The Journal of Open Source Software

CANNs: Continuous Attractor Neural Networks Toolkit with ASA for Attractor Structure Analysis

Sichao He[1], **Aiersi Tuerhong**[2], **Shangjun She**[3], **Tianhao Chu**[3], **Yuling Wu**[1], **Junfeng Zuo**[1], **and Si Wu**[1, 3, 4, 5, 6]

1 Peking-Tsinghua Center for Life Sciences, Academy for Advanced Interdisciplinary Studies, Peking University, Beijing, China **2** College of Mathematics and Statistics, Chongqing University, Chongqing, China **3** School of Psychological and Cognitive Sciences, Peking University, Beijing, China **4** School of Psychology and Cognitive Sciences, Peking University, Beijing, China **5** PKU-IDG/McGovern Institute for Brain Research, Peking University, Beijing, China **6** Center of Quantitative Biology, Peking University, Beijing, China

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

CANNs (Continuous Attractor Neural Networks toolkit) is a Python library built on BrainPy, a powerful framework for brain dynamics programming. It streamlines experimentation with continuous attractor neural networks and related brain-inspired models. The library delivers ready-to-use models, task generators, analysis tools, and pipelines—enabling neuroscience and AI researchers to move quickly from ideas to reproducible simulations.

Statement of need

Continuous Attractor Neural Networks (CANNs) provide a theoretical framework for understanding how the brain encodes continuous variables through stable neural activity patterns. Despite their importance in computational neuroscience, CANN research suffers from fragmentation: researchers implement models from scratch using incompatible codebases, creating reproducibility barriers and steep learning curves.

CANNs addresses this gap by providing a unified Python toolkit built on BrainPy (Wang et al. 2023). It delivers standardized CANN implementations, integrated task generation and analysis pipelines, and high-performance computation via JAX JIT compilation with optional Rust acceleration.

Software design

The CANNs library follows a modular architecture (Figure 1) with five independent modules: Models, Tasks, Analyzers, Trainers, and Pipeline. This separation ensures maintainability and extensibility through abstract base classes that define standard interfaces.

The library supports CANN modeling and simulation, experimental data analysis, brain-inspired learning, and automated parameter sweeps. All models inherit from BrainPy's `DynamicalSystem` base class (Wang et al. 2023), leveraging JAX JIT compilation for GPU/TPU acceleration. A companion Rust library provides optional accelerated backends for performance-critical operations.

The Journal of Open Source Software

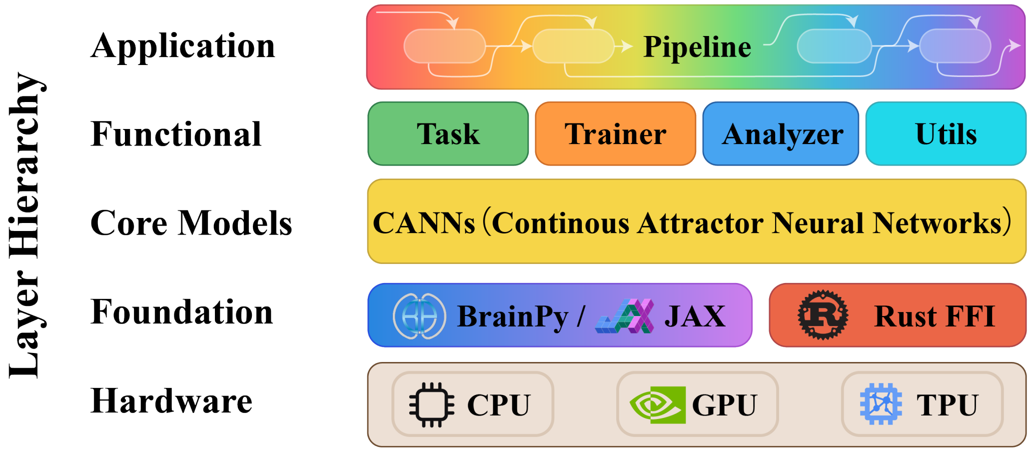

Figure 1: Layer hierarchy of the CANNs library showing five levels: Application (Pipeline orchestration), Functional (Task, Trainer, Analyzer, Utils modules), Core Models (CANN implementations), Foundation (BrainPy/JAX and Rust FFI backends), and Hardware (CPU/GPU/TPU support).

Related Works

While general-purpose neural network simulators like NEST (Gewaltig and Diesmann 2007), Brian 2 (Stimberg, Brette, and Goodman 2019), and NEURON (Hines and Carnevale 1997) exist, they lack specialized support for continuous attractor dynamics. Existing CANN implementations remain fragmented, lab-specific codebases without standardized APIs or comprehensive tooling.

CANNs builds upon BrainPy (Wang et al. 2023), a modern brain dynamics framework leveraging JAX (Bradbury et al. 2018) for JIT compilation and GPU/TPU acceleration. CANNs extends BrainPy with CANN-specific abstractions: standardized model implementations, task-generation APIs, analysis pipelines, and optional Rust-accelerated backends for performance-critical operations.

AI usage disclosure

AI-assisted tools were used for code quality reviews and documentation writing. All core library code was written by human developers, and AI-generated content was reviewed and validated by the authors.

Acknowledgements

We acknowledge the BrainPy development team for providing the foundational framework upon which this library is built, and the broader open-source community for tools and libraries that enabled this work.

References

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, et al. 2018. *JAX: Composable Transformations of*

Python+NumPy Programs (version 0.3.13). http://github.com/jax-ml/jax.

Gewaltig, Marc-Oliver, and Markus Diesmann. 2007. "NEST (Neural Simulation Tool)." *Scholarpedia* 2 (4):1430.

Hines, Michael L, and Nicholas T Carnevale. 1997. "The Neuron Simulation Environment." *Neural Computation* 9 (6). MIT Press:1179–1209.

Stimberg, Marcel, Romain Brette, and Dan FM Goodman. 2019. "Brian 2, an Intuitive and Efficient Neural Simulator." *Elife* 8. eLife Sciences Publications, Ltd:e47314.

Wang, Chaoming, Tianqiu Zhang, Xiaoyu Chen, Sichao He, Shangyang Li, and Si Wu. 2023. "BrainPy, a Flexible, Integrative, Efficient, and Extensible Framework for General-Purpose Brain Dynamics Programming." Edited by Marcel Stimberg. *eLife* 12 (December). eLife Sciences Publications, Ltd:e86365. https://doi.org/10.7554/eLife.86365.