py_neuromodulation: Signal processing and decoding for neural electrophysiological recordings

Timon Merk[1¶], **Antonio Brotons**[1], **Samed R. Vossberg**[1], **Richard M. Köhler**[1], **Thomas S. Binns**[1,2,3], **Ahmed Tarek Kamel Abdalfatah**[1], **Alessia Cavallo**[1,2], **Elisa Garulli**[4], **Ashley Walton**[5], **Jojo Vanhoecke**[1,2], **R. Mark Richardson**[5], and **Wolf-Julian Neumann**[1,2,3]

1 Movement Disorders Unit, Charité - Universitätsmedizin Berlin, Berlin, Germany **2** Bernstein Center for Computational Neuroscience Berlin, Berlin, Germany **3** Einstein Center for Neurosciences Berlin, Berlin, Germany **4** Department of Neurology with Experimental Neurology, Charité – Universitätsmedizin Berlin, Berlin, Germany **5** Department of Neurosurgery, Massachusetts General Hospital, Harvard Medical School, Boston, Massachusetts, USA ¶ Corresponding author

DOI: 10.21105/joss.08258

Software

- Review
- Repository
- Archive

Editor: Julia Romanowska

Reviewers:

- @finsberg
- @danielmk

Submitted: 31 March 2025
Published: 06 August 2025

Summary

Invasive brain signal decoding can revolutionize the clinical utility of neurotechnological therapies. Potential signal sources stem from electroencephalography (EEG), electrocorticography (ECoG) and local field potentials (LFP) recorded from deep brain stimulation (DBS) electrodes. The application of machine learning methods to these signals requires pre-processing and feature extraction – complex analyses, often lacking standardization and clear documentation. Here, we introduce py_neuromodulation, a toolbox designed for standardized signal processing, feature extraction, and decoding of electrophysiological data. All parameters are explicitly defined in dedicated settings and channel parameterization files. The framework processes both data streamed in real-time and from recordings on disk using the same pipeline. Additionally, a browser-based graphical user interface (GUI) enables intuitive usage and visualization of the processing pipeline without requiring any code modification.
By introducing `py_neuromodulation`, we aim to simplify and standardize the analysis of electrophysiological recordings, facilitating reproducibility and accessibility in the field of brain signal decoding. Our tool bridges the fields of neuroscience and neural engineering, providing machine learning and neurotechnology researchers with reproducible methods for the development of generalizable machine learning algorithms.

Introduction

Recently, deep learning foundation models were presented that showed a performance leap on many clinical relevant downstream tasks (Li et al., 2025; Yuan et al., 2024). Neural signal processing for machine learning and medical device engineering is a critical step for the advancement of the fields of neural decoding and brain computer interfaces. We, therefore, developed a neural signal processing, feature estimation and decoding toolbox py_neuromodulation, that includes several established and standardized pipelines that simplify processing of neural recordings.
Analysis of electrophysiological recordings is commonly started by multiple preprocessing steps: resampling, notch-filtering, normalization, re-referencing (subtraction of neighboring channels), filtering (lowpass, highpass, bandpass), or artifact rejection (Cohen, 2014). Next, the signals or features of interest are computed. Most commonly, signals are analyzed in the spectral domain. Here, different methods can be utilized, including Fast Fourier Transform, Welch transform, bandpass filtering, and others. In addition to spectral features, different feature

modalities such as waveform-shape (Cole & Voytek, 2017), periodic and aperiodic spectral parametrization (Donoghue et al., 2020), burst features (Tinkhauser et al., 2018), and others can be computed.

