.g.*, pattern and frequency of stimulation pulses) to the changes in neuronal activity. However, RL methods usually provide limited safety and performance guarantees, and directly deploying them on patients may be hindered due to clinical regulations. Thus, in this work, we introduce a model-based offline policy evaluation (OPE) methodology to estimate the performance of RL policies using historical data. As a first step, the BG region of the brain is modeled as a Markov decision process (MDP). Then, a deep latent MDP (DL-MDP) model is learned using variational inference and previously collected control trajectories. The performance of RL controllers is then evaluated on the DL-MDP models instead of patients directly, ensuring safety of the evaluation process. Further, we show that our method can be integrated into offline RL frameworks, improving control performance when limited training data are available. We illustrate the use of our methodology on a computational Basal Ganglia model (BGM); we show that it accurately estimates the expected returns of controllers trained following state-of-the-art RL frameworks, outperforming existing OPE methods designed for general applications.

KEYWORDS

Deep Brain Stimulation, Reinforcement Learning, Variational Inference

1 INTRODUCTION

Millions of individuals in the US are affected by nervous system disorders, such as Parkinson's disease (PD) [34] and epilepsy [14]. Deep brain stimulation (DBS) is effective in treating such disorders by delivering electric pulses to the basal ganglia (BG) region of the brain through an implantable device [4, 12, 13, 37], as illustrated in Fig. 1. Existing commercial DBS devices can be programmed to provide stimuli with fixed parameters (*e.g.*, pulse frequency and amplitude) and switch on/off following pre-defined protocols – *i.e.,* switching on predefined thresholds for certain physiological biomarkers.[1] The programming of the implantable pulse generator is patient-specific and the detailed configurations are obtained by leveraging the physician's domain expertise, as well as trial-and-error for fine-tuning [39]. However, the configuration process (sometimes referred as *device programming*) is time-consuming, and stimulating with constant high frequency and amplitude significantly shortens the battery life of the implantable device and can result in serious side effects, such as induced dyskinesia [6].

Consequently, there has been significant recent interest in closed-loop DBS; the main focus has been on developing adaptive DBS (aDBS) methods that can turn on/off stimulation or adjust the intensity using very simple adaptation methods (e.g., ramp-based increase) when specific biomarker signals cross predefined thresholds [2, 3, 6, 30, 31]. Specifically, various neurosignals such as BG local field potentials (LFPs) and internal electroencephalography (iEEG), as well as measurements from external wearable devices (*e.g.*, accelerometer readings and electromyography) have been used as feedback signals, with the thresholds manually obtained by physicians looking over data collected from trials. Although still in early development, such approaches have shown potential to reduce energy consumption and side effects of stimulation [19, 31]; however, currently aDBS still requires substantial efforts to configure the devices such that desirable *stimulation efficacy* and *energy efficiency* are jointly attained.

Reinforcement learning (RL) has demonstrated its strengths in solving sophisticated control problems from various cyber-physical system (CPS) domains including robotics, smart transportation, etc [7, 9, 17, 35, 41]. Several recent works leverage RL to derive *closed-loop* controllers for DBS [16, 18, 36, 39]. Specifically, [18, 36, 39] propose the use of EEG and LFP signals to define the state space of the RL environment, followed by temporal difference or fitted Q-iteration algorithms to learn RL control policies that can select appropriate stimulation frequencies to reduce energy consumption. Although these methods can improve energy efficiency, the resulting controllers are *patient-agnostic* since periodic stimuli are still used across different patients. On the other hand, in [16], deep

This work is supported in part by NIH R37 NS040894, NIH UH3 NS103468, AFOSR award FA9550-19-1-0169, as well as the NSF CNS-1652544, CNS-1837499 and CNS-2112562 awards.

[1]The use of ON/OFF switching for DBS devices has not been FDA approved; currently, only research devices, with specific research study protocols, can do this.

actor-critic RL is used to design optimal stimuli patterns specific to each patient. The approach not only enables the controller to jointly achieve better stimulation efficacy and energy efficiency, but also to adapt to neurological changes over time (e.g., changes in the severity of PD symptoms, taking medications such as levodopa).

However, evaluating the performance of RL controllers for physiological control implemented by medical devices (either the final or a control policy active in a specific learning iteration) remains a significant challenge. Unlike testing in benchmark environments (e.g., [8], robotics or video games simulators), where a trained RL controller can be directly deployed (and potentially subsequently updated), the procedures of evaluating new controllers on patients are highly scrutinized, where the controller's effectiveness and safety need to be demonstrated even before experiments start [38]. Consequently, it is imperative to develop frameworks that can evaluate the performance of RL controllers in an *offline* manner (*i.e.*, without direct testing on patients).

In this paper, we introduce a *model-based* offline policy evaluation (OPE) methodology to accurately estimate the performance of RL-based DBS controllers without the need of deploying them *in vivo*, which ensures safety of the evaluation process. More importantly, we also demonstrate that the OPE module can be integrated into offline RL training frameworks, resulting in better-performing DBS policies even with limited training data. Our methodology starts by modeling the neuronal activity in the BG region as a Markov decision process (MDP). Then, we design a probabilistic machine learning model using variational inference [25], which we refer to as the *deep latent Markov decision process* (DL-MDP); using historical interactions, the model enables capturing the activity of BG neurons, as well as the transitions in the MDP, in response to DBS stimuli. Finally, the DL-MDP is used to interact with the RL controllers that are to be evaluated, and we show how the expected return (i.e., control performance) of each controller can be extrapolated from the resulting trajectories.

The problem of evaluating RL controllers offline has attracted significant attention in general, and some existing OPE methods [10, 15, 21, 32, 40, 47, 48, 50, 51] could be adapted to the DBS scenarios considered in this work. However, these methods mostly adopt the idea of importance sampling (IS) where the importance weights are shown to result in high variance in estimating the expected returns of RL policies, especially when the system has a long horizon [10, 32], as is the case for DBS. For example, in [40], a self-normalized step-wise IS method is developed to reduce the scale of importance weights, which is capable of decreasing the variance. In addition, [21, 48] propose the use of additional value estimators for variance reduction. On the other hand, [10, 32, 50, 51] introduce a distribution correction estimation family of OPE methods, which estimate the correction ratio of the stationary distribution used for generating importance weights; thus, the results are expected to be associated with lower variance. However, such methods are shown to introduce bias in estimations while compensating for high variance [47]. As a result, these methods may not be suitable for the medical scenarios considered in this work, as it is crucial to devise an accurate OPE method for DBS controllers.

To evaluate our methodology for DBS controllers, we adopt a commonly used computational Basal Ganglia Model (BGM) [44] as the testbed; the BGM facilitates the use of various DBS research

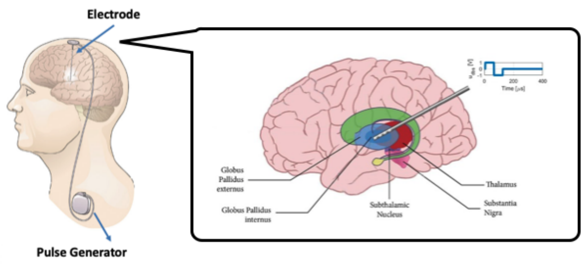

Figure 1: Deep brain stimulation (DBS): the implantable pulse generator is placed in the patient's chest, and multi-contact electrodes that can record local field potentials (LFPs) and deliver stimulation are implanted in the basal ganglia (BG).

platforms [16, 22]. Specifically, we first train two types of controllers to change stimulation pulse patterns and frequencies, respectively, following state-of-the-art RL frameworks (*i.e.,* deep actor-critic [16, 29, 43] and deep Q-learning [35]). Then, the introduced OPE method is used to estimate the expected return resulting from the use of these RL controllers, with the errors being evaluated using the mean squared errors (MSEs) and rank correlations. Compared to the existing IS based OPE baselines [32, 40], our approach results in estimations with low variances, achieving consistently better performance in terms of both metrics. In addition, we show that the OPE module can be integrated into *offline* RL frameworks, leading to improved performance even when limited training data is available, which is critical for medical scenarios (as described in Section 2.1.2).

The contributions of this work are three-fold: (*i*) to the best of our knowledge, this is the first method to evaluate RL-based DBS controllers in an *offline* manner, reducing the level of direct interactions needed between the patient and controllers during the evaluation and learning; (*ii*) our OPE approach is shown to be more *effective* compared to existing OPE methods designed for general applications; and (*iii*) we show that the OPE method can be easily integrated into offline RL training frameworks, improving control *efficacy* even with limited training data.

This paper is organized as follows. Section 2 briefly overviews DBS and the computational BGM, as well as motivates the need for OPE of DBS controllers, which is considered in this work. Section 3 describes the RL frameworks to design DBS control policies that can adapt stimulation (i) patterns, or (ii) frequencies. In Section 4, the OPE approach and the integration of OPE into RL training are introduced. Our methodhodology is evaluated in Section 5, before discussion and avenues for future work are presented in Section 6.

2 PRELIMINARIES AND MOTIVATIONS

In this section, we briefly introduce the preliminaries of DBS and the computational Basal Ganglia model (BGM). We refer readers to [16, 22, 29, 43, 44] for in-depth reviews.

2.1 Deep Brain Stimulation: The Need for OPE

2.1.1 Parkinson's Disease and Deep-Brain Stimulation. PD originates from degenerative changes in the BG region and can cause various motor symptoms including bradykinesia and rigidity [11, 26]. Such symptoms can be captured by the changes in local field potentials (LFPs), or electrical potentials, from the *globus pallidus pars*

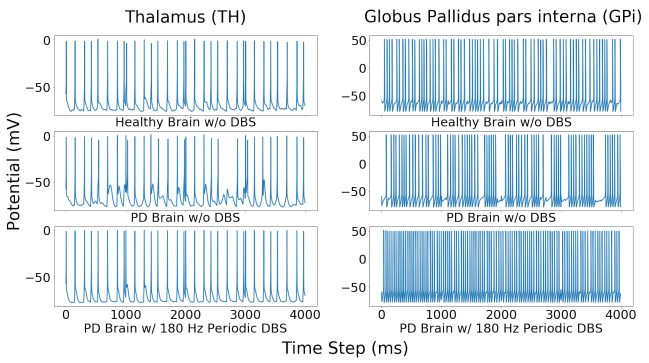

Figure 2: Activity of model neurons in the thalamus (TH) and globus pallidus pars interna (GPi). Substantial pathophysiological patterns can be found in PD brain without DBS stimulation (middle row). Such effects are reduced significantly using periodic stimulation at 180 Hz (bottom row).

interna (GPi) and *thalamus* (TH) sub-regions in the BG. DBS devices can record these potentials through electrodes implanted into the BG, as shown in Fig. 1. In healthy GPi and TH, the neurons follow sporadic spiking (*i.e.*, neuron activations) at a stable firing rate. However, when affected by PD, pathological neuron activations can be found in TH and GPi, captured by reduced triggering potentials and clustered spiking, respectively (see Fig. 2).

