A NeuroAI-derived Brain-Computer Interface Simulator

Joel Ye[1]
[1]Carnegie Mellon University

We are interested in simulating BCI decoder controllability by evaluating control metrics with deep network policies. We have trained simple control policies in a cursor control environment, and trained simple BCI decoders on these network states. We show that control is equivalent between carefully altered decoders after allowing the policies to tune on BCI-control trajectories, simulating the user adaptation process.

1. Project Overview and Motivation

Current brain-computer interface research is prohibitively bottlenecked by the expense and inaccessibility of real world experiments. A key challenge in simulating these experiments is to account for human learning and adaptation to different BCI decoders. This project's goal is to emulate the outcome of user adaptation to different decoder algorithms by substituting the human for a deep network controller, and substituting human adaptation with deep RL. The problem formulation remains as proposed.

2. Progress and Technical Work So Far [13 points]

We have set up a codebase based on stable baselines, modifying their implementation of PPO (MLP + LSTM) to allow for integration of "BCI decoders" as controllers instead of the agent's native controller. The protocol of interest is as follows:

- Train a "native control" policy.
- Produce a static dataset of policy hidden states and actions in an evaluation environment.
- Train a BCI decoder from this static dataset, and optionally perturb it.
- Fine-tune the native control policy to convergence, with actions now generated by the BCI decoder instead of the native readout.

The key challenge was to reason about how to implement this fine-tuning step. While it was clear that the environment actions should be taken by the BCI decoded actions, it was unclear whether we should use the native policy or BCI actions in the RL updates. Since this RL tuning is not meant to emulate human adaptation, only its outcome, we could plausibly use the log-probabilities of generating different BCI actions. This is undesirable, however, for two reasons. First, such tuning would be "privileged" relative to human learning, seemingly allowing freeform adaptation of policy representations to adjust BCI decoder outputs, which likely doesn't occur in the brain. Second, it would restrict evaluation to differentiable decoders.

The alternate strategy of using native policy actions was complicated due to the need for exploration. Most every RL strategy requires stochastic behavior in learning to discover which actions are better. However, this stochasticity is nearly universally introduced once the decoder produces a prediction. The

Corresponding author(s): joelye9@gmail.com

issue then is that the network's internal state, which the BCI readout depends on, is deterministic, so environment transitions remain deterministic. On the other hand, if we solely noise the BCI action (as in DDPG), this noise is not reflected in the native-policy RL updates.

The solution here is to noise the network state instead of the action readouts. Many RL algorithms (PPO/SAC) require explicit log probabilities of sampled actions, so we cannot afford to push noise deep into the network, but we can afford to push it to the penultimate hidden state before the native policy's linear action readout. Doing this, we can have noise alter the trajectories induced by arbitrary BCI decoders, while still having an analytically tractable log-prob for RL updates through the native policy head. I deem the need to restrict BCI readout to the final hidden representation an acceptable constraint (mainly because there is no clear alternative).