State of the field

Several toolboxes for pre-processing, feature estimation and decoding for neural recordings exist in the Python programming language (*Openlists/ElectrophysiologySoftware*, 2025). Notably mne-python is a general purpose toolbox for processing and analyzing MEG and EEG data [GramfortEtAl2013a]. Braindecode is an deep learning toolbox for end-to-end neural decoding and benchmarking (Schirrmeister et al., 2017). OSL-Ephys is a MEG and EEG toolbox for automatized preprocessing and source reconstruction (Es et al., 2025). Several other tools implement computation of specific individual neural characteristics, such as spectral parametrization, connectivity or nonlinear dynamics. The previous mentioned neural signal processing and decoding tools are primarily optimized for analysis of non-invasive data. While many of the pre-processing and signal characterization steps remain similar in non-invasive recordings, there is a specific need to derive optimal analysis parameters for invasive recordings, for example from deep brain stimulation or electrocorticographical electrodes. Invasive local field potential recordings contain commonly signals of low number of channels, which results in signal processing without source reconstruction. Second, specific features are shown to be particularly relevant for subcortical recordings, such as bursts in different frequency bands (Merk, Peterson, Lipski, et al., 2022). Therefore, there is a lack of analysis tools for invasive recordings that enable explorative neural data analysis and machine learning neural decoding based on local field potential recordings from either deep brain stimulation or stereotactic EEG electrodes.

Statement of need

We aim to simply the pre-processing and feature-estimation using those invasive modalities. To simply this analysis, `py_neuromodulation` utilizes two parametrization files: *settings.yaml* and *channels.tsv*, that include all required steps for reproducibility. It allows for a streamlined pipeline of pre-processing, feature estimation and decoding. This enables the assessment of performance contributions of various pre-processing steps in combination with feature estimation modalities and machine learning model architectures for different neural signal processing and decoding applications. Offline analysis often includes whole-recording normalization, non-causal filtering, or feature computation with test-set data leakage (Merk, Peterson, Köhler, et al., 2022), which limits real-time compatibility. In `py_neuromodulation`, the data-source is represented as a Python generator, and can be interchangeable for online and offline processing, meaning that there is no difference in handling data streamed directly from a brain implant, or stored from an archival dataset. This enables training of generalizable models that are directly applicable to online applications, using exactly the same algorithms. Finally, electrophysiological research entails multiple technical and conceptual hurdles that pose substantial entry barriers for scientists without software development training. By providing an intuitive GUI for parametrization and raw-data and feature visualization, we aim to lower the domain knowledge requirements for entering this field. `py_neuromodulation` was already used in several scientific projects with various applications: ECoG and DBS-LFP movement decoding (Cavallo et al., 2025; Merk, Peterson, Lipski, et al., 2022), movement intention classification (Köhler et al., 2024), seizure classification (Merk et al., 2023), emotional valence decoding (Merk et al., 2023), gait symptom estimation in rodents (ELGarulli, 2025), amongst others.

Parametrization

A data-source can be specified to be either a two-dimensional array (channels x time) read through mne.io.read_raw supported recording standards or a LabStreamingLayer stream based on mne-lsl (Figure 1). If the data is passed as a simple numpy array, the sampling rate needs to be additionally specified. The parametrization **settings** and **channels** can be passed as additional optional parameters or automatically inferred. All parametrization is described in the GitHub documentation in detail Usage page. The main parameters are the *sampling_rate_features_hz* and *segment_length_ms*. The feature sampling rate defines in which interval features are computed, and the segment length defines the signal duration that is used for computation of each feature. For pre-processing we defined several methods that can be reordered freely: notch_filtering, raw_resampling, re_referencing (channel-specific common-average or individual bipolar re-referencing), raw_normalization and preprocessing_filter (lowpass, highpass, bandpass, bandstop). Additionally, a modifiable list of frequency ranges is passed, which are used for spectral feature computation. A list of currently implemented features is the following: raw_hjorth, return_raw, bandpass_filter, stft, fft, welch, sharpwave_analysis, Hjorth, line length, bispectrum, fooof, bursts, linelength, coherence, nolds, mne_connectivity and bispectrum.

