The Journal of Open Source Software

hdlib 2.0: extending machine learning capabilities of Vector-Symbolic Architectures

Fabio Cumbo [1¶], **Kabir Dhillon** [2], and **Daniel Blankenberg** [1,3]

1 Center for Computational Life Sciences, Cleveland Clinic Research, Cleveland Clinic, Cleveland, Ohio, United States of America **2** College of Engineering, Ohio State University, Columbus, Ohio, United States of America **3** Department of Molecular Medicine, Cleveland Clinic Lerner College of Medicine, Case Western Reserve University, Cleveland, Ohio, United States of America ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 23 September 2025
Published: unpublished

Summary

Following the initial publication of *hdlib* (Cumbo et al., 2023), a Python library for designing Vector-Symbolic Architectures (VSA), we introduce a major extension that significantly enhances its machine learning capabilities. VSA, also known as Hyperdimensional Computing, is a computing paradigm that represents and processes information using high-dimensional vectors. While the first version of *hdlib* established a robust foundation for creating and manipulating these vectors, this update addresses the growing need for more advanced, data-driven modeling within the VSA framework. This paper describes three key extensions: a regression model for predicting continuous variables, a clustering model for unsupervised learning, a module for encoding graph-based data structures, and significant enhancements to the existing supervised classification model also enabling feature selection.

The library's code remains open source and available on GitHub at https://github.com/cumbof/hdlib under the MIT license and is distributed through the Python Package Index (*pip install hdlib*) and Conda (*conda install -c conda-forge hdlib*). Documentation and examples of these new features are available at https://github.com/cumbof/hdlib/wiki.

Statement of need

The successful application of VSA across diverse scientific domains has created a demand for more sophisticated machine learning models that go beyond basic classification. Researchers now require tools to tackle regression tasks, model complex relationships in structured data like graphs, and better optimize models by identifying the most salient features.

This new version of *hdlib* directly addresses this need. While other libraries provide foundational VSA operations (Heddes et al., 2023; Kang et al., 2022; Simon et al., 2022), *hdlib* now introduces a cohesive toolkit for advanced machine learning that is, to our knowledge, unique in its integration of regression, clustering, graph encoding, and enhanced feature selection within a single, flexible VSA framework. These additions empower researchers to move from rapid prototyping of core VSA concepts to building and evaluating complex, end-to-end machine learning pipelines that are now used in the context of different problems in different scientific domains (Cumbo et al., 2020; Cumbo, Truglia, et al., 2025; Cumbo, Dhillon, Joshi, Chicco, et al., 2025; Cumbo, Dhillon, Joshi, Raubenolt, et al., 2025; Cumbo & Chicco, 2025; Joshi et al., 2025).

Extending Machine Learning functionalities

The primary contribution of this work is the expansion of the *hdlib.model* module with new functionalities to enhance existing methods and the introduction of new modules for handling different data structures. The new architecture is summarized in Figure 1.