Two quality of control (QoC) metrics can be used to evaluate such abnormal neuronal activities quantitatively (*e.g.*, [16, 22, 44]). Error Index (EI) captures the ratio of erroneous firings in TH, while the clustering of spikes in GPi causes increased spectral density in the *beta band* (*i.e.*, frequencies in the $[13, 35] Hz$ range). Both metrics are defined in Section 2.2 and Appendix B.

Implantable DBS devices can also deliver stimulation pulses to GPi or *subthalamic nucleus* (STN) part of the BG, to suppress PD symptoms. Specifically, the device can continuously generate trains of short voltage pulses at a high frequency (about 180 Hz), which activate the BG around the electrode [28]. As shown in Fig. 2 (bottom row), erroneous TH activations are corrected by the stimuli. However, it is worth noting that the spiking frequency in GPi increased after DBS, compared to healthy brains. This could potentially lead to side-effects such as speech impairment and facial contraction, especially when the stimuli are too strong [5]. Furthermore, constantly stimulating with high frequencies significantly reduces the battery lifetime of the DBS device. Consequently, it is important to design DBS controllers that are not only *effective* but also *energy-efficient*.

2.1.2 The Necessity of OPE for Learning-based DBS Control. As discussed in Introduction, several RL-based controllers have shown promising results in balancing control efficacy and stimulation efficiency by enabling adaption to changes of BG neuronal activities *on-the-fly*. For example, [16] adopts deep actor-critic RL methods to generate control policies that can adjust the time-intervals between two consecutive stimulation spikes (*i.e.*, pulse *patterns*). However, substantial training data, obtained from extensive interactions with the plant/environment, is usually necessary to learn suitable controllers [23], especially for control of complex physiological processes. This, on the other hand, may be intractable in real world.

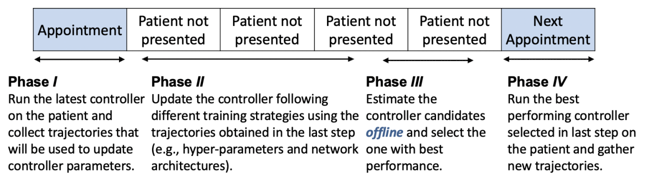
Appointment
Patient not presented
Patient not presented
Patient not presented
Patient not presented
Next Appointment
Phase I
Run the latest controller on the patient and collect trajectories that will be used to update controller parameters.
Phase II
Update the controller following different training strategies using the trajectories obtained in the last step (e.g., hyper-parameters and network architectures).
Phase III
Estimate the controller candidates offline and select the one with best performance.
Phase IV
Run the best performing controller selected in last step on the patient and gather new trajectories.

Figure 3: Typical timeline for training RL-based DBS controllers in clinical studies. During trials, only limited data can be collected for updating and evaluating the controllers, since patients are only occasionally available and a physician's approval is required before using each new controller.

During clinical trials/studies, patients may participate in studies sparsely over time (*e.g.*, once in a month). Moreover, before each trial starts, any controller needs to be assessed by a physician [38], which further limits the amount of time left for performance evaluation and data collection for future training/improvements.

A typical timeline to train learning-based DBS controllers is summarized in Fig. 3. During each trial, one is expected to focus on evaluating and collecting new data resulted from the use of (currently) top-performing controllers (Phase *I*). In what follows, the control policies are updated following suitable training procedures (*e.g.*, using different sets of hyper-parameters) to improve the likelihood of producing a better candidate to be deployed at the next trial (Phase *II*). Then, a few controllers, which can potentially result in better performance, need to be determined (Phase *III*). Otherwise, all of them would have to be evaluated at the next available trial (Phase *IV*). Although techniques such as *offline* RL [29, 35] can facilitate Phase *II*, there exist a critical need to fill in the blank of Phase *III*.

OPE refers to methods that can approximate the performance of RL-based controllers using historical data, *i.e., without* requiring the patients to be presented while evaluating RL policies, which aligns with the objectives and constraints of Phase *III*. However, most existing OPE methods, such as [10, 15, 21, 32, 40, 47, 48, 50, 51], are heavily based on importance sampling (IS) and could result in inconsistent estimations due to the high variance of the IS weights [10, 32]. In contrast, in this work, we introduce a model-based OPE that can robustly and accurately learn a belief space using variational inference [25], in order to capture and reconstruct neuronal activities in the BG. Consequently, the learned model can be used to interact with each controller candidate extensively; thus, can evaluate their performance thoroughly. In addition, as we introduce in Sec. 4.2, the OPE can also be integrated into offline RL frameworks to improve efficacy of the learned policies, even with limited training data.

2.2 Computational Basal Ganglia Model

We exploit the BGM introduced in [44], which was also adopted in [16, 22]. The BGM models four important sub-regions in the BG region of the brain, from which the effects of PD can be quantitatively captured; specifically, they are GPi, TH, STN and *globus pallidus pars externa* (GPe), and the connectivity among the regions is illustrated in Fig. 4. The system (i.e., neuronal) dynamics as well as information transmitted among neurons can be captured by electrical potentials of the neurons. Due to space constraints, details about the BGM are provided in Appendix B.

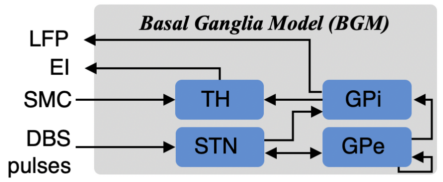

Figure 4: An overview of the computational BGM. The DBS stimulation is applied to the STN, after which it is propagated to the other sub-regions. Sensorimotor cortex (SMC) inputs are also considered, which send activations to thalamus (TH) and are essential for computing the error index (EI).

Two QoC metrics are used to quantify the severity of PD symptoms (see *e.g.,* [16, 22, 44] and references within), EI and beta power spectral density (P_β). Moreover, they can also be used to evaluate the efficacy of DBS, by capturing the changes in EI and P_β before and after when the stimulation is applied.

EI is defined as the portion of erroneous TH neuron activations in response to sensorimotor cortex (SMC) inputs.[2] Empirically, from Fig. 2 it can be observed that substantial erroneous TH neuron activations exist in the PD brain without DBS, resulting in higher EI as shown in Fig. 5. Furthermore, such effects can be mitigated by using periodic DBS at 180 Hz (or as we show in Section 3, by *e.g.,* the RL-based stimulation pattern controller with an average 45 Hz pulse frequency).

The other metric, P_β, measures the power spectral density of GPi neuron potentials within the beta band. It can distinguish between the oscillations of GPi neuronal activities exhibited in healthy and PD brains. As shown in Fig. 6, P_β is exaggerated in the case of PD; *i.e.,* P_β of the PD brain without DBS is significantly higher than for a healthy brain. Similar to EI, P_β can be reduced using periodic DBS at 180 Hz, or as we will show, RL-based controllers. Note that, as discussed in Section 2.1.1 and Fig. 2, 180 Hz periodic DBS pulses may reduce P_β to below the level present in healthy brains, due to the spike overflow.

3 RL FOR DBS CONTROLLER DESIGN

We start by adopting the approach proposed in [16] to design RL control policies that change stimulation *patterns* for the BGM (Section 3.1). Furthermore, we extend this approach to allow for the RL-based design of controllers that adapt stimulation *frequencies* (Section 3.2). The performance of both types of controllers will be evaluated by the OPE method introduced in Section 4.

3.1 DBS Pattern Control

To design RL policies that automatically adapt DBS stimuli patterns to the changes in neuronal activities and PD symptoms, an MDP model needs to be formulated as the *environment* upon which the RL agents can be trained. Specifically, the MDP should characterize the neuron activities in the BGM as well as their responses to DBS stimulation. It is usually formulated as a 6-tuple $\mathcal{M} = (\mathcal{S}, s_0, \mathcal{U}, \mathcal{P}, R, \gamma)$, where each of its elements is introduced as follows.

[2]Healthy brains could also respond to SMC_τ erroneously with a low probability ($< 0.1\%$).

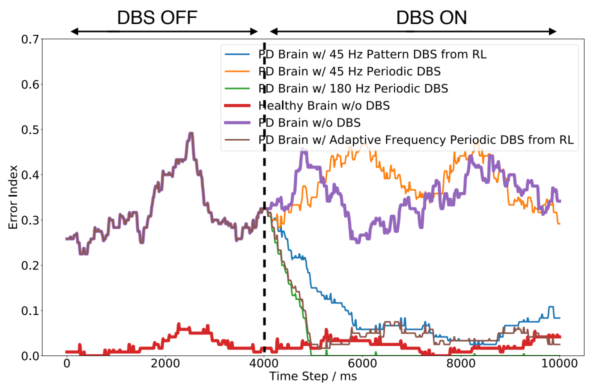

Figure 5: Error Index (EI) over time in model PD brains without and with various types of stimulation, as well as model healthy brains. The RL-based pattern controller with the average 45 Hz stimulation, RL controller that adapts frequency of the stimulation, and (standard) periodic stimulation at 180 Hz all reduce the EI in PD brains to the levels as in healthy (*i.e.,* non-PD) brains.

State Space $\mathcal{S}$. The states of MDP should capture the status of neurons the BG region, characterized by the BGM. Specifically, the state at a *discrete* time step t, s_t, can be defined as a sequence of EI and P_β sampled at a fixed rate, $m \in \mathbb{Z}^+$, over a window of size T_w, *i.e.,*

$$s_t = \begin{bmatrix} e_{(t)}, e_{(t+m)}, e_{(t+2m)}, \dots, e_{(t+T_w-m)} \\ \beta_{(t)}, \beta_{(t+m)}, \beta_{(t+2m)}, \dots, \beta_{(t+T_w-m)} \end{bmatrix}; \tag{1}$$

here, the $e_{(\cdot)}$'s and $\beta_{(\cdot)}$'s represent the EI and P_β evaluated at $l = T_w/m$ number of equally-spaced intervals within the window, respectively, $T_w \in \mathbb{Z}$, $l \in \mathbb{Z}^+$ and $s_t \in \mathbb{R}^{2\times l}$.[3] The initial state s_0 is determined following the initialization of the electrical potentials vector of the BGM (i.e., the vector $\mathbf{v}$ introduced in (29) in Appendix B) which, in addition to the SMC response, is the source of stochasticity in the BGM [22, 44].

