VSAX: A JAX Library for Vector Symbolic Architectures

Vasanth Sarathy VASANTH@SARATHY.COM
Independent Researcher

Editor: (Editor information will be added upon acceptance)

Abstract

VSAX is an open-source Python library for Vector Symbolic Architectures (VSAs) with GPU acceleration via JAX. It provides complete implementations of three major VSA models (FHRR, MAP, Binary) under a unified API, advanced capabilities for compositional reasoning and continuous encoding (Clifford operators, Spatial Semantic Pointers, Vector Function Architecture), and production-ready quality with 94% test coverage and comprehensive documentation. VSAX enables researchers to prototype VSA systems rapidly, compare models fairly, and deploy to production with confidence. The library is available via PyPI (`pip install vsax`) and GitHub (`https://github.com/vasanthsarathy/vsax`) under the MIT license.

Keywords: vector symbolic architectures, hyperdimensional computing, GPU acceleration, JAX, machine learning

1 Introduction

Vector Symbolic Architectures (VSAs), also known as Hyperdimensional Computing, represent symbolic information in high-dimensional vector spaces (Kanerva, 2009; Plate, 1995). Concepts are encoded as randomly generated high-dimensional vectors (hypervectors) and combined using algebraic operations for compositional reasoning, analogical inference, and cognitive modeling (Kleyko et al., 2022, 2023).

Despite growing interest across robotics, edge computing, and cognitive science, VSA research faces significant tooling challenges. Existing implementations are fragmented across frameworks (PyTorch, NumPy), provide incomplete model coverage (missing FHRR or advanced operators), lack production-quality engineering (low test coverage, minimal documentation), and offer limited GPU acceleration. This fragmentation raises barriers for newcomers and hinders reproducibility for experienced researchers.

VSAX addresses these issues through: (1) **unified implementations** of all major VSA models (FHRR, MAP, Binary) with consistent APIs enabling fair comparisons, (2) **advanced capabilities** unavailable elsewhere (Clifford operators for exact compositional reasoning, complete Spatial Semantic Pointers, Vector Function Architecture), (3) **production-ready quality** with 94% test coverage, full type safety, and comprehensive documentation (11 tutorials, 9 user guides), and (4) **GPU acceleration** via JAX with 5-30× speedups over CPU implementations.

Installation is straightforward via PyPI:

```
pip install vsax
```

A minimal example demonstrates the unified API:

```python
import jax
from vsax import create_fhrr_model, VSAMemory

# Create model and symbol table
model = create_fhrr_model(dim=1024, key=jax.random.PRNGKey(0))
memory = VSAMemory(model)

# Add basis vectors
memory.add_many(["cat", "mat", "on"])

# Bind and bundle operations
cat_on_mat = model.opset.bind(
    model.opset.bind(memory["cat"].vec, memory["on"].vec),
    memory["mat"].vec
)
```

2 Package Design and Implementation

2.1 Core Functionality and Unified API

VSAX implements three major VSA models: FHRR (Fourier Holographic Reduced Representations) (Plate, 1995) uses complex-valued vectors with circular convolution via FFT for exact unbinding; MAP (Multiply-Add-Permute) uses real-valued vectors with element-wise multiplication for computational efficiency; Binary VSA uses bipolar vectors with XOR for binding, optimized for hardware. All models share a consistent API—switching models requires changing only the factory function:

```python
# Compare models by changing one line
model = create_fhrr_model(dim=1024)
# model = create_map_model(dim=1024)
# model = create_binary_model(dim=10000)

# All downstream code remains identical
memory = VSAMemory(model)
encoder = DictEncoder(model, memory)
result = model.opset.bind(vec1, vec2)
```