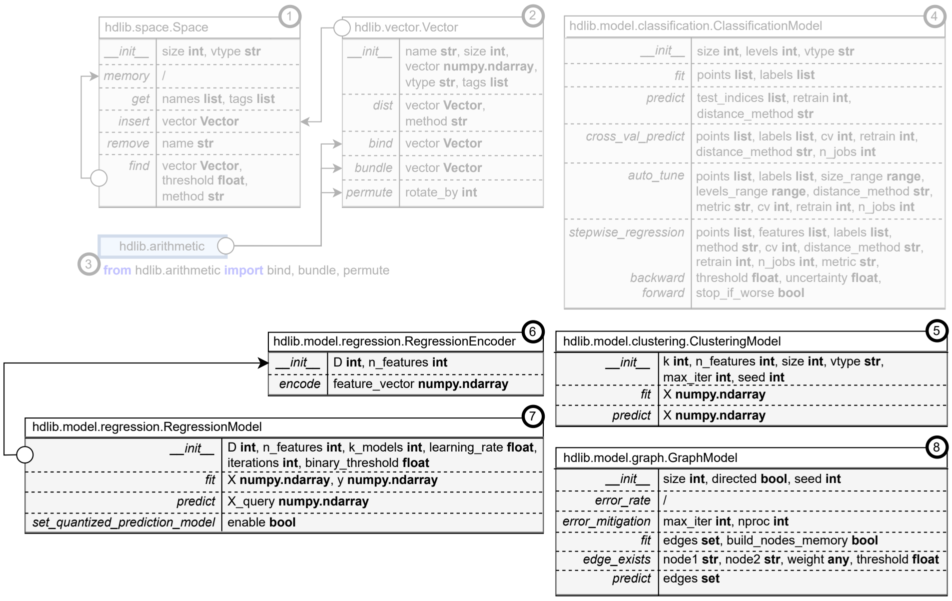

Figure 1: An overview of the *hdlib* 2.0 library architecture, highlighting the distinction between the original (top, transparent) and new components (bottom). Foundational classes from version 1.0 include `hdlib.space.Space` (Class 1), `hdlib.vector.Vector` (Class 2), `hdlib.arithmetic` module (Class 3), and the `hdlib.model.classification.ClassificationModel` (Class 4). This work introduces major new functionalities through the `hdlib.model` module comprising the new `clustering.ClusteringModel` (Class 5), `regression.RegressionEncoder` (Class 6) and `regression.RegressionModel` (Class 7), and `graph.GraphModel` (Class 8), creating a comprehensive toolkit for VSA-based machine learning.

Classification Model

A key focus of this update was to provide more robust and automated tools for model optimization:

- **Enhanced feature selection**: the original `hdlib.model.Model` class (now `hdlib.model.classificat` provided a `stepwise_regression` instance method for feature selection. This functionality has been significantly enhanced to offer greater control over the selection process, improved performance, and more detailed reporting on feature importance. This refinement helps in building more interpretable VSA models;
- **Advanced hyperparameter tuning**: the initial version of the library included an `auto_tune` instance method for performing a parameter sweep analysis on vector dimensionality and the number of level vectors. This has been upgraded to a more advanced hyperparameter optimization tool. The new implementation is more efficient and allows for a more thorough and effective search of the hyperparameter space to automatically maximize the model performances.

Clustering Model

Here, we introduced a new `hdlib.model.clustering` module that provides a `ClusteringModel` class that implements a k-means clustering algorithm working accordingly with the Hyperdimensional Computing principles as defined in (Gupta et al., 2022).

The algorithm operates by representing both the k cluster centroids and the input data points as hypervectors. The iterative `fit` process closely mirrors the classic k-means algorithm but uses VSA operations. In the assignment step of each iteration, data points are assigned to the cluster corresponding to the most similar centroid, determined by calculating the cosine distance in the high-dimensional space. In the subsequent update step, the centroid of each cluster is recalculated by performing a bundling operation (element-wise addition and normalization) on all the hypervectors of the data points assigned to it. This process naturally moves the centroid towards the center of its constituent points. This iterative process continues until the cluster assignments stabilize or a maximum number of iterations is reached. Once the model is trained, the `predict` method can be used to assign a new, unseen data point to the most appropriate cluster.

Regression Model

To address tasks involving the prediction of continuous variables, `hdlib` now implements a regression model based on the methodology described by (Hernández-Cano et al., 2021). This implementation is split into two main components: a `RegressionEncoder` and a `RegressionModel` as part of the `hdlib.model.regression` module. The encoder maps input features into a high-dimensional space using a non-linear function that combines the input with a set of random base hypervectors and biases. This mapping is specifically designed to preserve the similarity relationships of the original feature space.

The `RegressionModel` employs a sophisticated multi-model strategy, maintaining a set of k parallel cluster models and regression models. During the iterative `fit` process, an encoded input vector is compared against all cluster models to compute a set of confidence scores via a `softmax` function. A final prediction is produced by a confidence-weighted sum of the outputs from all regression models. The prediction error is then used to update the models: all regression models are adjusted based on their confidence score, while only the most similar cluster model is refined. This process allows the system to learn complex, non-linear relationships in the data. For efficiency, the module can maintain both full-precision and binarized versions of the models, and users can enable a `quantized_prediction` mode for accelerated inference using Hamming distance. This enables VSA to be applied to a new class of problems, such as predicting physical properties, financial values, or other scalar quantities.

Graph Model

A major extension in this release is the `hdlib.model.graph` module, which provides the `GraphModel` class for representing and reasoning with graph-based data. This implementation encodes an entire directed and undirected weighted graph into a single hyperdimensional vector, based on the methodology described by (Poduval et al., 2022). The process begins by assigning a unique random hypervector to each node and edge weight. The `fit` method then constructs the graph representation by first creating a memory vector for each node that encodes its local neighborhood. This is achieved by bundling the vectors of its neighbors, each binded with their respective edge-weight vector. Finally, the entire graph is compressed into one vector by bundling all node vectors, each binded with its corresponding memory vector. For directed graphs, a permute operation is used to preserve the directionality of edges within the final representation.