Action Space $\mathcal{U}$. We consider changing the stimulation pattern every T_w steps, so the actions the RL agent can take at time t are

$$u_t = [u_{(t)}, u_{(t+m)}, u_{(t+2m)}, \dots, u_{(t+T_w-m)}], \tag{2}$$

where $u_t \in \{0, 1\}^l$, and $u_{(t+n\cdot m)} = 1$ (or 0) means a stimulation pulse is triggered (or not) at step $t + n \cdot m$ for all $n \in [0, l-1) \subset \mathbb{Z}$.

Transition Dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$. The RL agent interacts with the MDP environment following the mechanism such that, every time after s_t is sampled, the agent applies DBS stimulation following pattern u_t (*i.e.,* the action) and the environment responds with the EI and P_β readings over the next window as

$$s_{t+T_w} = \begin{bmatrix} e_{(t+T_w)}, e_{(t+T_w+m)}, e_{(t+T_w+2m)}, \dots, e_{(t+2T_w-m)} \\ \beta_{(t+T_w)}, \beta_{(t+T_w+m)}, \beta_{(t+T_w+2m)}, \dots, \beta_{(t+2T_w-m)} \end{bmatrix}. \tag{3}$$

The interactions between the RL agent and the environment over a finite horizon $T \cdot T_w$ ($T \in \mathbb{Z}^+$ and $T < \infty$) can be summarized as follows. After initialization, the environment provides s_0 and the agent chooses u_0. Then, the environment responds with $s_{j \cdot T_w}$, after

[3]Without loss of generality, here we assume that T_w is a multiple of m.

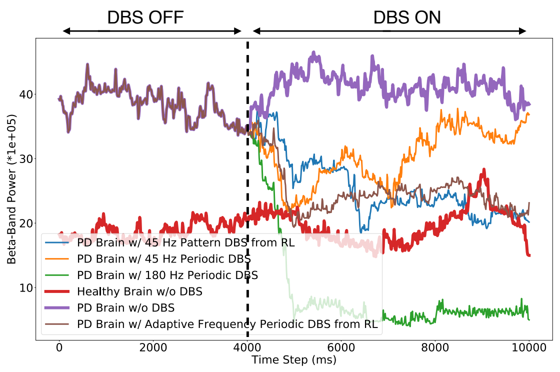

Figure 6: Beta power spectral density (P_β) over time in model PD brains without and with various types of stimulation, as well as in healthy brains. The RL pattern controller with the average 45 Hz stimuli, RL controller that adapts stimulation frequency, and (standard) periodic stimuli at 180 Hz all reduce P_β to the levels as in model healthy brains.

which the action $u_{j \cdot T_w}$ is taken immediately, for all $j \in [1, T] \subset \mathbb{Z}$. Since both the states and actions are updated every T_w steps, we can simplify notations $s_{j \cdot T_w}$ and $u_{j \cdot T_w}$, to s_j and u_j, respectively, in the rest of this paper.[4] Furthermore, as in [16], we consider each u_j to contain the same number of pulses equivalent to stimulating at a *fixed* frequency less than the maximally allowed rate, $f < f_{max}$, where in clinical practice $f_{max} \approx 180$ *Hz* (*e.g.*, [12, 33]).

Reward Function $R : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to \mathbb{R}$. An immediate reward $r_j = R(s_j, u_j, s_{j+1})$ is received by the RL agent after taking action u_j at s_j and observing the new state s_{j+1}. The reward function, R, can be designed to promote the proper actions taken at each state, as well as to penalize the ones that can cause undesirable consequences. Specifically, we consider the reward function

$$R(s_j, u_j, s_{j+1}) = \begin{cases} r_a, & \text{if } \bar{e}_{j+1} < \xi_e \text{ and } \bar{\beta}_{j+1} < \xi_\beta; \\ r_b, & \text{if } \bar{e}_{j+1} \geq \xi_e \text{ and } \bar{\beta}_{j+1} < \xi_\beta; \\ r_b, & \text{if } \bar{e}_{j+1} < \xi_e \text{ and } \bar{\beta}_{j+1} \geq \xi_\beta; \\ r_c, & \text{if } \bar{e}_{j+1} \geq \xi_e \text{ and } \bar{\beta}_{j+1} \geq \xi_\beta; \end{cases} \tag{4}$$

here, $\bar{e}_{j+1} = \frac{1}{l} \sum_{k=0}^{l-1} e_{((j+1)T_w + km)}$ and $\bar{\beta}_{j+1} = \frac{1}{l} \sum_{k=0}^{l-1} \beta_{((j+1)T_w + km)}$ are the average EI and P_β over the window T_w captured by the state s_{j+1}; $\xi_e, \xi_\beta \in \mathbb{R}$ are the thresholds; and $r_a >> r_b > 0 > r_c$. In other words, a large positive reward r_a is given if the averages of both EI and P_β are below the thresholds, ξ_e and ξ_β respectively, after stimulating following the pattern in u_j. Similarly, a smaller positive reward r_b, or a negative reward r_c, is issued by the environment if the action u_j only results in one of the QoC metric to go below the threshold, or both metrics to go above the thresholds, respectively.

Discounting Factor γ. Finally, we also consider a discounting factor $\gamma \in [0, 1) \subset \mathbb{R}$, which is a constant essential for deriving the learning objectives in below.

Before introducing the RL objective, we first define the control policy π and accumulated return G_j of an MDP.

[4]Here we slightly abuse notation j by considering $j \in [0, T] \subset \mathbb{Z}$.

Definition 3.1 (Control Policy of an MDP). *A policy π of an MDP $\mathcal{M}$ is a function, $\pi : \mathcal{S} \to \mathcal{U}$ that maps the set of states $\mathcal{S}$ to the set of (control) actions $\mathcal{U}$.*

Definition 3.2 (Accumulated Return). *Given an MDP $\mathcal{M}$ and a policy π, the accumulated return over a finite horizon starting from the stage j and ending at stage T, for $T > j$, is defined as*

$$G_j = \sum_{k=0}^{T-j} \gamma^{j+k} r_{j+k}, \tag{5}$$

where r_{j+k} is the return at the stage $j + k$.

Now, given an MDP with its dynamics $\mathcal{P}$ remaining ***unknown*** along with a pre-defined reward function R, the goal of RL is to find a policy π that maximizes the *expected return*

$$J(\pi) = \mathbb{E}_{s,u \sim \rho^\pi, r \sim R}[G_0], \tag{6}$$

where $\rho^\pi = \{(s_0, u_0), (s_1, u_1), \dots | u_j = \pi(s_j)\}$ is the sequence of states and actions drawn from the trajectory distribution determined by π. As a result, the *optimal* policy π^* can be obtained as

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi). \tag{7}$$

In [16], a deep actor-critic RL framework [29] is adapted to design policies that map states to corresponding stimulation patterns. It is shown effective in handling the large state and action space of the environment, significantly reducing EI and P_β over a finite horizon. As we adapt this method, here, we briefly overview it while we refer readers to [16] for details. We start with defining the state-action value functions, or the Q-value functions.

Definition 3.3 (State-Action Value Function). *Given an MDP $\mathcal{M}$ and policy π, the state-action value function $Q^\pi(s, u)$, where $s \in \mathcal{S}$ and $u \in \mathcal{U}$, is defined as the expected return for taking action u when at state s following policy π, i.e.,*

$$Q^\pi(s, u) = \mathbb{E}_{s,u \sim \rho^\pi, r \sim R}[G_0 | s, u]. \tag{8}$$

The method from [16] utilizes an *actor* $\pi_{\theta_u}(s) : \mathcal{S} \to \mathcal{U}$ and a *critic* $Q_{\theta_c}(s, u) : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$, parameterized by neural networks (NNs) with weights θ_u and θ_c, to approximate the optimal policy π^* and Q-values $Q^{\pi^*}(\cdot, \cdot)$, respectively. Unlike [29], the NN architectures with weights θ_c and θ_u are specifically designed toward deriving stimulation pulse *patterns* in DBS. To obtain the optimal policy in (7), the objective (6) can be re-formulated *w.r.t.* the state-action value functions as

$$J_\beta(\pi_{\theta_u}) = \mathbb{E}_{s \sim \rho^\beta} \left[Q_{\theta_c} \left(s, \pi_{\theta_u}(s) \right) \right]; \tag{9}$$

here, $\beta : \mathcal{S} \to \mathcal{U}$ is the *exploration policy* that is usually obtained by introducing disturbance to the actor π_{θ_u} to ensure sufficient exploration of the environment, and $\rho^\beta = \{(s_0, u_0), (s_1, u_1), \dots | u_j = \beta(s_j)\}$ is the state-action visitation distribution obtained over β. Then, the parameters of the actor, θ_u, can be updated iteratively following gradient ascent, *i.e.*, for the learning rate α_u

$$\theta_u' \leftarrow \theta_u + \alpha_u \nabla_{\theta_u} J_\beta(\pi_{\theta_u}). \tag{10}$$

To update the critic, θ_c is set to minimize

$$J_\beta(\theta_c) = \mathbb{E}_{s_j, u_j, s_{j+1} \sim \rho^\beta, r \sim R} \left[\left(r + \gamma Q_{\theta_c}(s_{j+1}, \pi_{\theta_u}(s_{j+1})) - Q_{\theta_c}(s_j, u_j) \right)^2 \right],$$

which can be obtained using gradient descent, *i.e.*,

$$\theta_c' \leftarrow \theta_c - \alpha_c \nabla_{\theta_c} J_\beta(\theta_c). \tag{11}$$

Using this RL-based approach, we obtained a control policy that can adjust DBS pulse patterns, whose average stimulation frequency is equivalent to 45 Hz in the periodic cases (*i.e.*, they stimulate the same amount of pulses within each T_w). On the other hand, as shown in Fig. 7 (top row) this controller effectively corrects majority of erroneous firings in both TH and GPi. Furthermore, it reduces EI and P_β to the level as in healthy brains, as shown in Fig. 5 and 6.