2.2 Advanced Capabilities

VSAX provides several advanced capabilities beyond standard VSA operations. **Clifford operators** (Aerts et al., 2007) enable phase-based transformations with near-perfect invertibility (similarity > 0.999) compared to traditional unbinding (0.3-0.6 similarity), critical for precise compositional reasoning. **Spatial Semantic Pointers** (Komer et al., 2019) encode continuous spatial coordinates as $S(x, y) = X^x \otimes Y^y$ using Fractional Power Encoding, enabling object-location binding and bidirectional spatial queries for robotic navigation and scene understanding. The **Vector Function Architecture** (Frady et al., 2021) represents functions as hypervectors in a Reproducing Kernel Hilbert Space, supporting function evaluation, arithmetic, and transformations for density estimation and nonlinear regression. **Resonator networks** (Kent et al., 2020; Frady et al., 2020) factorize superimposed hypervectors through iterative coupled dynamics, essential for parsing complex compositional structures.

2.3 Software Architecture

VSAX follows clean architecture principles with modular, immutable, and type-safe design. The core abstraction separates representations (`AbstractHypervector`) from operations (`AbstractOpSet`), enabling arbitrary combinations and easy extension. All data structures are immutable following JAX's functional paradigm, ensuring thread safety and enabling JIT optimization. Full type annotations verified by mypy in strict mode provide IDE support and catch errors at development time.

The library is organized into focused modules: `vsax.core` (base abstractions, factory functions), `vsax.representations` (Complex/Real/Binary hypervectors), `vsax.ops` (FHRR/MAP/Binary operations), `vsax.encoders` (Scalar/Sequence/Dict/Graph/FPE encoders), `vsax.operators` (Clifford operators), `vsax.spatial` (Spatial Semantic Pointers), `vsax.vfa` (Vector Function Architecture), `vsax.resonator` (factorization networks), `vsax.similarity` (distance metrics), and `vsax.io` (persistence).

Users extend VSAX by subclassing abstractions and implementing required methods, seamlessly integrating custom components with existing functionality.

2.4 Performance and GPU Acceleration

Built on JAX (Bradbury et al., 2018), VSAX achieves substantial performance improvements through GPU acceleration, JIT compilation, and automatic vectorization. Benchmarks on an NVIDIA RTX 4090 with 10,000-dimensional vectors demonstrate 5-30× speedups over CPU: binding operations achieve 26× speedup (0.31ms GPU vs 8.1ms CPU), bundling shows 18× speedup (0.52ms vs 9.3ms CPU), and resonator factorization achieves 20× speedup. JAX's `vmap` enables efficient batch operations for parallel processing of datasets. The library supports CUDA and TPU backends with automatic device placement.

3 Comparison with Existing Tools

Table 1 compares VSAX with major VSA libraries. Torchhd (Heddes et al., 2023) provides excellent PyTorch-based implementations with multiple models and GPU support, but lacks Clifford operators, complete Spatial Semantic Pointers, and Vector Function Architecture. hdlib offers basic CPU-only implementations with limited documentation. PyBHV focuses exclusively on Binary hypervectors with narrow scope. VSAX uniquely provides Clifford operators for exact compositional reasoning, complete SSP implementation with full 2D/3D scene encoding, Vector Function Architecture for RKHS function encoding, full resonator networks with iterative convergence, and JAX-based automatic differentiation for hybrid neuro-symbolic models.

4 Software Engineering and Community

VSAX maintains 618 automated tests achieving 94% code coverage, including unit tests for individual operations, integration tests for end-to-end workflows, and property-based tests for algebraic invariants. Continuous integration via GitHub Actions tests on Linux, macOS, and Windows for Python 3.9-3.12, with type checking (mypy), linting (ruff), and coverage reporting. Code quality is enforced through automated tooling and pre-commit hooks.