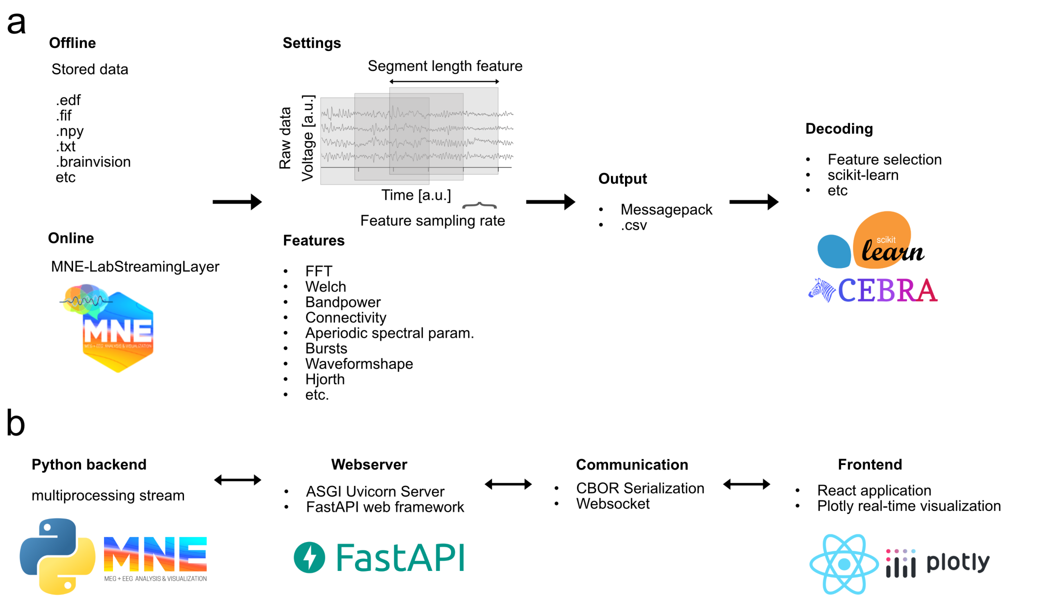
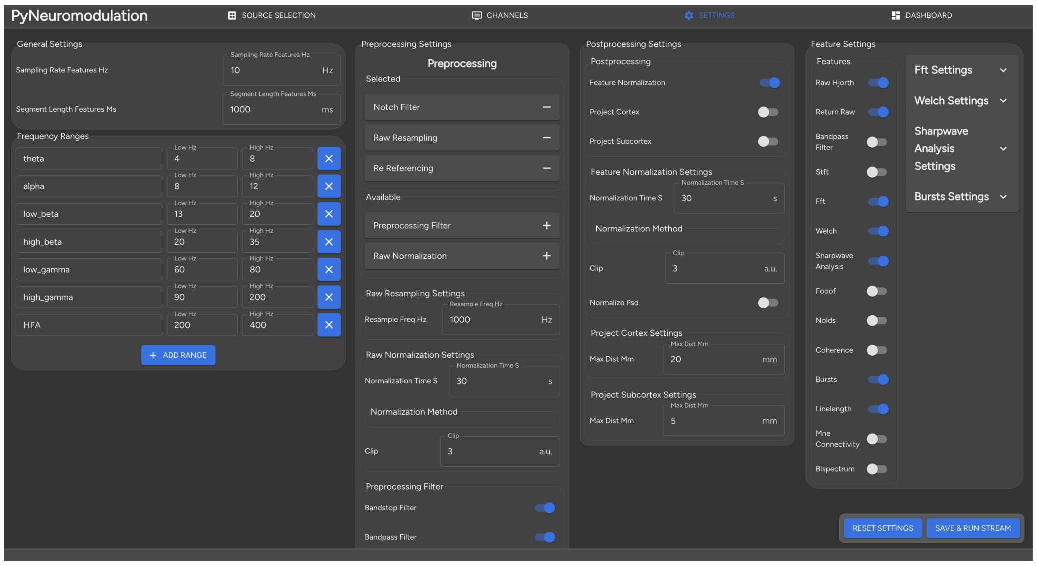


Figure 1: py-neuromodulation pre-processing and features estimation pipeline. a) Different offline-stored or online-streamed datasets can be used for feature estimation of different modalities and subsequent machine-learning neural decoding. b) py-neuromodulation backend and frontend setup. All backend routines are python-based and linked through a webserver application to a frontend that hosts parametrization and visualization.