3.2 DBS Frequency Control

We also investigate the use of RL to design controllers that adapt the frequency of periodic DBS stimulation pulses, since frequency adaptation is commonly considered to have significant potential for aDBS [36, 39, 45, 49, 52]. Specifically, the policy is expected to choose a stimulation frequency from a *discrete* set of integers between 0-180, as 180 Hz is usually used in open-loop DBS devices and can suppress most PD symptoms indiscriminately to different levels of neuronal activities in the BG region [12, 33]. Therefore, we define the action space of the MDP in this setting as 13 equally-spaced integers sliced from $[0, 180] \subset \mathbb{Z}$, *i.e.*,

$$u_j \in \{0, 15, 30, \cdots, 165, 180\}, \tag{12}$$

while the state space and transition dynamics remain the same as defined in Section 3.1. We also modify the reward function to account for energy consumption, such as introducing penalties while stimulating with large frequencies, *i.e.*,

$$R(s_j,u_j,s_{j+1})=\begin{cases} r_a - C\cdot u_j, & \text{if } \bar{e}_{j+1} < \xi_e \text{ and } \bar{\beta}_{j+1} < \xi_\beta; \\ r_b - C\cdot u_j, & \text{if } \bar{e}_{j+1} \geq \xi_e \text{ and } \bar{\beta}_{j+1} < \xi_\beta; \\ r_b - C\cdot u_j, & \text{if } \bar{e}_{j+1} < \xi_e \text{ and } \bar{\beta}_{j+1} \geq \xi_\beta; \\ r_c - C\cdot u_j, & \text{if } \bar{e}_{j+1} \geq \xi_e \text{ and } \bar{\beta}_{j+1} \geq \xi_\beta; \end{cases} \tag{13}$$

here, $C \in \mathbb{R}$ is a constant balancing the scales between r's and u_j.[5]

Since the action space of this problem is sparse and dramatically smaller than for the MDP for pattern adaptation from Section 3.1, deep Q-networks (DQNs) [35] can be used to derive control policies as they are sufficient for such (i.e., smaller) *discrete* action spaces. In this case, only a critic $Q_{\theta_q}(s,u)$, parameterized by an NN with weights θ_q, needs to be trained to minimize

$$J_\beta(\theta_q)=\mathbb{E}_{s_j,u_j,s_{j+1}\sim\rho^\beta,r\sim R}\Big[\Big(r+\gamma Q_{\theta_q}(s_{j+1},\pi_{\theta_q}(s_{j+1}))-Q_{\theta_q}(s_j,u_j)\Big)^2\Big];$$

the resulting policy defined over the critic is then

$$\pi_{\theta_q}(s_j) = \operatorname*{argmax}_u Q_{\theta_q}(s_j,u). \tag{14}$$

Consequently, θ_q can be updated following gradient descent, *i.e.*,

$$\theta'_q \leftarrow \theta_q - \alpha_q \nabla_{\theta_q} J_\beta(\theta_q). \tag{15}$$

Fig. 7 (bottom row) shows scenario when the obtained RL-based controller chooses to stimulate at 75 Hz in a 4-second period, which corrects most of the pathological activations in the two sub-regions. As a result, it effectively reduces EI and P_β (Figs. 5 and 6). In general, the RL controller capable of pattern adaptation outperforms the one with periodic stimuli where only frequency can be adapted; it achieves acceptable EI and P_β levels for lower average pulse frequency as it can both change the average stimulation frequency and timing. However, unlike the frequency controller, it is currently not supported by existing FDA-approved DBS device hardware.

[5]Note that in (4) we do not need the extra term since the number of pulses (*i.e.*, energy consumption) across all u_j's remains the same.

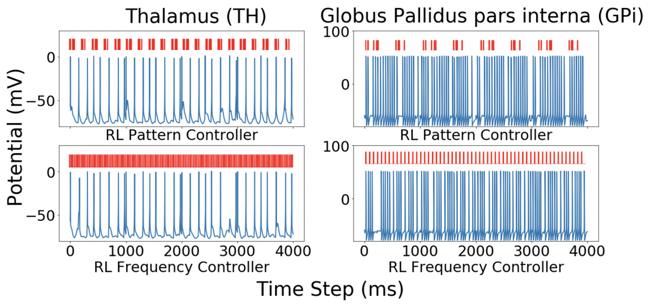

Figure 7: Activity of model neurons in TH and GPi after stimulating with RL-based pattern (top) and frequency (bottom) controllers for 4 s. The stimulation pulses are shown at the top of each sub-plot (in red); GPi neural activity is displayed at a reduced rate as it is originally too dense to be visualized.

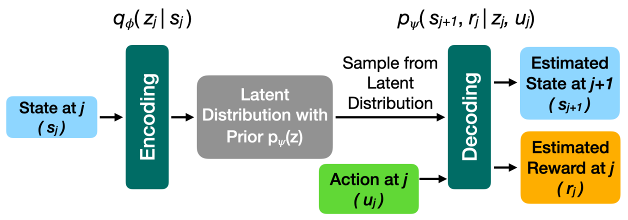
$q_\phi(z_j \mid s_j)$
$p_\psi(s_{j+1}, r_j \mid z_j, u_j)$
State at j (s_j)
Encoding
Latent Distribution with Prior $p_\psi(z)$
Sample from Latent Distribution
Decoding
Action at j (u_j)
Estimated State at $j+1$ (s_{j+1})
Estimated Reward at j (r_j)

Figure 8: The architecture of the DL-MDP, capturing the prior $p_\psi(z_j)$, encoder $q_\phi(z_j|s_j)$ and decoder $p_\psi(s_{j+1}, r_j|z_j, u_j)$.

4 MODEL-BASED OFFLINE POLICY EVALUATION

In this section, we introduce a model-based approach to solve the OPE problem. We then show how to integrate the OPE method into *offline* RL training frameworks, which allows the agent to update and evaluate the policies more efficiently during training.

In Section 3, we showed how to learn DBS controllers (*i.e.*, control policies) that can adjust stimulation artifacts (*i.e.*, patterns or frequencies) responding to the changes in neuronal activities. However, as previously discussed, it is important to demonstrate the efficacy of the RL controllers before patient trials start. We formally define the OPE as follows.

Problem 1 (Offline Policy Evaluation). *Consider a target policy π, and off-policy trajectories $\rho^\mu = \{(s_0,u_0),(s_1,u_1),\ldots|u_j = \mu(s_j)\}$, collected following a behavioral policy $\mu \neq \pi$, over an MDP $\mathcal{M}$ capturing the BG neuronal activities. The goal of the OPE is to estimate the expected return of the target policy π, i.e., $\mathbb{E}_{s,u\sim\rho^\pi,r\sim R}[G_0]$.*

4.1 Deep Latent MDP Model for OPE

We introduce a model-based approach for OPE of DBS controllers. Specifically, a deep latent MDP (DL-MDP) model is learned using variational inference [25], capturing the transition dynamics and rewards of the MDP using trajectories obtained following a behavioral policy μ, *i.e.*, $\rho^\mu = \{(s_0,u_0),(s_1,u_1),\ldots|u_j = \mu(s_j)\}$.

The formulation of DL-MDPs follows from the general Bayesian inference frameworks [25] and they consist of three components, *i.e.*, priors, posteriors and sampling distributions. Specifically, a DL-MDP includes: (*i*) a set of priors $p_\psi(z_j)$, parameterized by ψ, over the *latent variable space* (LVS) $\mathcal{Z} \subset \mathbb{R}^d$, where $d \in \mathbb{Z}^+$ is a hyper-parameter; the priors are usually represented by the parametric family of distributions such as multivariate Gaussian and represent one's belief over the latent distribution of the states, $\mathcal{Z}$, before sampling, where $\mathcal{Z}$ can be seen as the feature space characterizing high-level representations over the state space $\mathcal{S}$; (*ii*) the encoder (or approximated posterior) $q_\phi(z_j|s_j)$, parameterized by ϕ, which maps the MDP state $s_j \in \mathcal{S}$, obtained at stage j, to the latent variable $z_j \in \mathcal{Z}$; note that the true posterior $p_\psi(z_j|s_j)$ cannot be obtained due to the intractable marginal distribution (see (19)); however, the variational inference framework allows it to be approximated using q_ϕ (see Theorem 4.1 as well as [25] for more details); and (*iii*) a decoder (or sampling distribution) $p_\psi(s_{j+1}, r_j|\ z_j, u_j)$, which enforces the transition from stage j to $j+1$ in the corresponding MDP and reconstructs the next state s_{j+1} and reward r_j conditioned on the latent variable z_j and action u_j.

As a result, the DL-MDP is used to interact with an RL agent via

$$z_j \sim q_\phi(z_j|s_j), \tag{16}$$

$$s_{j+1}, r_{j+1} \sim p_\psi(s_{j+1}, r_j|z_j, u_j). \tag{17}$$

Specifically, the DL-MDP first maps the state at stage j, s_j, into the latent variable z_j by sampling from the distribution q_ϕ. After the agent takes action u_j, the DL-MDP responds with the next state s_{j+1} and reward r_j. Fig. 8 shows the mapping flow from s_j, u_j to s_{j+1}, r_j.

To learn such a DL-MDP, we first enrich the trajectories that will be used for training, by including into each tuple the rewards following the reward function as well as the next states; this results in

$$\tilde{\rho}^\mu = \{(s_0, u_0, r_0, s_1), (s_1, u_1, r_1, s_2), \dots | u_j = \mu(s_j), \\ r_j = R(s_j, u_j, s_{j+1})\}. \tag{18}$$

Then, we learn the DL-MDP by maximizing the sum of marginal log-likelihood

$$\sum\nolimits_{j=0}^{T-1} \log p_\psi(s_{j+1}, r_j) = \sum\nolimits_{j=0}^{T-1} \log \int_\mathcal{Z}\int_\mathcal{S}\int_\mathcal{U} \Big[q_\phi(z_j|s_j) \\ p_\psi(s_{j+1}, r_j|z_j, u_j)\mu(u_j|s_j)p(s_j)\Big] ds_j du_j dz_j, \tag{19}$$

where $p(s_j)$ is the probability of ending up in state s_j at stage j, which could be estimated using Monte Carlo methods. However, it is intractable to integrate over the latent space $\mathcal{Z}$ as it remains unknown. Hence, variational inference can be used to learn a DL-MDP by maximizing a lower bound of (19), which is referred to as the evidence lower bound (ELBO).

The next result provides a way to derive the ELBO for DL-MDP.

Theorem 4.1 (ELBO for DL-MDP). *Consider deterministic policy μ, and assume that there exist $\epsilon \in (0, 1)$ such that for each action $u \in \mathcal{U}$, there always exist a set of states $\tilde{\mathcal{S}}_u \subset \mathcal{S}$ from which the action u is taken following μ. Furthermore, the states in $\tilde{\mathcal{S}}_u$ can be visited infinitely often, i.e., exists ϵ s.t. $0 < \epsilon \le \int_{\tilde{\mathcal{S}}_u} p(s)ds \le 1$. Then, an ELBO for any tuple $(s_j, u_j, s_{j+1}, r_j) \sim \tilde{\rho}^\mu$ can be obtained as*

$$\log p_\psi(s_{j+1}, r_j) \ge -KL\Big(q_\phi(z_j|s_j)||p_\psi(z_j)\Big) + \log\epsilon \\ + \mathbb{E}_{z_j\sim q_\phi(z_j|s_j)}\Big[\log p_\psi(s_{j+1}, r_j|z_j, u_j)\Big], \tag{20}$$

where $KL(\cdot||\cdot)$ is the Kullback–Leibler (KL) divergence [27].