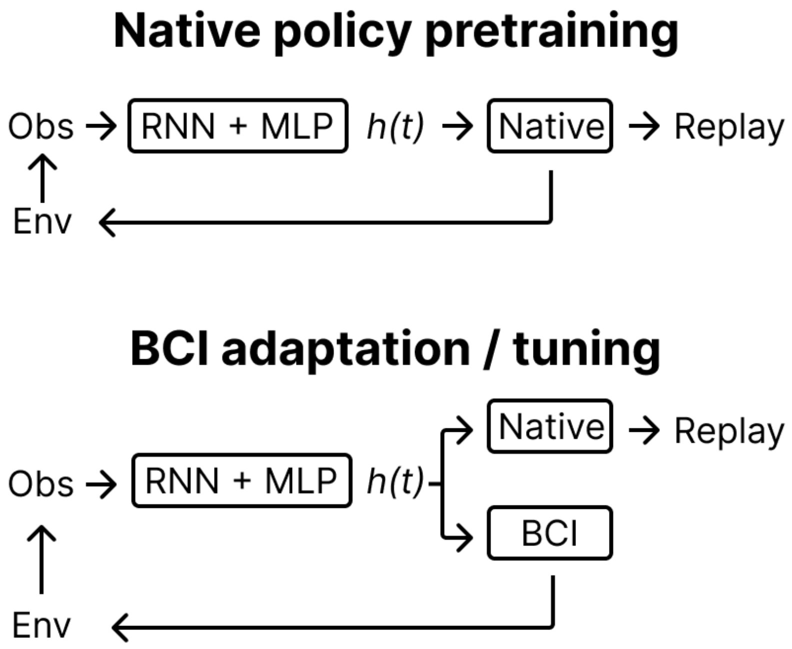
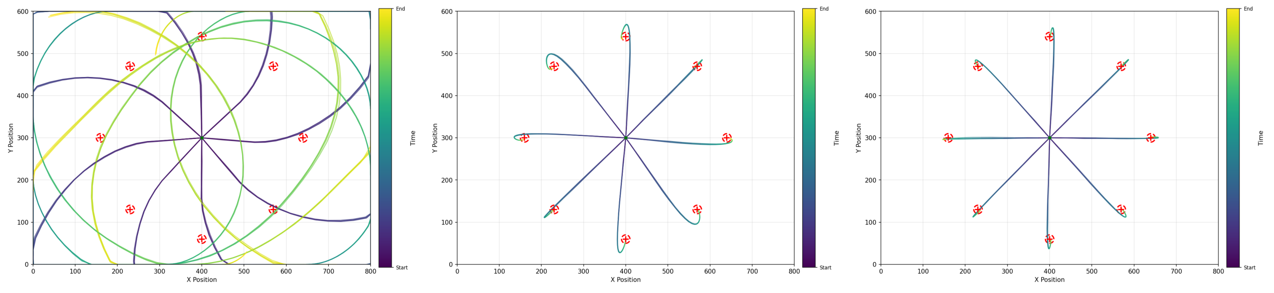


Figure 1: Environment actions are driven by native policy in pretraining but BCI in fine-tuning.

We test this in a 2D cursor control environment. Example trajectories are shown over tuning with a decoder that rotates the native readout by 90 degrees.



(a) Init (b) Halfway (c) Converged

Figure 2: Example control trajectories over tuning with a 90 degree rotated decoder.

We also include training curves showing the reward progression of 45, 90, 180 degree decoders, as well as a *random* decoder which *cannot* be learned due to a lack of guaranteed correlations between the native action space and BCI readout space.

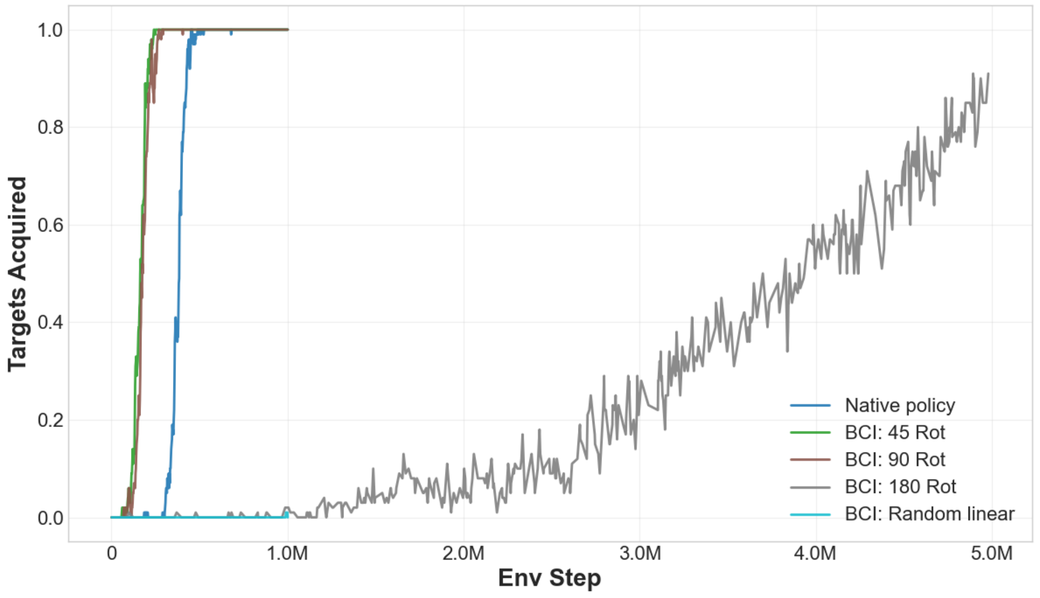


Figure 3: Tuning progress for 45, 90 degree rotations (trains quickly), 180 degree rotated decoders (very slow) and a random decoder (never nontrivial).

