




DiRe - JAX: A JAX based Dimensionality Reduction Algorithm for Large-scale Data

Alexander Kolpakov ^{1*} and Igor Rivin ^{2*}

¹ University of Austin, Austin TX, USA; akolpakov@uaustrin.org ² Temple University, Philadelphia PA, USA; rivin@temple.edu ¶ Corresponding author * These authors contributed equally.

DOI: [10.21105/joss.08264](https://doi.org/10.21105/joss.08264)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Neea Rusch](#) 

Reviewers:

- [@Treys925](#)
- [@crhea93](#)

Submitted: 21 May 2025

Published: 07 June 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

DiRe - JAX is a new dimensionality reduction toolkit designed to address some of the challenges faced by traditional methods like UMAP and tSNE such as loss of global structure and computational efficiency. Built on the JAX framework, DiRe leverages modern hardware acceleration to provide an efficient, scalable, and interpretable solution for visualizing complex data structures and for quantitative analysis of lower-dimensional embeddings. The toolkit shows considerable promise in preserving both local and global structures within the data as compared to state-of-the-art UMAP and tSNE implementations. This makes it suitable for a wide range of applications in machine learning, bioinformatics, and data science.

Statement of need

Traditional dimensionality reduction techniques such as UMAP and tSNE are widely used for visualizing high-dimensional data in lower-dimensional spaces, usually 2D and sometimes 3D. Other uses include dimensionality reduction to other, possibly higher and thus non-visual dimensions, for the subsequent use of classifiers such as SVMs.

However, these methods often struggle with scalability, interpretability, and preservation of global data structures. For example, while fast and scalable, UMAP may overemphasize local structures at the expense of global data relationships ([Chari & Pachter, 2023](#)). And tSNE, while known for producing high-quality visualizations, may be computationally expensive and sensitive to hyperparameter tuning ([Kobak & Berens, 2019](#)).

DiRe-JAX addresses these challenges by offering a scalable solution that balances the preservation of both local and global structures. Leveraging the JAX framework allows DiRe-JAX to efficiently handle large datasets by utilizing GPU/TPU acceleration, making it significantly faster than CPU-based implementations without compromising on the quality of the embeddings.

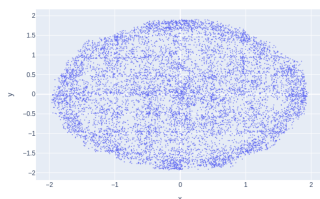
DiRe-JAX also includes a wealth of metrics for analyzing embedding quality and for hyperparameter tuning. Given its runtime efficiency, tasks such as grid-search hyperparameter optimization become feasible even in low-cost environments like Google Colab.

This makes DiRe-JAX an essential toolkit for researchers and practitioners working with complex, high-dimensional data.

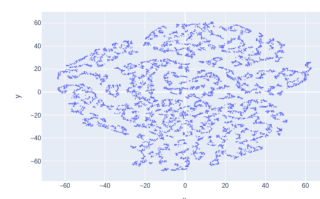
Benchmarks

A few benchmark below provide some visuals regarding the global structure changes or preservation by various methods, such as

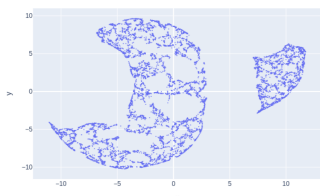
Dataset: Disk Uniform



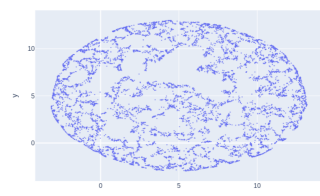
Disk DiRe-JAX embedding



Disk tSNE embedding

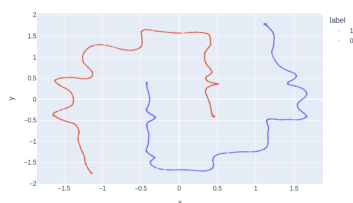


Disk cuML UMAP embedding

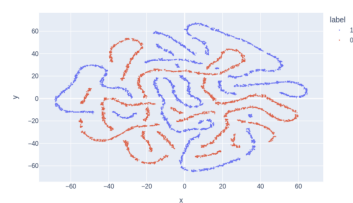


Disk UMAP embedding

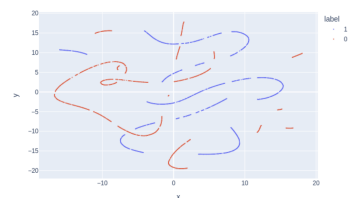
Dataset: Two Half-Moons



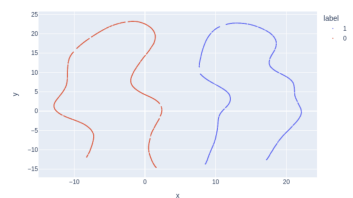
Moons DiRe-JAX embedding



Moons tSNE embedding

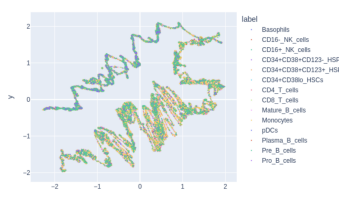


Moons cuML UMAP embedding

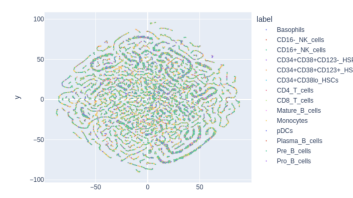


Moons UMAP embedding

Dataset: Levine 32



Levine 32 DiRe-JAX embedding



Levine 32 tSNE embedding



Levine 32 cuML UMAP embedding



Levine 32 UMAP embedding

Main methods

The main class of DiRe-JAX is DiRe. Let $X \subset \mathbb{R}^n$ be the input data realized as a NumPy array. Then, DiRe performs the following main steps:

1. **Capturing dataset topology:** Create the kNN graph of X , say Γ , for a given number of neighbors k (`n_neighbors`) by calling `make_knn_adjacency`. This step uses a JAX kernel specifically developed by the authors to perform the computation on CPU, GPU, or TPU settings. Other libraries like FAISS ([Douze et al., 2024](#)) may also be used, although they do not provide the same hardware universality.
2. **Initial dimension reduction:** Produce $Y \subset \mathbb{R}^d$, the initial embedding of X , with $d \ll n$ (usually $d = 2$ or 3) given by the dimension parameter, using one of the available embedding methods: `random` (random projections from the Johnson–Lindenstrauss Lemma), `spectral` (using the kNN graph Γ to construct the weighted Laplacian, optionally applying a similarity kernel), or `pca` (classical or kernel-based).
3. **Layout optimization:** Call `do_layout` to adjust the lower-dimensional embedding Y to conform to the similarity structure of the higher-dimensional data X . This is done via a force-directed layout where the role of “forces” is played by probability kernels (the distributions can be adjusted via parameters `min_dist` and `spread`).

The initial embedding is stored in `self.init_embedding`, while the optimized layout is stored in `self.layout`. Both can be accessed after the main method `fit_transform` is called, for detailed comparison and analysis.

Random Projection embedding

This embedding is based on the following classical Johnson–Lindenstrauss Lemma ([Johnson & Lindenstrauss, 1984](#)):

Johnson–Lindenstrauss Lemma (Probabilistic Form)

Given $0 < \epsilon < 1$ and an integer n , let X be a set of n points in \mathbb{R}^d . For a random linear map $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ where $k = O\left(\frac{\log n}{\epsilon^2}\right)$, with high probability, for all $u, v \in X$:

$$(1 - \epsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon)\|u - v\|^2.$$

The value

$$\text{dist}(f) = \frac{\|f(u) - f(v)\|}{\|u - v\|}$$

is called the *distortion* of f , and is expected to be close to 1.0 for a high-quality embedding.

Random projections are simple and computationally inexpensive, but can suffer from cluttered outputs when $k \ll d$ due to variance reduction in the projected data.

Principal Component Analysis embedding

PCA seeks to preserve as much dataset variance as possible by projecting onto the top k singular vectors. Assume X is column-centered, with covariance matrix $\text{Cov}(X) = \frac{1}{n-1} X^T X$. Compute the singular value decomposition:

$$X = U \Sigma W^T,$$

and truncate to the top k singular values Σ_k and vectors W_k . The rank- k approximation is:

$$\widehat{X} = U_k \Sigma_k W_k^T,$$

and the PCA embedding is:

$$X_k = X W_k.$$

Under mild conditions on the spectrum, the bottleneck distance between persistence diagrams of X and X_k satisfies (Chazal et al., 2014):

$$d_b(D(X), D(X_k)) \leq \|\widehat{X} - X\|_F = \varepsilon \|X\|_F.$$

Thus, PCA approximately preserves topological features up to controlled error.

Spectral Laplacian embedding

Using the kNN graph Γ of X , construct the graph Laplacian (optionally with a similarity kernel) and compute the bottom- k eigenvectors for embedding. This method often captures manifold structure but may not preserve global relationships.

Force-directed layout

After obtaining an initial embedding Y , DiRe-JAX applies an iterative force-directed layout to align Y 's local structure with that of X . Attraction and repulsion forces are modeled after tSNE and UMAP kernels:

$$\varphi(x) = \frac{1}{1 + a\|x\|^{2b}},$$

tuned by $\text{min_dist} = \delta$ and $\text{spread} = \sigma$ so that:

- $\varphi(x) \approx 1.0$ for $\|x\| < \delta$, and
- $\varphi(x) \approx \exp(-(\|x\| - \delta)/\sigma)$ otherwise.

Attraction forces apply to kNN neighbors in Γ , while all other pairs experience repulsion. Layout iterations run until a preset number of steps is reached.

Code availability

The DiRe - JAX repository is available on [GitHub](https://github.com). An installable package is available on [PyPI](https://pypi.org).

Whitepaper

The DiRe - JAX whitepaper is available on the [arXiv](#) preprint server.

Acknowledgements

The authors would like to thank [@Treys925](#) (Trey Smith, University of Michigan, Ann Arbor), [@crhea93](#) (Carter Lee Rhea, Université de Montréal), and [@lmcinnes](#) (Leland McInnes, Tutte Institute for Mathematics and Computing) for their helpful comments and suggestions. This work is supported by the Google Cloud Research Award number GCP19980904.

References

- Chari, T., & Pachter, L. (2023). The specious art of single-cell genomics. *PLoS Computational Biology*, 19(8), e1011288. <https://doi.org/10.1371/journal.pcbi.1011288>
- Chazal, F., Silva, V. de, & Oudot, S. (2014). Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1), 193–214. <https://doi.org/10.1007/s10711-013-9937-z>
- Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., & Jégou, H. (2024). *The FAISS library*. <https://arxiv.org/abs/2401.08281>
- Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26, 189–206. <https://doi.org/10.1090/conm/026>
- Kobak, D., & Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. *Nature Communications*, 10(1), 5416. <https://doi.org/10.1038/s41467-019-13056-x>