Importantly, pre-processing and feature estimation can directly affect performances of decoding models. We added a **decode** module that can be parametrized with a scikit-learn compatible model, different cross-validation strategies, performance metrices, and dimensionality reduction methods, such that the computed features can be directly used to evaluate decoding performance within the same pipeline. A pre-trained model can also be passed to directly generate decoding outputs.

Graphical User Interface (GUI)

Finally, to simplify electrophysiological analysis and decoding, we added a React-based frontend application, which runs within an ASGI (Asynchronous Server Gateway Interface) uvicorn server and communicates through FastAPI. Through this frontend, the same settings as in the Python backend can be modified (Figure 2), a stream can then be selected from an offline file or via LabStreamingLayer, and raw data and features can be visualized in real time. Both backend and frontend processing makes use of the same **settings** and **channels** parametrization. Therefore, reproducing analyses including pre- and postprocessing, and feature estimation, is enabled by making use of the same underlying parametrization files.



General Settings

Sampling Rate Features Hz	10 Hz
Segment Length Features Ms	1000 ms

Frequency Ranges

	Low Hz	High Hz
theta	4	8
alpha	8	12
low_beta	13	20
high_beta	20	35
low_gamma	60	80
high_gamma	90	200
HFA	200	400

Preprocessing Settings

Preprocessing

Selected: Notch Filter; Raw Resampling; Re Referencing

Available: Preprocessing Filter; Raw Normalization

Raw Resampling Settings: Resample Freq Hz 1000 Hz

Raw Normalization Settings: Normalization Time S 30 s; Normalization Method; Clip 3 a.u.

Preprocessing Filter: Bandstop Filter ●; Bandpass Filter ●

Postprocessing Settings

Postprocessing: Feature Normalization ●; Project Cortex ○; Project Subcortex ○

Feature Normalization Settings: Normalization Time S 30 s; Normalization Method; Clip 3 a.u.; Normalize Psd ○

Project Cortex Settings: Max Dist Mm 20 mm

Project Subcortex Settings: Max Dist Mm 5 mm

Feature Settings

Features: Raw Hjorth ●; Return Raw ●; Bandpass Filter ○; Stft ○; Fft ●; Welch ●; Sharpwave Analysis ●; Fooof ○; Nolds ○; Coherence ○; Bursts ●; Linelength ●; Mne Connectivity ○; Bispectrum ○

Fft Settings; Welch Settings; Sharpwave Analysis Settings; Bursts Settings

Figure 2: Frontend parametrization page representing `settings.yaml` configurations for pre-processing, feature estimation and post-processing.

Conclusion

In summary, `py_neuromodulation` provides a comprehensive, standardized framework for electrophysiological signal processing and neural decoding, addressing existing limitations in reproducibility and parameter documentation. Its unified pipeline supports both real-time and offline analyses, accompanied by an intuitive graphical interface to lower technical barriers in neural research. The successful use of `py_neuromodulation` across diverse applications highlights its potential as a broadly applicable tool for the analysis and machine-learning based decoding of electrophysiological data.

Acknowledgements

The study was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 424778381 and a US-German Collaborative Research in Computational Neuroscience (CRCNS) grant from the German Federal Ministry for Research and Education and NIH (R01NS110424). WJN received funding from the European Union (ERC, ReinforceBG, project 101077060).

References

Cavallo, A., Köhler, R. M., Busch, J. L., Habets, J. G. V., Merk, T., Zvarova, P., Vanhoecke, J., Marcelino, A. L. de A., Darcy, N., Herz, D. M., Schneider, G.-H., Faust, K., Yttri, E., Cagnan, H., Kühn, A. A., & Neumann, W.-J. (2025). Reinforcement of movement speed through speed-selective deep brain stimulation in Parkinson's disease. *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation*, *18*(1), 529. https://doi.org/10.1016/j.brs.2024.12.917

Cohen, M. X. (2014). *Analyzing Neural Time Series Data: Theory and Practice*. The MIT Press. https://doi.org/10.7551/mitpress/9609.001.0001