Takeaways box

It is possible to train deep RL agents to control BCI decoders conditioned on the agent's hidden states, through optimization of the agent's native action space. Decoders that rotate the agent's native readout in a 2D environment can be compensated for with RL to restore full performance.

3. Datasets, Environments, and Resources [2 points]

- We currently use simple, custom point-reaching environments with sensorimotor noise. This emulates 1-2D cursor control environments frequently used in BCI testing. The initial state spawns the cursor in the center of the environment. Goal states are randomly placed locations in training, and radially arranged in evaluation. Datasets for training decoders (mapping from agent state to agent behavior) are created from evaluation trajectories.
- All experiments to date have been CPU-only (seeing slowdowns on GPU, working to diagnose).

4. Updated Plan and Next Steps [5 points]

My hope is to complete "broader project derisking" for the remainder of this semester. This entails the following:

- Train policies in either Mujoco (via MuSim [1]) or MotorNet [2] (a simpler fallback), environments that create policies to actuate more detailed plants rather than kinematics directly. This control scheme is necessary to produce policies that resemble motor cortex [4].
- Demonstrate control quality differs from static dataset cloning metrics. Ideally, train nonlinear decoders – either mechanically clamped linear decoders or deep network decoders I have worked with before [5], and demonstrate they can also be tuned for use, but control at convergence may not be as good as static metrics indicate.
- Stretch goal: Demonstrate both of the above together.

Once these are implemented, yes, there will be corresponding experiments to demonstrate each of these points. I expect light analysis in this proof of concept stage of the project. The writeup may be more in depth about methods and motivation.

Risk mitigation: In case the native action space RL doesn't work in these more complex settings, the project may pivot to focus more on understanding the mechanism of action of why it does work in the simpler setting we've achieved so far.

Next milestones:

- Create a metric suite to produce plots that control plots used in BCI cursor control [3].
- The above epochs.

5. Team Contributions

I (Joel) am responsible for all components except for BCI decoder implementation. Said implementation is being provided by a collaborator outside the class, Chris Ki.

6. Reflection and Feedback

The issue with native policy optimization was unexpectedly technically subtle, so it was pretty satisfying to make progress on. (Effectively stalled the project for a few weeks...)

I'd like for the scope for this project to exceed the class, so I decided to work on a fresh codebase, despite related work. This does make it seem, however, like I'm playing catch up with existing projects in the area that are not exactly on topic, but contain several existing results that I'd like to integrate, so it still feels little progress has been made relative to global knowledge.

The timeline for the project is unfortunately short. I'm thankful other homeworks were cancelled, at least, but even better if the horizon for the project were somehow longer. It's tricky to parallelize work even with only one other collaborator, in the early stages of a research codebase.

References

[1] Muhammad Noman Almani, John Lazzari, and Shreya Saxena. musim: A goal-driven framework for elucidating the neural control of movement through musculoskeletal modeling. *bioRxiv*, 2024. doi: 10.1101/2024.02.02.578628.

[2] Olivier Codol, Guillaume Hennequin, and Claudia Clopath. Motornet: A python toolbox for biologically detailed and differentiable motor control. *eLife*, 12:e88591, 2023. doi: 10.7554/eLife.88591.

[3] Ken-Fu Liang and Jonathan C. Kao. A reinforcement learning based software simulator for motor brain-computer interfaces. *bioRxiv*, November 2024. doi: 10.1101/2024.11.25.625180. URL https://www.biorxiv.org/content/10.1101/2024.11.25.625180v1. bioRxiv preprint.

[4] David Sussillo, Mark M. Churchland, Matthew T. Kaufman, and Krishna V. Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature Neuroscience*, 18(7): 1025–1033, 2015. doi: 10.1038/nn.4042.

[5] Joel Ye, Fabio Rizzoglio, Xuan Ma, Adam Smoulder, Hongwei Mao, Gary H Blumenthal, William Hockeimer, Nicolas Guazzelli Kunigk, Dalton D. Moore, Patrick J. Marino, Raeed H. Chowdhury, J. Patrick Mayo, Aaron Batista, Steven Chase, Michael L Boninger, Charles M. Greenspon, Andrew B. Schwartz, Nicholas G. Hatsopoulos, Lee E. Miller, Kristofer Bouchard, Jennifer L Collinger, Leila Wehbe, and Robert Gaunt. A generalist intracortical motor decoder. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=utXSSdD9mt.

Appendices