The proof can be found in Appendix C.

In practice, it was shown that introducing a constant (*i.e.*, hyper-parameter) $\kappa \in \mathbb{R}$ to the ELBO, provides more flexibility for the model to focus on learning disentangled latent representations (*i.e.*, using smaller κ) or maximizing the likelihood for s_{j+1} and r_j (*i.e.*, using larger κ) [20]. Therefore, the learning objective is set to be

$$\max_{\phi,\psi} \mathcal{L}(\phi, \psi; \tilde{\rho}^\mu) = \sum_{j=0}^{T-1}\Big(-\kappa\cdot KL\big(q_\phi(z_j|s_j)||p_\psi(z_j)\big) \\ + \mathbb{E}_{z_j\sim q_\phi(z_j|s_j)}\big[\log p_\psi(s_{j+1}, r_j|z_j, u_j)\big]\Big). \tag{21}$$

Gradient descent [24] can be used to optimize (21) following the reparameterization trick [25]. Specifically, the prior for the latent variable is usually set to be the centered isotropic multivariate Gaussian $p_\psi(z) = \mathcal{N}(z; 0, \mathbf{I})$. For the MDP we consider, both the encoder $q_\phi(z_j|s_j)$ and decoder $p_\psi(s_{j+1}, r_j|z_j, u_j)$ can be captured by multivariate Gaussian distributions with the mean and diagonal covariance determined by ϕ and ψ, respectively.

Furthermore, both ϕ and ψ can be represented by NNs taking as inputs the variables that the encoder and decoder are conditioned on in (21), respectively − *i.e.*,

$$\phi^j = [\mu_\phi^j, \sigma_\phi^j]^T = f_1(s_j), \tag{22}$$

$$\psi^j = [\mu_\psi^j, \sigma_\psi^j]^T = f_2(z_j, u_j), \tag{23}$$

with f_1 and f_2 captured by NNs. Then, z_j, s_{j+1}, r_j can be obtained via $z_j \sim \mathcal{N}\Big(\mu_\phi^j, (\sigma_\phi^j)^2\mathbf{I}\Big)$ and $[s_{j+1}, r_j]^T \sim \mathcal{N}\Big(\mu_\psi^j, (\sigma_\psi^j)^2\mathbf{I}\Big)$, with $\mathbf{I}$ being the identity matrix. The reparameterization ensures the gradients to be tractable, by replacing the sampling process with

$$z_j = \mu_\phi^j + \sigma_\phi^j\cdot\epsilon_\phi, \tag{24}$$

$$[s_{j+1}, r_j]^T = \mu_\psi^j + \sigma_\psi^j\cdot\epsilon_\psi, \tag{25}$$

with $\epsilon_\phi \sim \mathcal{N}(0, \mathbf{I})$ and $\epsilon_\psi \sim \mathcal{N}(0, \mathbf{I})$ that could be treated as constants during training (*i.e.*, gradient back-propagation). Specifically, by defining $z_j = g_1(\mu_\phi^j, \sigma_\phi^j)$ and $[s_{j+1}, r_j]^T = g_2(\mu_\psi^j, \sigma_\psi^j)$, we have $[s_{j+1}, r_j]^T = g_2\Big(f_2(z_j, u_j)\Big) = g_2\Big(f_2\Big(g_1\big(f_1(s_j)\big), u_j\Big)\Big)$, where the gradients of f_1, f_2, g_1, g_2 could be obtained using the chain rule. We refer to [25] for more details about the reparameterization.

Remark 4.2. *Note that the major difference between the DL-MDP and the regular variational auto-encoders (VAEs) proposed in [25] is that VAEs were originally designed to reconstruct samples to be similar to the training inputs. As a result, there do not exist temporal correlations between the inputs and outputs. Moreover, the encoder q_ϕ is usually discarded after training since during sampling the latent variable z can be obtained directly using the prior $p_\psi(z)$. However, in the MDP environment we consider, temporal correlations are imperative* *between two consecutive stages, following from the Markov property [46]. Therefore, the sampling process described in* (16) *and* (17) *enforces such correlations, and thus the ELBO specific to the setting we consider had to be considered and derived. In addition, the encoder* $q_\phi(z_j|s_j)$ *will not be discarded after training.*

Consequently, after training, the target policy π can be evaluated by interacting with the learned DL-MDP $\hat{\mathcal{M}}$ following (16) and (17), and by choosing $u_j = \pi(s_j)$ for all j. Then, the accumulated return (5) is obtained using the rewards issued from the DL-MDP.

4.2 Integrating OPE into RL Training

As described in Section 2.1.2, in the real world, access to trajectories, that can be used to train RL policies, is usually limited. As illustrated in Fig. 3, new training data may only become available very sparsely over a specific period of time (*i.e.*, between two scheduled patient visits). Although when a patient is not present, a few patient-specific controller candidates can be trained leveraging offline RL methods [29, 35, 43] using historical data, it is unclear which one could result in better performance until they are all tested at the next available trial. Thus, it is critical to devise efficient learning pipelines when limited training resources are accessible.

Alg. 1 in Appendix E introduces an efficient offline RL training framework, in the context of training DQNs, utilizing the OPE method introduced in Section 4.1. Specifically, it takes as input the MDP $\mathcal{M}$ as defined in Section 3, the DL-MDP $\hat{\mathcal{M}}$, the set of policies $\Pi = \{\pi^{(1)}_{\theta_q}, \pi^{(2)}_{\theta_q}, \dots\}$, each parameterized by a corresponding NN, that will be updated and used to collect trajectories for training and evaluation, a constant $f_{patient}$ specifying the frequency in terms of when the patient is available for DBS control evaluation (*i.e.*, once in $f_{patient}$ number of training episodes), another constant f_{eval} indicating the frequency the polices in Π are evaluated by OPE, and a buffer $\mathcal{B}$ to store the trajectories collected during patient trials.

The algorithm starts by initializing the parameters of the DL-MDP $\hat{\mathcal{M}}$, as well as all policies in Π. It also randomly assigns a policy $\pi^{(\cdot)}_{\theta_q} \in \Pi$ to the variable $\tilde{\pi}_{\theta_q}$ which will be updated online and used to collect trajectories once the patient becomes available (*i.e.,* trials). Lines 5-15 of Alg. 1 correspond to the case when the patient is available; thus, the policy currently stored in $\tilde{\pi}_{\theta_q}$ is updated online by directly interacting with the patient, and the state-action tuples (s, u, s', r) collected are appended to the buffer $\mathcal{B}$. For simplicity, here we only use a single tuple to update the policy; yet, it can be extended to batch updates (see [16]). When the patient becomes unavailable (*i.e.*, lines 17-19), all policies in Π are updated offline, using the data in $\mathcal{B}$. Finally, the OPE method is used to evaluate all policies in Π every f_{eval} number of episodes (lines 21-29). Specifically, the parameters of DL-MDP are first updated using the data in $\mathcal{B}$, after which it is used to interact with each policy $\pi^{(\cdot)}_{\theta_q} \in \Pi$ to estimate the expected return over $max\_iter + 1$ steps. Then, the policy with the maximum estimated return is assigned to $\tilde{\pi}_{\theta_q}$.

Note that, unlike (18), where the trajectory is collected following a single policy, the buffer $\mathcal{B}$ contains trajectories collected following a mixture of policies obtained at different training episodes. Also, the algorithm could be easily extended to actor-critic RL methods, such as deep deterministic policy gradient (DDPG) [29], by declaring two NNs at the beginning, to represent the actor and critic, and following updates (10) and (11) in line 12 of Alg. 1.

5 NUMERICAL EXPERIMENTS

We employed the BGM to evaluated the proposed OPE methodology. The evaluation was done from two perspectives: (*i*) estimating the expected return of a *target* policy π using trajectories obtained by a *behavioral* policy μ, and (*ii*) enhancing RL training with limited access to the controlled process (*i.e.,* the BGM). All experimental results were obtained using a server with 3 Nvidia RTX Quadro 6000 GPUs. The deep learning models were implemented in Python using Tensorflow [1]. The BGM sampling window size was set to be $T_w = 2$ seconds in both case studies. Adam optimizer [24] was used to calculate the gradients for (10), (11), (15), (24) and (25).

5.1 Expected Return Estimation

We first considered *offline* estimations of the *expected returns*, as defined in (6), w.r.t. 20 target policies $\{\pi_1, \dots, \pi_{20}\}$, in the context of two DBS control problems for the BGM, *i.e.*, stimulation pulse pattern synthesis (Section 3.1) and frequency selection (Section 3.2).

To achieve this, in both cases, the trajectories ρ^μ, obtained following 4 behavioral policies $\{\mu_1, \dots, \mu_4\}$, were used to train the DL-MDP. Specifically, ρ^μ was obtained by deploying each μ_i, $i \in [1, 4]$, to the BGM for 15 episodes, with horizon $T = 200$ steps each, and concatenated together all the trajectories obtained. As a result, it contained $|\rho^\mu| = 12,000$ recorded (s_j, u_j, r_j, s_{j+1}) tuples. The details for synthesizing π's and μ's can be found in the next subsections.

In DL-MDP, both the encoder q_ϕ and decoder p_ψ were modeled by a 2-layer NN with 384 and 128 nodes each. During training, the learning rate was set to 0.001 and the NNs were trained with 80,000 gradient descent steps. Then, the estimated return of π_i, $i \in [1, 20]$, was estimated by interacting with the DL-MDP for 50 episodes, following (16) and (17) and choosing $u_j = \pi_i(s_j)$ at all time steps, followed by averaging the *accumulated return* (5), over all episodes.

Two existing OPE methods were used as baselines, *i.e.*, the step-wise weighted IS from [40], and the density estimation IS (DEIS) from [32], which uses estimated density ratio of stationary state distributions to reduce variance in IS weights, and is considered a state-of-the-art OPE approach (details provided in Appendix D). The methods were evaluated using two metrics: (*i*) root mean squared error (RMSE) between the *estimated* returns and *ground-truth* returns, obtained by deploying each π_i to the BGM for 50 episodes and reporting the averaged accumulated returns over all the target policies; and (*ii*) rank correlation, captured by Spearman's correlation coefficient [42] between the rank of estimated and ground-truth returns. Note that the metric (*i*) evaluates the estimation error made by OPE, while (*ii*) inspects if the rank of estimated returns over all target policies, is aligned with the returns that could be obtained by directly interacting with the controlled process. To demonstrate the *robustness* of each method to system stochasticity, the above procedure was repeated 3 times for each method, and the resulting mean and variance for all metrics are reported.