Cole, S. R., & Voytek, B. (2017). Brain Oscillations and the Importance of Waveform Shape. *Trends in Cognitive Sciences*, *21*(2), 137–149. https://doi.org/10.1016/j.tics.2016.12.008

Donoghue, T., Haller, M., Peterson, E. J., Varma, P., Sebastian, P., Gao, R., Noto, T., Lara, A. H., Wallis, J. D., Knight, R. T., Shestyuk, A., & Voytek, B. (2020). Parameterizing neural power spectra into periodic and aperiodic components. *Nature Neuroscience*, *23*(12), 1655–1665. https://doi.org/10.1038/s41593-020-00744-x

ELGarulli. (2025). *ELGarulli/neurokin*. https://github.com/ELGarulli/neurokin

Es, M. W. J. van, Gohil, C., Quinn, A. J., & Woolrich, M. W. (2025). Osl-ephys: A Python toolbox for the analysis of electrophysiology data. *Frontiers in Neuroscience*, *Volume 19 - 2025*. https://doi.org/10.3389/fnins.2025.1522675

Köhler, R. M., Binns, T. S., Merk, T., Zhu, G., Yin, Z., Zhao, B., Chikermane, M., Vanhoecke, J., Busch, J. L., Habets, J. G. V., Faust, K., Schneider, G.-H., Cavallo, A., Haufe, S., Zhang, J., Kühn, A. A., Haynes, J.-D., & Neumann, W.-J. (2024). Dopamine and deep brain stimulation accelerate the neural dynamics of volitional action in Parkinson's disease. *Brain*, *147*(10), 3358–3369. https://doi.org/10.1093/brain/awae219

Li, J., Chen, X., Shen, F., Chen, J., Liu, Y., Zhang, D., Yuan, Z., Zhao, F., Li, M., & Yang, Y. (2025). *Deep Learning-Powered Electrical Brain Signals Analysis: Advancing Neurological Diagnostics*. arXiv. https://doi.org/10.48550/arXiv.2502.17213

Merk, T., Köhler, R., Peterson, V., Lyra, L., Vanhoecke, J., Chikermane, M., Binns, T., Li, N., Walton, A., Bush, A., Sisterson, N., Busch, J., Lofredi, R., Habets, J., Huebl, J., Zhu, G., Yin, Z., Zhao, B., Merkl, A., … Neumann, W.-J. (2023). *Invasive neurophysiology and whole brain connectomics for neural decoding in patients with brain implants*. Research Square. https://doi.org/10.21203/rs.3.rs-3212709/v1

Merk, T., Peterson, V., Köhler, R., Haufe, S., Richardson, R. M., & Neumann, W.-J. (2022). Machine learning based brain signal decoding for intelligent adaptive deep brain stimulation. *Experimental Neurology*, *351*, 113993. https://doi.org/10.1016/j.expneurol.2022.113993

Merk, T., Peterson, V., Lipski, W. J., Blankertz, B., Turner, R. S., Li, N., Horn, A., Richardson, R. M., & Neumann, W.-J. (2022). Electrocorticography is superior to subthalamic local field potentials for movement decoding in Parkinson's disease. *eLife*, *11*, e75126. https://doi.org/10.7554/eLife.75126

Openlists/ElectrophysiologySoftware. (2025). OpenLists. https://github.com/openlists/ElectrophysiologySoftware

Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., & Ball, T. (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*. https://doi.org/10.1002/hbm.23730

Tinkhauser, G., Torrecillos, F., Duclos, Y., Tan, H., Pogosyan, A., Fischer, P., Carron, R., Welter, M. L., Karachi, C., Vandenberghe, W., Nuttin, B., Witjas, T., Régis, J., Azulay,

J. P., Eusebio, A., & Brown, P. (2018). Beta burst coupling across the motor circuit in Parkinson's disease. *Neurobiology of Disease*. https://doi.org/10.1016/j.nbd.2018.06.007

Yuan, Z., Shen, F., Li, M., Yu, Y., Tan, C., & Yang, Y. (2024). *BrainWave: A Brain Signal Foundation Model for Clinical Applications*. arXiv. https://doi.org/10.48550/arXiv.2402.10251