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—such as spatial position, head direction, and movement trajectories—through stable neural activity patterns. These models explain key phenomena in hippocampal place cells (O'Keefe and Dostrovsky 1971), entorhinal grid cells (Hafting et al. 2005), and head direction systems (Taube, Muller, and Ranck 1990). Despite their importance, CANN research suffers from fragmentation: researchers implement models from scratch, use incompatible codebases, and face significant reproducibility barriers. This lack of standardization slows progress and creates steep learning curves for newcomers.

CANNs addresses this gap by providing a unified Python toolkit built on BrainPy (Wang et al. 2023). It delivers: (1) standardized implementations of CANN and related brain-inspired models, including canonical frameworks (Wu-Amari-Wong (Amari 1977; Wu, Hamaguchi, and Amari 2008), adaptation-augmented CANNs (Mi et al. 2014; Li, Chu, and Wu 2025), grid cell networks (Burak and Fiete 2009)), alongside additional attractor architectures; (2) integrated task generation, simulation, and analysis pipelines; and (3) high-performance computation via JAX JIT compilation and optional Rust acceleration. By standardizing workflows—analogous to Hugging Face Transformers in deep learning—this library accelerates reproducible research and lowers barriers for computational neuroscientists, AI engineers, and students exploring attractor dynamics.

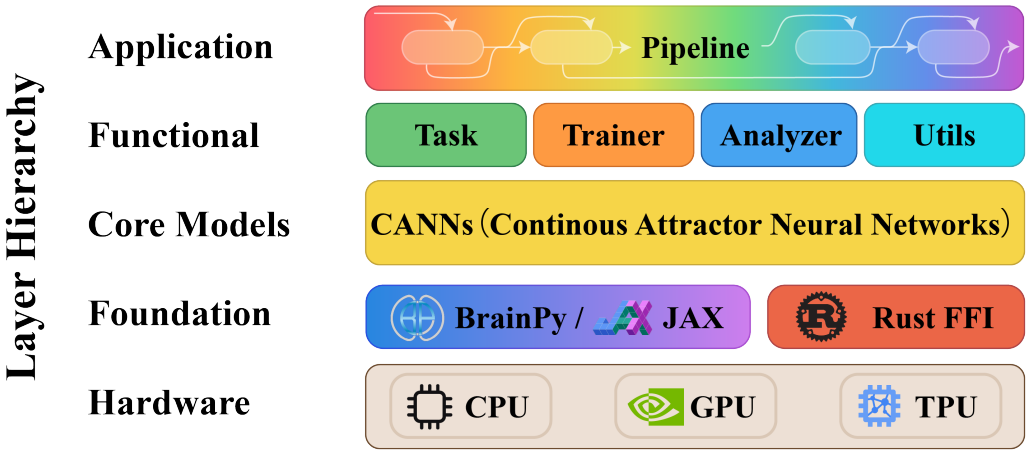

Figure 1: Layer hierarchy of the CANNs library showing five levels: Application (Pipeline orchestration), Functional (Task, Trainer, Analyzer, Utils modules), Core Models (CANN implementations), Foundation (BrainPy/JAX and Rust FFI backends), and Hardware (CPU/GPU/TPU support).

Software design

The CANNs library follows a modular architecture (Figure 1) guided by two core principles: **separation of concerns** and **extensibility through base classes**. The design separates functional responsibilities into five independent modules: (1) **Models** (`canns.models`) define neural network dynamics; (2) **Tasks** (`canns.task`) generate experimental paradigms and input data; (3) **Analyzers** (`canns.analyzer`) provide visualization and analysis tools; (4) **Trainers** (`canns.trainer`) implement learning rules for brain-inspired models; and (5) **Pipeline** (`canns.pipeline`) orchestrates complete experimental workflows.

Each module focuses on a single responsibility—models don't generate input data, tasks don't analyze results, and analyzers don't modify parameters. This separation ensures maintainability, testability, and extensibility. All major components inherit from abstract base classes (`BasicModel`, `BrainInspiredModel`, `Trainer`) that define standard interfaces, enabling users to create custom implementations that seamlessly integrate with the built-in ecosystem.

The library supports four distinct research workflows: (1) CANN modeling and simulation for studying attractor dynamics; (2) data analysis for processing experimental neural recordings; (3) brain-inspired learning with biologically plausible plasticity rules (Hebbian, STDP, BCM); and (4) end-to-end pipelines for automated parameter sweeps and reproducible experiments. All models inherit from BrainPy's `DynamicalSystem` base class (Wang et al. 2023), leveraging JAX's JIT compilation for GPU/TPU acceleration while maintaining simple Python APIs. For operations where Python overhead is significant, the companion `canns-lib` Rust library provides optional accelerated backends—notably achieving 400× speedup for spatial navigation tasks and 1.13-1.82× speedup for topological data analysis—without requiring code structure changes.

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

Amari, Shun-ichi. 1977. "Dynamics of Pattern Formation in Lateral-Inhibition Type Neural Fields." *Biological Cybernetics* 27 (2). Springer:77–87. https://doi.org/10.1007/BF00337259.

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.

Burak, Yoram, and Ila R Fiete. 2009. "Accurate Path Integration in Continuous Attractor Network Models of Grid Cells." *PLoS Computational Biology* 5 (2). Public Library of Science San Francisco, USA:e1000291.

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

Hafting, Torkel, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. 2005. "Microstructure of a Spatial Map in the Entorhinal Cortex." *Nature* 436 (7052). Nature Publishing Group UK London:801–6. https://doi.org/10.1038/nature03721.

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

Li, Yujun, Tianhao Chu, and Si Wu. 2025. "Dynamics of Continuous Attractor Neural Networks with Spike Frequency Adaptation." *Neural Computation* 37 (6). MIT Press 255 Main Street, 9th Floor, Cambridge, Massachusetts 02142, USA ...:1057–1101. https://doi.org/10.1162/neco_a_01757.

Mi, Yuanyuan, CC Fung, KY Wong, and Si Wu. 2014. "Spike Frequency Adaptation Implements Anticipative Tracking in Continuous Attractor Neural Networks." *Advances in Neural Information Processing Systems* 27.

O'Keefe, John, and Jonathan Dostrovsky. 1971. "The Hippocampus as a Spatial Map: Preliminary Evidence from Unit Activity in the Freely-Moving Rat." *Brain Research*. Elsevier Science. https://doi.org/10.1016/0006-8993(71)90358-1.

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

Taube, Jeffrey S, Robert U Muller, and James B Ranck. 1990. "Head-Direction Cells Recorded from the Postsubiculum in Freely Moving Rats. I. Description and Quantitative Analysis." *Journal of Neuroscience* 10 (2). Society for Neuroscience:420–35. https://doi.org/10.1523/JNEUROSCI.10-02-00420.1990.

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.

Wu, Si, Kosuke Hamaguchi, and Shun-ichi Amari. 2008. "Dynamics and Computation of Continuous Attractors." *Neural Computation* 20 (4). MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ...:994–1025. https://doi.org/10.1162/neco.2008.10-06-378.