5.1.1 *OPE for DBS Pattern Control.* Four target policies, $\{\pi_1, \dots, \pi_4\}$, were first obtained by training directly on the BGM for 50 episodes, with horizon $T = 200$ steps each, using different combinations of actor and critic learning rates, *i.e.*, $\{(\alpha_u = 5\text{e-}5, \alpha_c = 5\text{e-}4), (\alpha_u =$

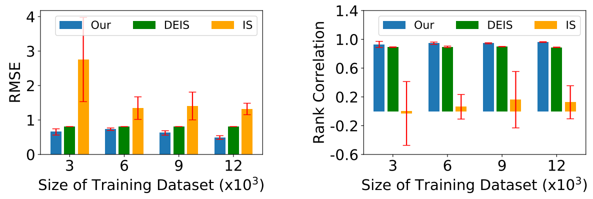

Figure 9: RMSE (left) and rank correlation (right) obtained from the proposed OPE method versus the baselines when evaluating the RL-based stimulation pattern controllers.

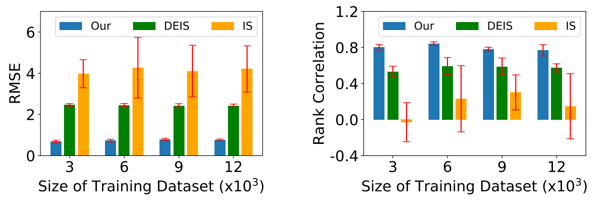

Figure 10: RMSE (left) and rank correlation (right) obtained from the proposed OPE method versus the baselines when evaluating the RL-based stimulation frequency controllers.

1e-5, α_c = 1e-4), (α_u = 5e-6, α_c = 5e-5), (α_u = 1e-6, α_c = 1e-5)} as in (10) and (11). Then, four behavioral policies, $\{\mu_1, \ldots, \mu_4\}$, were obtained by training on the BGM for 10 episodes using the same hyper-parameters as above. To ensure the target policies could result in a wide range of expected returns, we also considered *synthetic* target policies as suggested in [21], which facilitated illustrating the *robustness* of the OPE methods to policies with various levels of performance. Specifically, such policies were defined as

$$\pi^{syn}_{i,c} = \begin{cases} \pi_i & \text{with prob. } c, \\ \mu_i & \text{with prob. } 1-c, \end{cases} \quad i \in [1, 4]. \tag{26}$$

By choosing $c \in \{.2, .4, .6, .8\}$, four additional target policies were defined for each π_i. The average ground-truth return obtained across 20 target policies, and 4 behavioral policies, were -0.51 (max 1.56, min -1.59, std 0.93) and -0.91 (max 0.83, min -1.52, std 1.0), respectively; these were obtained using discounting factor $\gamma = 0.5$.

The resulting RMSE and rank correlation obtained by our OPE method, DEIS, and IS are shown in Fig. 9. The IS leads to the highest estimation error and lowest rank correlations, along with the highest standard deviations (shown in the error bars) across different runs of the experiments. In contrast, our approach significantly outperforms IS in all experimental settings. Furthermore, we also achieve lower RMSE and higher rank correlation than the state-of-the-art DEIS; they become more significant once sufficient training data are provided (*i.e.*, $|\rho^\mu| \geq 9 \times 10^3$.)

5.1.2 OPE for DBS Frequency Control. Four target policies were first obtained by training on the BGM for 50 episodes with $T = 200$, using DQN with the architecture same as the critic used in Section 5.1.1, and learning rates from $\alpha_q \in$ {1e-3, 1e-4, 5e-5, 1e-5}. Then, four behavioral policies were obtained by training DQNs using the learning rates above, but for 10 episodes only. Finally, 16 additional (*i.e.*, synthetic) target policies were generated following (26), which resulted in a total of 20 target policies subject to

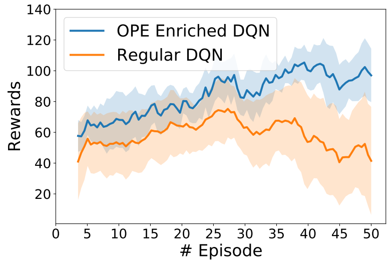

Figure 11: Per-episode (non-discounted) accumulated rewards obtained by OPE enriched DQN versus vanilla DQN. Standard deviations over 4 runs are shown in the shadow.

be evaluated by OPE. The average ground-truth return obtained across 20 target policies, and 4 behavioral policies, were 2.14 (max 3.65, min -0.78, std 1.42) and -0.37 (max 2.08, min -1.7, std 1.25), respectively, for $\gamma = 0.5$. The results are summarized in Fig. 10, showing that our method significantly outperforms both baselines in terms of the two considered metrics.

5.1.3 Summary. As demonstrated above, the proposed method consistently achieves low RMSE and high rank correlations while evaluating policies at various levels (*i.e.*, with a wide range of ground-truth expected returns). This shows that it can effectively and accurately evaluate the performance of target policies in an offline manner. In addition, the low standard deviations of the metrics attained in most experiments illustrate that our method is robust to stochasticity from the controlled physical process/environment.

5.2 Enhanced RL Training with Limited Data

In this case study, we employ Alg. 1 to learn policies that adjust stimulation frequencies, with the patient represented by the BGM that is only occasionally available for trials. We set that the total number of training episodes $max\_iter = 50$, the horizon $max\_iter = 200$, BGM comes online every $f_{patient} = 5$ episodes, and the OPE method was called one episode before the BGM becomes available. In addition, four policy candidates were considered (*i.e.*, $\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$), and each of them follows a different $\alpha_q \in$ {1e-3, 1e-4, 5e-5, 1e-5}.

The performance of Alg. 1 was compared to a baseline that simply sets the policy being updated once the BGM was available, $\tilde{\pi}_{\theta_q}$, to a policy randomly sampled from Π. Fig. 11 shows the accumulated rewards obtained at each episode resulting from Alg. 1-derived policy versus the baseline. Specifically, the mean and standard deviation over 4 different runs (*i.e.*, following 4 random seeds) are reported. Non-discounted returns are used since they capture the raw performance propagated from each step. It can be observed that the OPE-enriched RL leads to consistently higher returns, especially at the later stage (*i.e.*, ≥ 25 episodes) where the mean performance is always greater than the best one obtained by the regular DQN; thus, showing effectiveness of our OPE-enhanced RL framework.

6 DISCUSSION AND CONCLUSION

In this work, we have developed a model-based OPE method that can estimate the performance of RL-based DBS controllers using historical data, which facilitate *safe* (*i.e.*, no patient interactions needed) and *effective* evaluations in clinical settings. Furthermore, we have shown how the OPE approach can be integrated into offline RL frameworks to allow for efficient controller training.

We evaluated our approach using a computational BGM which models the neuronal activity of the BG. The results show that our method can estimate and rank the expected returns of the target policies *precisely*, and is *robust* to stochasticity in the BG (*i.e.*, plant) and environmental disturbance, as well as target policies with various levels of performance. Moreover, the second case study mimics the limited data access setting as in clinical trials. The improved efficacy of the resulting controller illustrates that our method can also improve offline RL training of DBS controllers even when only limited trajectories can be obtained.

As part of future work, the proposed OPE method will be implemented practically in clinical studies to serve as a *safe* and *effective* approach for evaluating learning-based DBS controllers. Moreover, the OPE framework could also be extended to other types of more classical (*e.g.*, non-learning) controllers. On the other hand, although our method lays out the foundation for developing *data-efficient* learning-based controllers, it could be potentially generalized to solve other clinical decision-making problems. For example, evaluating/creating future treatment plans for PD patients using their past electrical health records (EHRs). Here, the treatment plans can be modeled as policies while the EHRs are analogous to the trajectories used for updating and evaluating them.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/

[2] Mattia Arlotti, Manuela Rosa, et al. 2016. The adaptive deep brain stimulation challenge. Parkinsonism & related disorders 28 (2016), 12–17.

[3] Mattia Arlotti, Lorenzo Rossi, et al. 2016. An external portable device for adaptive deep brain stimulation (aDBS) clinical research in advanced Parkinson's Disease. Medical engineering & physics 38, 5 (2016), 498–505.

[4] Alim Louis Benabid. 2003. Deep brain stimulation for Parkinson's disease. Current opinion in neurobiology 13, 6 (2003), 696–706.

[5] Aleksandar Beric, Patrick J Kelly, et al. 2001. Complications of deep brain stimulation surgery. Stereotactic and functional neurosurgery 77, 1-4 (2001), 73–78.

[6] M Beudel and P Brown. 2016. Adaptive deep brain stimulation in Parkinson's disease. Parkinsonism & related disorders 22 (2016), S123–S126.

[7] A. K. Bozkurt, Y. Wang, and M. Pajic. 2021. Secure Planning Against Stealthy Attacks via Model-Free Reinforcement Learning. In ICRA. 10656–10662.

[8] Greg Brockman, Vicki Cheung, et al. 2016. Openai gym. arXiv:1606.01540 (2016).

[9] Noe Casas. 2017. Deep deterministic policy gradient for urban traffic light control. arXiv preprint arXiv:1703.09035 (2017).

[10] Bo Dai, Ofir Nachum, et al. 2020. Coindice: Off-policy confidence interval estimation. arXiv preprint arXiv:2010.11652 (2020).

[11] Lonneke ML De Lau and Monique MB Breteler. 2006. Epidemiology of Parkinson's disease. The Lancet Neurology 5, 6 (2006), 525–535.

[12] Günther Deuschl, Carmen Schade-Brittinger, et al. 2006. A randomized trial of deep-brain stimulation for Parkinson's disease. New England Journal of Medicine 355, 9 (2006), 896–908.

[13] Kenneth A Follett, Frances M Weaver, et al. 2010. Pallidal versus subthalamic deep-brain stimulation for Parkinson's disease. New England Journal of Medicine 362, 22 (2010), 2077–2091.

[14] Centers for Disease Control and Prevention. 2020. Epilepsy Data and Statistics. https://www.cdc.gov/epilepsy/data/index.html

[15] Justin Fu, Mohammad Norouzi, et al. 2020. Benchmarks for Deep Off-Policy Evaluation. In ICLR.

[16] Qitong Gao, Michael Naumann, et al. 2020. Model-Based Design of Closed Loop Deep Brain Stimulation Controller using Reinforcement Learning. In 2020 ACM/IEEE 11th Int. Conf. on Cyber-Physical Systems (ICCPS). IEEE, 108–118.

[17] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Int. Conf. on robotics and automation (ICRA). IEEE, 3389–3396.

[18] A. Guez, R. D. Vincent, M. Avoli, and J. Pineau. 2008. Adaptive Treatment of Epilepsy via Batch-mode Reinforcement Learning. In AAAI. 1671–1678.

