**Closed-Loop Neural Control Using Deep Reinforcement Learning**

Brandon S. Coventry1, Edward L. Bartlett1,2

1Weldon School of Biomedical Engineering, the Center for Implantable Devices, and the Institute for Integrative Neuroscience, Purdue University, West Lafayette, IN USA

2Department of Biological Sciences, Purdue University, West Lafayette, IN USA

Author Contributions:

BSC and ELB conceived the study, acquired funding, and wrote and reviewed the manuscript. BSC designed the closed-loop algorithms, performed surgeries, acquired data, and designed the protocol. ELB supervised the study

Address for correspondence:

Lead Contact

Edward L Bartlett

Weldon School of Biomedical Engineering

Department of Biological Sciences

Purdue University

206 S. Martin Jischke Dr.

West Lafayette, IN 47907

email: [ebartle@purdue.edu](mailto:ebartle@purdue.edu)

Technical Contact

Brandon S Coventry

Department of Neurological Surgery

University of Wisconsin-Madison

1111 Highland Ave RM 3531

Madison, WI 53705

email: [coventry@wisc.edu](mailto:coventry@wisc.edu)

Author Competing Interests:

BSC and ELB hold a provision patent (USPTO: 18/083490) related to the technology described in this protocol.

**Summary**

Closed-loop neural control is a powerful tool for both scientific exploration of neural function as well as a potent clinical tool to mitigate deficits found in open-loop deep brain stimulation. Reinforcement learning (RL) offers the ability to learn subject-specific neural dynamics in real-time. We present a protocol to integrate RL closed-loop control into neuroscience and neuromodulation studies. We also outline the types of research questions that can be addressed through closed-loop RL control.

For complete details on the use and execution of this protocol, please refer to Coventry et al and Coventry and Bartlett1

**Before You Begin**

The protocol herein describes the use of reinforcement learning to learn neural firing patterns and stimulation parameters to drive firing patterns to target states. Critical to algorithm performance is appropriate computational hardware setup and proper surgical preparation to ensure quality neural recordings. This section describes setup of RL algorithms and surgical preparation.

**Institutional Permissions**

To ensure both animal health and welfare as well as obtaining the highest quality data, all animal work should be approved by an appropriate institutional animal governing committee with all procedures following the appropriate ethical guidelines. All animal procedures in this protocol were approved by Purdue University Animal Care and Use Committee (PACUC protocol #120400631) and in accordance with AAALAC laboratory animal practice.

**SpikerNet, Deep RL setup**

**Timing: 1 day, contingent on availability of inhouse computers**

This step describes the implementation of SpikerNet RL based neural control algorithms.

1. Acquire an AI enabled computer. The authors suggest a Windows or Linux operating system-based computer with minimum intel i7 or AMD Ryzen 7 8700 CPU. Check respective company documents for most up-to-date AI processor solutions. A minimum of 16 GB of ram is recommended, with higher providing better processing speed. Both Nvidia GTX/RTX series and Intel ARC A7x GPUs have PyTorch backends and are recommended. However, Nvidia GPUs have the benefit of wide scale implementation in PyTorch over Intel A7x solutions.
2. Install a Python package management software. The authors utilize Anaconda, but other solutions can work just as well.
3. Download the SpikerNet source code from the following repository:
   1. All source repositories can be installed automatically by running the following line in anaconda: . Packages can also be manually installed using Python pip.
   2. Test the SpikerNet distribution by running the following line of code: This runs a toy video game and tests SpikerNet’s RL system and associated packages.
4. Define the algorithm reward function. Reward functions encode the goals of closed-loop control through mathematical expressions. In this implementation, we define a mean-square error (MSE) reward of the form:

where are the measured and target responses respectively. Responses can be any measurable bio signal, such as a peristimulus time histogram (PSTH) with bins, EEG activity with time points, etc. This MSE reward “penalizes” observed responses which diverge from the target as evidenced by relatively small reward values and “rewards” measured values closer to the target with higher reward values. Our implementation measured reward from observed and target firing rate density functions from PSTHs estimated with Bayesian adaptive regression splines(BARS)2.

**Critical:** The choice of reward function is critical to SpikerNet performance. Choosing a reward or target function that is outside of physiological realizability will cause poor algorithm performance.

1. Define action space and action space bounds. The action space represents the range of stimulus parameters that can be chosen by the algorithm. For an electrical stimulator, this might be the stimulator current, pulse widths, and frequencies. Action spaces can be continuous (ie stimulus currents of 100.5, 94.5, 123.4 ) or discrete (ie mapping actions to individual stimulation classes such as electrical stimulator monophasic or biphasic modes). Our implementation uses an INS stimulator with a mixed continuous-discrete action space of number of stimulation pulses (discrete) stimulation laser power (continuous), pulse width(continuous), and interstimulus intervals(continuous). The declaration of action spaces is performed using the OpenAI gym command Box2d with parameter upper and lower bounds as follows:

1. #Setup action bounds: numPulses – number of optical pulses, stim – Optical power (mW), PW – Pulse width(ms), tBetPulses – Interstimulus interval(ms). np – numpy python toolbox.