Crucially, the library can query the existence of an edge directly from this single graph vector. The `edge_exists` method uses binding operations to probe the graph vector, retrieve a noisy version of a node's memory, and check its similarity to a potential neighbor. Furthermore, the module includes a `predict` method for edge weight classification and an `error_mitigation` routine for iteratively refining the graph model to reduce prediction errors, making it a complete toolkit for graph-based machine learning.

With the integration of these modules, `hdlib` 2.0 provides the scientific community with a unified and powerful framework, paving the way for the development of novel, brain-inspired solutions to a broader spectrum of machine learning problems.

References

Cumbo, F., Cappelli, E., & Weitschek, E. (2020). A brain-inspired hyperdimensional computing approach for classifying massive DNA methylation data of cancer. *Algorithms*, *13*(9), 233. https://doi.org/10.3390/a13090233

Cumbo, F., & Chicco, D. (2025). Hyperdimensional computing in biomedical sciences: A brief review. *PeerJ Computer Science*, *11*, e2885. https://doi.org/10.7717/peerj-cs.2885

Cumbo, F., Dhillon, K., Joshi, J., Chicco, D., Aygun, S., & Blankenberg, D. (2025). A novel vector-symbolic architecture for graph encoding and its application to viral pangenome-based species classification. *bioRxiv*, 2025–2009. https://doi.org/10.1101/2025.09.08.674958

Cumbo, F., Dhillon, K., Joshi, J., Raubenolt, B., Chicco, D., Aygun, S., & Blankenberg, D. (2025). Predicting the toxicity of chemical compounds via hyperdimensional computing. *bioRxiv*, 2025–2009. https://doi.org/10.1101/2025.09.12.675894

Cumbo, F., Truglia, S., Weitschek, E., & Blankenberg, D. (2025). Feature selection with vector-symbolic architectures: A case study on microbial profiles of shotgun metagenomic samples of colorectal cancer. *Briefings in Bioinformatics*, *26*(2), bbaf177. https://doi.org/10.1093/bib/bbaf177

Cumbo, F., Weitschek, E., & Blankenberg, D. (2023). Hdlib: A python library for designing vector-symbolic architectures. *Journal of Open Source Software*, *8*(89), 5704. https://doi.org/10.21105/joss.05704

Gupta, S., Khaleghi, B., Salamat, S., Morris, J., Ramkumar, R., Yu, J., Tiwari, A., Kang, J., Imani, M., Aksanli, B., & others. (2022). Store-n-learn: Classification and clustering with hyperdimensional computing across flash hierarchy. *ACM Transactions on Embedded Computing Systems (TECS)*, *21*(3), 1–25. https://doi.org/10.1145/3503541

Heddes, M., Nunes, I., Vergés, P., Kleyko, D., Abraham, D., Givargis, T., Nicolau, A., & Veidenbaum, A. (2023). 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. http://jmlr.org/papers/v24/23-0300.html

Hernández-Cano, A., Zhuo, C., Yin, X., & Imani, M. (2021). Reghd: Robust and efficient regression in hyper-dimensional learning system. *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 7–12. https://doi.org/10.1109/DAC18074.2021.9586284

Joshi, J., Cumbo, F., & Blankenberg, D. (2025). Large-scale classification of metagenomic samples: A comparative analysis of classical machine learning techniques vs a novel brain-inspired hyperdimensional computing approach. *bioRxiv*, 2025–2007. https://doi.org/10.1101/2025.07.06.663394

Kang, J., Khaleghi, B., Rosing, T., & Kim, Y. (2022). Openhd: A gpu-powered framework for hyperdimensional computing. *IEEE Transactions on Computers*, *71*(11), 2753–2765. https://doi.org/10.1109/TC.2022.3179226

Poduval, P., Alimohamadi, H., Zakeri, A., Imani, F., Najafi, M. H., Givargis, T., & Imani, M. (2022). Graphd: Graph-based hyperdimensional memorization for brain-like cognitive learning. *Frontiers in Neuroscience*, *16*, 757125. https://doi.org/10.3389/fnins.2022.

757125

Simon, W. A., Pale, U., Teijeiro, T., & Atienza, D. (2022). Hdtorch: Accelerating hyperdimensional computing with gp-gpus for design space exploration. *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 1–8. https://doi.org/10.1145/3508352.3549475