[19] J. Habets, M. Heijmans, et al. 2018. An update on adaptive deep brain stimulation in Parkinson's disease. Movement Disorders 33, 12 (2018), 1834–1843.

[20] Irina Higgins, Loic Matthey, et al. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. In ICLR.

[21] Nan Jiang and Lihong Li. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In ICML. PMLR, 652–661.

[22] Ilija Jovanov, Michael Naumann, et al. 2018. Platform for model-based design and testing for deep brain stimulation. In Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems. IEEE Press, 263–274.

[23] Sanket Kamthe and Marc Deisenroth. 2018. Data-efficient reinforcement learning with probabilistic model predictive control. In AISTATS. PMLR, 1701–1710.

[24] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[25] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013).

[26] A.A. Kühn, A. Kupsch, GH. Schneider, and P Brown. 2006. Reduction in subthalamic 8–35 Hz oscillatory activity correlates with clinical improvement in Parkinson's disease. Euro. J. of Neuroscience 23, 7 (2006), 1956–1960.

[27] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. The annals of mathematical statistics 22, 1 (1951), 79–86.

[28] Alexis M Kuncel and Warren M Grill. 2004. Selection of stimulus parameters for deep brain stimulation. Clinical neurophysiology 115, 11 (2004), 2431–2441.

[29] Timothy P Lillicrap, Jonathan J Hunt, et al. 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).

[30] Simon Little, Alex Pogosyan, et al. 2013. Adaptive deep brain stimulation in advanced Parkinson disease. Annals of neurology 74, 3 (2013), 449–457.

[31] Simon Little, Elina Tripoliti, et al. 2016. Adaptive deep brain stimulation for Parkinson's disease demonstrates reduced speech side effects compared to conventional stimulation in the acute setting. J Neurol Neurosurg Psychiatry 87, 12 (2016), 1388–1389.

[32] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. 2018. Breaking the Curse of Horizon: Infinite-Horizon Off-Policy Estimation. In NeurIPS.

[33] Andres M Lozano, Jonathan Dostrovsky, Robert Chen, and Peter Ashby. 2002. Deep brain stimulation for Parkinson's disease: disrupting the disruption. The Lancet Neurology 1, 4 (2002), 225–231.

[34] C Marras, JC Beck, et al. 2018. Prevalence of Parkinson's disease across North America. NPJ Parkinson's disease 4, 1 (2018), 21.

[35] Volodymyr Mnih, Koray Kavukcuoglu, et al. 2015. Human-level control through deep reinforcement learning. Nature 518, 7540 (2015), 529.

[36] Vivek Nagaraj, Andrew Lamperski, and Theoden I Netoff. 2017. Seizure control in a computational model using a reinforcement learning stimulation paradigm. International J. of Neural Sys. 27, 07 (2017), 1750012.

[37] Michael S Okun. 2012. Deep-brain stimulation for Parkinson's disease. New England Journal of Medicine 367, 16 (2012), 1529–1538.

[38] Bahram Parvinian, Christopher Scully, et al. 2018. Regulatory considerations for physiological closed-loop controlled medical devices used for automated critical care: food and drug administration workshop discussion topics. Anesthesia and analgesia 126, 6 (2018), 1916.

[39] J. Pineau, A. Guez, et al. 2009. Treating epilepsy via adaptive neurostimulation: a reinforcement learning approach. Int. J. of Neural Sys. 19, 04 (2009), 227–240.

[40] Doina Precup. 2000. Eligibility traces for off-policy policy evaluation. Computer Science Department Faculty Publication Series (2000), 80.

[41] Joshua Redding, Alborz Geramifard, and Jonathan How. 2010. Actor-critic policy learning in cooperative planning. In 2010 AAAI Spring Symposium Series.

[42] P. Schober, C. Boer, and L. A. Schwarte. 2018. Correlation coefficients: appropriate use and interpretation. Anesthesia & Analgesia 126, 5 (2018), 1763–1768.

[43] David Silver, Guy Lever, et al. 2014. Deterministic policy gradient algorithms.

[44] Rosa Q So, Alexander R Kent, and Warren M Grill. 2012. Relative contributions of local cell and passing fiber activation and silencing to changes in thalamic fidelity during deep brain stimulation and lesioning: a computational modeling study. Journal of computational neuroscience 32, 3 (2012), 499–519.

[45] Fei Su, Karthik Kumaravelu, Jiang Wang, and Warren M Grill. 2019. Model-based evaluation of closed-loop deep brain stimulation controller to adapt to dynamic changes in reference signal. Frontiers in neuroscience 13 (2019), 956.

[46] R. Sutton and A. Barto. 2018. Reinforcement learning: An introduction. MIT press.

[47] Ziyang Tang, Yihao Feng, et al. 2019. Doubly Robust Bias Reduction in Infinite Horizon Off-Policy Estimation. In ICLR.

[48] Philip Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In ICML. PMLR, 2139–2148.

[49] Jeremy Watts, Anahita Khojandi, Oleg Shylo, and Ritesh A Ramdhani. 2020. Machine learning's application in deep brain stimulation for Parkinson's disease: A review. Brain Sciences 10, 11 (2020), 809.

[50] Mengjiao Yang, Ofir Nachum, et al. 2020. Off-Policy Evaluation via the Regularized Lagrangian. In NeurIPS, Vol. 33.

[51] Shangtong Zhang, Bo Liu, and Shimon Whiteson. 2020. Gradientdice: Rethinking generalized offline estimation of stationary values. In ICML. PMLR, 11194–11203.

[52] Yulin Zhu, Jiang Wang, et al. 2021. Adaptive Parameter Modulation of Deep Brain Stimulation Based on Improved Supervisory Algorithm. Frontiers in neuroscience (2021), 1187.

A NOTATION

Here, we define the notation and terms used in the paper. For the real number, integer and positive integer spaces, we denote them as $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{Z}^+$, respectively. We also define $\mathbb{1}$ is the indicator function, *i.e.*,

$$\mathbb{1}(x) = \begin{cases} 1 & \text{if condition } x \text{ is satisfied,} \\ 0 & \text{if condition } x \text{ is not satisfied.} \end{cases} \tag{27}$$

$\mathbf{I}$ is the identity matrix of suitable size. Moreover, $x \sim p(x)$ means that random variable x is sampled from distribution $p(x)$. We also use $\mathcal{N}(x; \mu, \Sigma)$ to denote Gaussian distributions with mean μ and covariance matrix Σ over variable x. For simplicity, we write $x \sim \mathcal{N}(\mu, \Sigma)$ during sampling. The KL-divergence between distributions $p(x)$ and $q(x)$ is defined as

$$KL(p||q) = \mathbb{E}_p \left[\log \frac{q(x)}{p(x)} \right]. \tag{28}$$

B COMPUTATIONAL BASAL GANGLIA MODEL

We now provide details about the BGM introduced in [44]. Assuming that there exist N neurons in each sub-region, the state of each BGM sub-region, at time step t, can be represented as a vector of potentials, *i.e.*,

$$\mathbf{v}^q(t) = [v_1^q(t), \dots, v_N^q(t)]; \tag{29}$$

here, $v_i^q(\cdot)$ denotes the electrical potential of the i^{th} neuron in the sub-region $q \in \{STN, GPe, GPi, TH\}$. Note that [44] showed that by selecting $N = 10$ the fidelity of BGM is very close to the ones with $N >> 10$.

The activation of each neuron, at time step t, is captured by a discrete event as $a_i^q(t) \in \{0, 1\}$, defined as

$$a_i^q(t) = \mathbb{1}\left(\left[v_i^q(t) > h_i^q \right] \wedge \left[\exists \delta > 0, \forall \varepsilon \in (0, \delta], v_i^q(t - \varepsilon) < h_i^q \right] \right);$$

here, $\mathbb{1}$ is the indicator function, h_i^q is a pre-defined threshold such that the neuron is considered activated once the potential $v_i^q(t)$, which has highly nonlinear dynamics, crosses over it.

We formally define the two QoC metrics used to quantify the severity of PD symptoms (see *e.g.*, [16, 22, 44] and references within), *i.e.*, EI and beta power spectral density (P_β). EI is defined as the portion of erroneous TH neuron activations in response to sensorimotor cortex (SMC) inputs at $t = \tau$, *i.e.*, SMC_τ. Specifically, SMC_τ can change the TH neuron potentials, $\mathbf{v}^{TH}(t)$, and should activate all TH neurons *exactly once* within a $25ms$ window in *healthy brains*, *i.e.*, $\forall i\ \exists! t_i \in [\tau, \tau + 25ms]$ s.t. $a_i^{TH}(t_i) = 1$. However, in PD brains, the TH neurons may not be activated or may respond with multiple activations within the $25ms$ window following an SMC input. Formally, EI is defined as

$$EI(t) = \frac{\sum_{i=1}^{N} \sum_{t=t_0}^{t} a_i^{TH,err}(t)}{N \left| SMC_\tau \big|_{t_0}^{t} \right|}; \tag{30}$$

here, $a_i^{TH,err}(t) \in \{0, 1\}$, and $a_i^{TH,err}(t) = 1$ (or 0) represents an erroneous (or correct) TH neuron activation at time t, and $\left| SMC_\tau \big|_{t_0}^{t} \right|$ is the cumulative number of SMC inputs received between the initial time step t_0 and current step t.

The other metric P_β is defined as

$$P_\beta = \frac{1}{N} \sum_{i=1}^{N} \int_{\omega = 2\pi \cdot 13Hz}^{2\pi \cdot 35Hz} P_i^{GPi}(\omega) d\omega, \tag{31}$$

where $P_i^{GPi}(\omega)$ is the single-sided power spectral density of the i^{th} neuron's potential in the GPi sub-region.

C PROOF OF THEOREM 4.1.