2. Paramslow = np.array([numPulsesLow,stimLow,PWLow,tBetPulsesLow])

3. Paramshigh = np.array([numPulsesHigh, stimHigh, PWHigh,tBetPulsesHigh])

4. action\_space = gym.spaces.Box(low,high,dtype=np.float16)

Actions are clamped to fall between the declared bounds during search.

**Critical:** The choice of stimulus bounds should be made with stimulus safety considerations in mind. Stimulus limits should be constrained to well below ablation levels to ensure that SpikerNet does not damage neural tissue during search phases. This is contingent to the stimulation paradigm. For INS, we constrain pulse energies to per pulse3–6.

1. Define the observation space. The observation space describes the mathematical space of the observed biosignals. For neural recordings, the observation space is generally continuous. Bounds of the observation space generally follow the properties of the biosignal of interest. For example, PSTH responses from cortex have firing rates that are strictly positive (lower bound 0) with upper bounds generally set above maximum realizable firing rates. EEG responses, alternatively, contain positive and negative voltage values with analyses generally performed on negative and positive peaks. As such, observation space bounds should be aligned with maximum positive and negative values. Declaration of the observation space is similar to action space declaration:

1. observation\_space = gym.spaces.Box(low = obsLow, high = obsHigh)

Our implementation

**Note:** The TD3 reinforcement learning algorithm is specified for continuous observation spaces only. A discrete observation space will require a different deep Q learning method, such as double deep-Q networks7.

**Key Resource Table**

|  |  |  |
| --- | --- | --- |
| **Resource or Reagent** | **Source** | **Identifier** |
| **Stimulation and Recording Devices** | | |
| Planar neural recording Array | Tucker-Davis Technologies (TDT) | ZIF2030-32 |
| Optical Stimulation Probe (if using INS) | Thor Labs | CFML22L10 |
| Concentric Bipolar Electrical Stimulation Probe (if using electrical stimulation) | Microprobes | CEA-200-SS |
| RZ-2 Bioamp Processor | TDT | RZ-2 |
| RX-7 Stimulator | TDT | RX-7 |
| **Computational Tools** | | |
| NVIDIA AI-ready graphics processing unit (GPU) | NVIDIA | GTX 10x, RTX 20x, RTX 30x, RTX 40x, RTX Titan |
| Intel or AMD AI ready processor | Intel/AMD | Intel i7, i9 AMD Ryzen |
| Python Distribution | Python Software Foundation | N/A |

**Acknowledgments**

Funding for this project was provided by (PI: ELB) and a Purdue Institute for Integrative Neuroscience Collaborative Training Grant (PI: BSC). The authors would like to thank Georgia L. Lawlor for discussions on the implementation of reinforcement learning. BSC is now affiliated with the Department of Neurological Surgery and the Wisconsin Institute for Translational Neuroengineering, University of Wisconsin-Madison, Madison WI, USA.

**References**

1. Coventry, B.S., and Bartlett, E.L. (2023). Closed-Loop Reinforcement Learning Based Deep Brain Stimulation Using SpikerNet: A Computational Model. In 11th International IEEE EMBS Conference on Neural Engineering, pp. 1–4. 10.1109/NER52421.2023.10123797.

2. Dimatteo, I., Genovese, C.R., and Kass, R.E. (2001). Bayesian curve‐fitting with free‐knot splines. Biometrika *88*, 1055–1071. 10.1093/biomet/88.4.1055.

3. Chernov, M.M., Chen, G., and Roe, A.W. (2014). Histological Assessment of Thermal Damage in the Brain Following Infrared Neural Stimulation. Brain Stimulation *7*, 476–482. 10.1016/j.brs.2014.01.006.

4. Goyal, V., Rajguru, S., Matic, A.I., Stock, S.R., and Richter, C.-P. (2012). Acute damage threshold for infrared neural stimulation of the cochlea: functional and histological evaluation. Anatomical record (Hoboken, N.J. : 2007) *295*, 1987–1999. 10.1002/ar.22583.

5. Wells, J.D., Thomsen, S., Whitaker, P., Jansen, E.D., Kao, C.C., Konrad, P.E., and Mahadevan-Jansen, A. (2007). Optically mediated nerve stimulation: Identification of injury thresholds. Lasers in Surgery and Medicine *39*, 513–526. 10.1002/lsm.20522.

6. Cayce, J.M., Wells, J.D., Malphrus, J.D., Kao, C., Thomsen, S., Tulipan, N.B., Konrad, P.E., Jansen, E.D., and Mahadevan-Jansen, A. (2015). Infrared neural stimulation of human spinal nerve roots in vivo. Neurophotonics *2*, 015007. 10.1117/1.NPh.2.1.015007.

7. van Hasselt, H., Guez, A., and Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. 10.48550/ARXIV.1509.06461.