Author Contributions Checklist Form

This form documents the artifacts associated with the article (i.e., the data and code supporting the computational findings) and describes how to reproduce the findings.

# Part 1: Data

This paper **does not** involve analysis of external data (i.e., no data are used or the only data are generated by the authors via simulation in their code).

I certify that the author(s) of the manuscript have legitimate access to and permission to use the data used in this manuscript.

## Abstract

Each dataset contains EEG signals (simulated or real), stimulus-occurring indicators (1-12), stimulus-type indicators (binary). For dataset solved by existing ML methods, EEG signals are extracted by the fixed response window length such that it has the dimension of (sample size, channel dimension, response window length).

## Availability

Data **are** publicly available

Data **cannot be made** publicly available

If the data are publicly available, see the *Publicly available data* section. Otherwise, see the *Non-publicly available dat*a section, below.

### Publicly available data

Data are available online at:

Data are available as part of the paper’s supplementary material.

Data are publicly available by request, following the process described here:

Data are or will be made available through some other mechanism, described here:

### Non-publicly available data

Discussion of lack of publicly available data:

For simulation study, the entire data files are too large to save and upload. However, the python code under simulation folder can generate the exact dataset given the same seed number.

The participants in the study did not consent to release their data in public directly, even for the de-identified version. However, the de-identified data are available upon request for other researchers. According to the guideline from the University of Michigan medical school office of research, an outgoing data user agreement (DUA) needs to be created for external sharing of any individual-level clinical data, regardless of de-identified or not.

## Description

### File format(s)

CSV or other plain text:

Software-specific binary format (.Rda, Python pickle, etc.):

Standardized binary format (e.g., netCDF, HDF5, etc.):

Other (described here):

### Data dictionary

Provided by the authors in the following file(s):

Data file(s) is (are) self-describiing (e.g., netCDF files)

Available at the following URL:

The data dictionary (as well as experimental design) can be found in this paper “A plug-and-play brain-computer interface to operate commercial assistive technology” via the link <https://doi.org/10.3109/17483107.2013.785036>.

### Additional information (optional)

We provide the step-by-step data pre-processing for your convenience.

First, we applied a notch filter at 60Hz to remove the power line noise and a bandpass filter between 0.5 and 6 Hz to all 16 channels and then down-sampled raw signals with a decimation factor of eight.

Second, we truncated each (character-specific) super-sequence into 15 sequence segments, where each sequence segment contained 12 consecutive stimuli and subsequent signals of 20 time points to record the entire ERP response to the last stimulus within the single sequence. Each sequence segment contained 2,500 ms with 80 sampling points.

# Part 2: Code

## Abstract

MATLAB codes are used to extract EEG signals from raw data file, perform data pre-processing, and perform swLDA anlaysis.

Python codes are used to generate simulated datasets, implement the xDAWN filter, fit BSM-related methods, MDWM, predict with all methods, and perform sensitivity check.

R codes are used to generate tabular and graphic outputs including parameter estimates and prediction accuracy.

## Description

### Code format(s)

Script files

R  Python  Matlab

Other:

Package

R  Python  MATLAB toolbox

Other:

Reproducible report

R Markdown  Jupyter notebook

Other:

Shell script

Other (described here):

### **Supporting software requirements**

Version of primary software used

Python version 3.7

R Version 4.2.2

MATLAB R2022b

Libraries and dependencies used by the code

Python: numpy, matplotlib, scipy, sklearn, pyriemann, numpyro, jax, json

MATLAB: train\_SWLDAmatlab.m, stepwisefit.m

R: ggplot2, gridExtra, R.matlab, dplyr, RJSONIO

### Supporting system/hardware requirements (optional)

Apple macOS Sequoia (Version 15.3) and Linux environments (institutional server) were used for preforming all the analyses above.

PyCharm 2024.02.01 to locally run Python code.

### Parallelization used

No parallel code used

Multi-core parallelization on a single machine/node

Number of cores used:

Multi-machine/multi-node parallelization

Number of nodes and cores used:

### License

MIT License (default)

BSD

GPL v3.0

Creative Commons

Other (described here):

### Additional information (optional)

Sample script for slurm code is attached to answer the parallelization question.

# Part 3: Reproducibility workflow

## Scope

The provided workflow reproduces:

Any numbers provided in text in the paper

The computational method(s) presented in the paper (i.e., code is provided that implements the method(s))

All tables and figures in the paper

Selected tables and figures in the paper, as explained and justified here:

## Workflow details

### Location

The workflow is available:

As part of the paper’s supplementary material

In this Git repository:

Other:

### Format(s)

Single master code file

Wrapper (shell) script(s)

Self-contained R Markdown file, Jupyter notebook, or other literate programming approach

Text file (e.g., a readme-style file) that documents workflow

Makefile

Other (more detail in 'Instructions' below)

### Instructions

Please refer to “README\_MATLAB.txt”, “README\_Python\_real.txt”, “README\_Python\_simulation.txt”, and “README\_R.txt” for first-hand instructions.

Expected run-time

Approximate time needed to reproduce the analyses on a standard desktop machine:

<1 minute

1-10 minutes

10-60 minutes

1-8 hours

>8 hours

Not feasible to run on a desktop machine, as described here:

### Additional documentation (optional)

For BSM method, it takes 2-3 hours to finish 9000 MCMC iterations (including 8000 burn-ins) for two chains.

BSM-Reference and prediction accuracy takes a few minutes.

# Notes (optional)