Proof. We start by simplifying the KL-divergence between q_ϕ and p_ψ – *i.e.*,

$$KL\left(q_\phi(z_j, u_j | s_j, s_{j+1}, r_j) || p_\psi(z_j, u_j | s_j, s_{j+1}, r_j) \right) \tag{32}$$
$$= KL\left(q_\phi(z_j | s_j) || p_\psi(z_j, u_j | s_{j+1}, r_j) \right) \tag{33}$$
$$= \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \left[\log \frac{q_\phi(z_j|s_j)}{p_\psi(z_j, u_j | s_{j+1}, r_j)} \right] \tag{34}$$
$$= \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log q_\phi(z_j|s_j) - \log p_\psi(z_j, u_j, s_{j+1}, r_j) + \log p_\psi(s_{j+1}, r_j) \Big] \tag{35}$$
$$= \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log q_\phi(z_j|s_j) - \log p_\psi(z_j, u_j) - \log p_\psi(s_{j+1}, r_j | z_j, u_j) + \log p_\psi(s_{j+1}, r_j) \Big] \tag{36}$$
$$= \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log \frac{q_\phi(z_j|s_j)}{p_\psi(z_j, u_j)} - \log p_\psi(s_{j+1}, r_j | z_j, u_j) \Big] + \log p_\psi(s_{j+1}, r_j) \geq 0. \tag{37}$$

Specifically, the transition between (32) and (33) follows from Bayes rule, *i.e.*,

$$q_\phi(z_j, u_j | s_j, s_{j+1}, r_{j+1}) = q_\phi(z_j | s_j, s_{j+1}, r_{j+1}, u_j) q_\phi(u_j | s_j, s_{j+1}, r_{j+1}).$$

Then, the first term can be reduced to $q_\phi(z_j|s_j)$ since z_j is independent from s_{j+1}, r_{j+1}, u_j by the DL-MDP definition. The second term can be reduced to $q_\phi(u_j|s_j)$ which should always equals to one since deterministic policies are considered. The transition from (33) to (34) follows from the definition of KL-divergence. In what follows, (34) to (37) take advantage of Bayes rule and rearrange terms. The inequality in (37) holds since the KL-divergence of any two distributions is greater than or equal to zero. Now by rearranging the terms we obtain that

$$\log p_\psi(s_{j+1}, r_j) \geq -\mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log \frac{q_\phi(z_j|s_j)}{p_\psi(z_j, u_j)} \Big] + \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log p_\psi(s_{j+1}, r_j | z_j, u_j) \Big] \tag{38}$$
$$= -\mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log \frac{q_\phi(z_j|s_j)}{p_\psi(z_j)} \Big] + \log p_\psi(u_j) + \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log p_\psi(s_{j+1}, r_j | z_j, u_j) \Big] \tag{39}$$
$$\geq -KL\left(\log q_\phi(z_j|s_j) || p_\psi(z_j) \right) + \log \epsilon + \mathbb{E}_{z_j \sim q_\phi(z_j|s_j)} \Big[\log p_\psi(s_{j+1}, r_j | z_j, u_j) \Big]. \tag{40}$$

Note that the transition between (38) and (39) follows from the fact that z_j and u_j are independent; thus $p_\psi(z_j, u_j) = p_\psi(z_j)p_\psi(u_j)$. Then, we have $p_\psi(u_j) = \int_S p_\psi(u_j|s)p_\psi(s)ds = \int_{\tilde{S}_u} p_\psi(s)ds \in [\epsilon, 1]$ by the assumption. □

D EXISTING IS BASED OPE METHODS

Importance sampling (IS) refers to a statistical technique that can calculate the expectation of function $f(x)$ w.r.t. an unknown distribution $p(x)$ using a given distribution $q(x)$ through re-weighting [32], *i.e.*,

$$\mathbb{E}_p[f(x)] = \mathbb{E}_q\left[\frac{f(x)p(x)}{q(x)}\right]. \tag{41}$$

This technique can be applied in the context of OPE by setting $f(x)$ as the accumulated return G_0, p as the trajectory distribution ρ^π over the target policy π, and q as the trajectory distribution ρ^μ over the behavioral policy μ. Specifically, suppose that there exist a total of n trajectories in ρ^μ and each corresponds to horizon T, *i.e.*,

$$\begin{aligned}\rho^\mu = \Big\{&\big[(s_{0,0}, u_{0,0}, r_{0,0}), \dots (s_{0,T}, u_{0,T}, r_{0,T})\big], \dots, \\ &\big[(s_{n,0}, u_{n,0}, r_{n,0}), \dots (s_{n,T}, u_{n,T}, r_{n,T})\big] \\ &\Big| u_{i,j} = \mu(s_{i,j}), r_{i,j} = R(s_{i,j}, u_{i,j}, s_{i,j+1})\Big\}.\end{aligned} \tag{42}$$

Then, the expected return over the unknown ρ^π can be obtained as

$$\mathbb{E}_{s\sim\rho^\pi}[G_0] = \frac{1}{n}\sum_{i=1}^{n} G_0^{(i)} \prod_{j=0}^{T} \frac{\pi(u_{i,j}|s_{i,j})}{\mu(u_{i,j}|s_{i,j})}; \tag{43}$$

here, $G_0^{(i)}$ is the accumulated return from the i^{th} trajectory in ρ^μ, and $\prod_{j=0}^{T} \frac{\pi(u_{i,j}|s_{i,j})}{\mu(u_{i,j}|s_{i,j})}$ are usually referred to as the IS weights. It contains multiplications of distributions (*i.e.*, behavioral and target policies) over horizon T; thus, it is considered to have high variance across different trajectories [32, 40]. A common and intuitive way of reducing the variance is to normalize the IS weights, *i.e.*,

$$\mathbb{E}_{s\sim\rho^\pi}[G_0] = \frac{\sum_{i=1}^{n} G_0^{(i)} \prod_{j=0}^{T} \frac{\pi(u_{i,j}|s_{i,j})}{\mu(u_{i,j}|s_{i,j})}}{\sum_{i=1}^{n} \prod_{j=0}^{T} \frac{\pi(u_{i,j}|s_{i,j})}{\mu(u_{i,j}|s_{i,j})}}. \tag{44}$$

Moreover, the weighted IS is extended to the step-wise weighted IS in [40], which we used as a baseline in Section 5.1. Specifically, each reward r_j is weighted along a trajectory according to the likelihood up to stage j as, *i.e.*,

$$\mathbb{E}_{s\sim\rho^\pi}[G_0] = \frac{\sum_{i=1}^{n}\sum_{j=0}^{T}\sum_{k=1}^{T-j} \gamma^{k-1} r_{j+k} \prod_{l=j+1}^{j+k-1} \frac{\pi(u_{i,l}|s_{i,l})}{\mu(u_{i,l}|s_{i,l})}}{\sum_{i=1}^{n}\sum_{j=0}^{T}\sum_{k=1}^{T-j} \gamma^{k-1} \prod_{l=j+1}^{j+k-1} \frac{\pi(u_{i,l}|s_{i,l})}{\mu(u_{i,l}|s_{i,l})}}, \tag{45}$$

which can be approximated using eligibility traces introduced in [40, 46].

The state-of-the-art density estimation importance sampling (DEIS), introduced in [32], finds that significant decrease in estimation variance is possible by applying importance weighting directly on the state space, instead of the trajectory space as in the IS introduced above. The authors start with defining the average state visitation distribution as

$$\lim_{T\to\infty}\left(\sum_{j=0}^{T} \gamma^t d_{\pi,j}(s)\right) \Big/ \left(\sum_{j=0}^{T}\gamma^t\right), \tag{46}$$

where $d_{\pi,j}(\cdot)$ is the distribution of state s_j when executing policy π starting from the initial state s_0. Then the expected return over policy π can be calculated as

$$\mathbb{E}_{s,u\sim d_\pi}[G_0] = \mathbb{E}_{s,u\sim d_\mu}\left[\frac{d_\pi(s)\pi(u|s)}{d_\mu(s)\mu(u|s)} G_0\right]; \tag{47}$$

thus, the IS estimator for the RHS of (47) can be obtained as

$$\sum_{i=1}^{n}\sum_{j=0}^{T} \frac{\gamma^j \frac{d_\pi(s_{i,j})\pi(a_{i,j}|s_{i,j})}{d_\mu(s_{i,j})\mu(a_{i,j}|s_{i,j})}}{\sum_{i',j'} \gamma^{j'} \frac{d_\pi(s_{i',j'})\pi(a_{i',j'}|s_{i',j'})}{d_\mu(s_{i',j'})\mu(a_{i',j'}|s_{i',j'})}} r_{i,j}, \tag{48}$$

with $d_\pi(\cdot)$ being the only term that is unknown. At last, the authors introduce various techniques to approximate it such as NNs.

E ALGORITHM FOR INTEGRATING OPE INTO RL-BASED CONTROL DESIGN

Algorithm 1 OPE enriched offline RL (DQN)

```
Require: $\mathcal{M}$, $\hat{\mathcal{M}}$, $\Pi = \{\pi^{(1)}_{\theta_q}, \pi^{(2)}_{\theta_q}, \dots\}$, $f_{patient}$, $f_{eval}$, $\mathcal{B}$
Ensure:
1: Initialize the network parameters, $\theta_q$'s, for all $\pi^{(\cdot)}_{\theta_q} \in \Pi$.
2: Initialize the parameters in the DL-MDP $\hat{\mathcal{M}}$.
3: $\tilde{\pi}_{\theta_q} \leftarrow \pi^{(\cdot)}_{\theta_q}$.
4: for $epi$ = 0 to $max\_epi$ do
5:     if $epi \mod f_{patient} == 0$ then            ▷ Interact with the patient
6:         Collect from the patient trial EI and $P_\beta$ with duration $l$ and form the initial state $s_0$ following (1).
7:         $s \leftarrow s_0$
8:         for $j$ = 0 to $max\_iter$ do
9:             $u \leftarrow \tilde{\pi}_{\theta_q}(s)$            ▷ Follow the chosen policy
10:            Provide control stimuli $u$. In the mean time, collect from the brain EI and $P_\beta$ with duration and form $s'$.
11:            Calculate reward $r = R(s, u, s')$.
12:            Use tuple $(s, u, r, s')$ to update the corresponding $\theta_q$ following (15).
13:            Append tuple $(s, u, r, s')$ to the training buffer $\mathcal{B}$.
14:            $s \leftarrow s'$
15:        end for
16:    else            ▷ Train with historical data
17:        for $j$ = 0 to $max\_iter$ do
18:            Sample $(s, u, r, s') \sim \mathcal{B}$ and update all $\pi^{(\cdot)}_{\theta_q} \in \Pi$.
19:        end for
20:    end if
21:    if $epi \mod f_{eval} == 0$ then            ▷ Evaluate policies using OPE
22:        Initialize an empty list $G = [\,]$.
23:        Update parameters of $\hat{\mathcal{M}}$ using all data in $\mathcal{B}$ following (21).
24:        for each $\pi^{(\cdot)}_{\theta_q} \in \Pi$ do
25:            Let $\pi^{(\cdot)}_{\theta_q}$ interacts with $\hat{\mathcal{M}}$ for $max\_iter + 1$ steps.
26:            Collect rewards and calculate accumulative returns following (5). Append the return to the end of list $G$.
27:        end for
28:        $\tilde{\pi}_{\theta_q} \leftarrow \pi^{(\text{argmax}\,G)}_{\theta_q}$
29:    end if
30: end for
31: $\pi^*(\cdot) \leftarrow \tilde{\pi}_{\theta_q}(\cdot)$
32: return $\pi$
```