Table 1: Feature comparison with major VSA libraries (January 2025)

Feature	VSAX	Torchhd	hdlib	PyBHV
FHRR/HRR	✓	✓	×	×
MAP (Real)	✓	✓	×	×
Binary	✓	✓	✓	✓
GPU Acceleration	✓	✓	×	×
Clifford Operators	✓	×	×	×
Spatial Semantic Pointers	✓	×	×	×
Vector Function Architecture	✓	×	×	×
Resonator Networks	Full	Single-step	×	×
Framework	JAX	PyTorch	NumPy	Multiple
Test Coverage	94%	85%	60%	70%
Tutorials	11	Many	2	3

API documentation is auto-generated from comprehensive docstrings using mkdocstrings with the Google format. Every public function includes purpose, parameters, return types, usage examples, and exception documentation. The library includes 11 hands-on tutorials with real datasets (MNIST classification, knowledge graph reasoning, analogical reasoning, Clifford operators, SSP, VFA) and 9 topic-specific user guides covering models, encoders, spatial encoding, and performance tuning.

Development occurs transparently on GitHub with issues tracking bugs and features, pull requests welcome with contribution guidelines, and semantic versioning for API stability. The `save_basis` and `load_basis` functions enable reproducible experiments by serializing named hypervectors. The MIT license enables both academic research and commercial deployment without restrictions.

VSAX has been publicly available on PyPI since December 2024 with active downloads and growing community engagement. The library is used in academic courses on cognitive architectures and production deployments for edge computing and robotic navigation.

5 Conclusion

VSAX provides the VSA community with comprehensive, production-ready software combining research capabilities with software engineering best practices. By unifying three VSA models under a consistent API, providing GPU acceleration through JAX, and introducing novel capabilities (Clifford operators, continuous encoding, resonator networks), VSAX lowers barriers to VSA research and enables confident deployment. The library's MIT license, extensive documentation, strong testing, and active development make it suitable for academic research, education, and production applications. Future enhancements include non-commutative operator algebras, learned encoder bases, multi-dimensional VFA, probabilistic extensions, and neuromorphic backends.

Acknowledgments

We thank the JAX team at Google for the excellent framework underlying VSAX's performance and the VSA research community for feedback on design and functionality.

References

Diederik Aerts, Marek Czachor, Liane Gabora, Maciej Kuna, Andrzej Posiewnik, Jaroslaw Pykacz, and Marek Syty. Quantum structure in cognition: fundamentals and applications. *arXiv preprint arXiv:0805.3850*, 2007.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. Jax: composable transformations of python+numpy programs, 2018. URL `https://github.com/google/jax`.

E Paxon Frady, Spencer J Kent, Bruno A Olshausen, and Friedrich T Sommer. Resonator networks, 2: Factorization performance and capacity compared to optimization-based methods. *Neural Computation*, 32(12):2332–2388, 2020.

E Paxon Frady, Denis Kleyko, and Friedrich T Sommer. Computing on functions using randomized vector representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9139–9152, 2021.

Mike Heddes, Igor Nunes, Pere Vergés, Denis Kleyko, Danny Abraham, Tony Givargis, Alexandru Nicolau, and Alexander Veidenbaum. Torchhd: An open source python library to support research on hyperdimensional computing and vector symbolic architectures. *Journal of Machine Learning Research*, 24(255):1–10, 2023.

Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1:139–159, 2009.

Spencer J Kent, E Paxon Frady, Friedrich T Sommer, and Bruno A Olshausen. Resonator networks, 1: An efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural Computation*, 32(12):2311–2331, 2020.

Denis Kleyko, Dmitri A Rachkovskij, Evgeny Osipov, and Abbas Rahimi. A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations. *ACM Computing Surveys*, 55(6):1–40, 2022.

Denis Kleyko, Dmitri A Rachkovskij, Evgeny Osipov, and Abbas Rahimi. A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges. *ACM Computing Surveys*, 55(9):1–52, 2023.

Brent Komer, Terrence C Stewart, Aaron R Voelker, and Chris Eliasmith. A neural representation of continuous space using fractional binding. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society*, pages 2041–2047, 2019.

Tony A Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.